TRANSFORMERS CAN ACHIEVE LENGTH GENERALIZATION BUT NOT ROBUSTLY

Yongchao Zhou[1,2] *, **Uri Alon**[1], **Xinyun Chen**[1], **Xuezhi Wang**[1], **Rishabh Agarwal**[1], **Denny Zhou**[1]
[1]Google DeepMind, [2] University of Toronto

ABSTRACT

Length generalization, defined as the ability to extrapolate from shorter training sequences to longer test ones, is a significant challenge for language models. This issue persists even with large-scale Transformers handling relatively straightforward tasks. In this paper, we test the Transformer's ability of length generalization using the task of addition of two integers. We show that the success of length generalization is intricately linked to the data format and the type of position encoding. Using the right combination of data format and position encodings, we show for the first time that standard Transformers can extrapolate to a sequence length that is $2.5\times$ the input length. Nevertheless, unlike in-distribution generalization, length generalization remains fragile, significantly influenced by factors like random weight initialization and training data order, leading to large variances across different random seeds.

1 INTRODUCTION

Transformer-based models have revolutionized natural language understanding and generation across diverse applications (Gemini et al., 2023). Despite their impressive abilities in mathematical reasoning (Lewkowycz et al., 2022), code synthesis (Li et al., 2022), and theorem proving (Wu et al., 2022), Transformers often struggle with length generalization, an ability that requires the model to generalize to longer sequences than seen during training (Anil et al., 2022; Abbe et al., 2023; Zhou et al., 2024). This limitation raises an essential question: do Transformers genuinely grasp the correct underlying algorithms for a given task, or are they merely resorting to superficial memorization or shortcut strategies that fail to scale to more complex problems Liu et al. (2023b)?

Recent work has scrutinized Transformers' shortcomings in length generalization across formal language learning Deletang et al. (2023) and algorithmic reasoning tasks Anil et al. (2022); Zhang et al. (2022); Dziri et al. (2023). These investigations consistently indicate a notable deficiency in length generalization capabilities. This recurring issue raises a crucial question: Is there an inherent limitation in Transformers' design preventing effective length generalization?

In this paper, we systematically examine the Transformer's capability of length generalization, specifically focusing on the N-digit decimal addition problem. We view the addition problem as a form of synthetic language learning, which despite its relative simplicity compared to natural language, provides valuable insights into the Transformer's ability to internalize fundamental algorithms. Notwithstanding its simplicity, recent work has demonstrated that Transformers exhibit limited length generalization in this task (Lee et al., 2023; Shen et al., 2023).

Previous attempts to improve Transformer's length generalization ability primarily focus on two areas: refining position encodings (Shen et al., 2023) and optimizing data formats (Lee et al., 2023; Zhou et al., 2024). Therefore, we perform an extensive empirical evaluation of combinations of widely used position encoding and various data formats, resulting in a recipe for successful length generalization. Our final recipe consists of: FIRE position encodings (Li et al., 2024), with randomized positions (Ruoss et al., 2023), in reversed format, with index hints (Zhou et al., 2024).

*Student Researcher at Google Deepmind. Corresponding authors: <yczhou@cs.toronto.edu>, <dennyzhou@google.com>

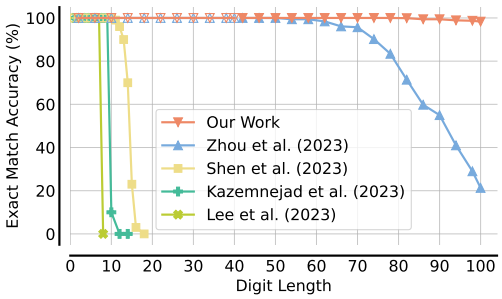


Figure 1: Using an appropriate position encoding and data formatting, we demonstrate that Transformers can generalize to 100-digit decimal addition tasks with more than 98% of accuracy when trained up to 40-digit addition, resulting in a length extension ratio of $2.5\times$, which is much more than the ratio of Lee et al. (2023) ($1.0\times$), Kazemnejad et al. (2023) ($1.125\times$), Shen et al. (2023) ($1.1\times$), and Zhou et al. (2024) ($1.5\times$).

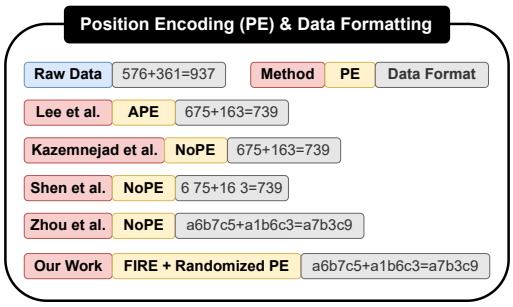


Figure 2: Comparative overview of PEs and data formats: While most related studies focus on APE or NoPE, our approach integrates FIRE (Li et al., 2024) and Randomized PE (Ruoss et al., 2023). All studies utilize a reversed format. Shen et al. (2023) enhance this with random space augmentation, and both Zhou et al. (2024) and Our Work incorporate index hints.

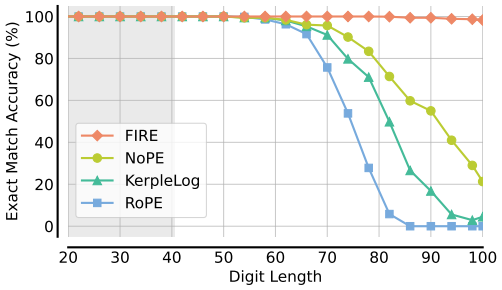


Figure 3: EM accuracy (best of 10 trials), trained exclusively on sequences of lengths 1 to 40, the best trials involving FIRE exhibit near-perfect generalization on 100-digit addition.

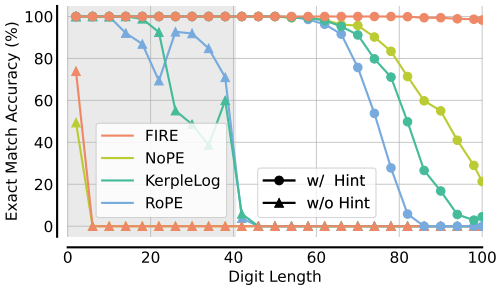


Figure 4: EM accuracy of models trained with and without index hints (best of 10 trials): Without index hints, all PE methods fail in generalization, both within and beyond trained lengths.

As shown in Figure 1, when trained on only 40 digits, our model successfully extrapolates to sequences of up to 100 digits, exceeding the input length by $2.5\times$. To the best of our knowledge, this is the strongest known generalization result for text-based Transformers on addition. Nevertheless, we observe that the robustness of this length generalization is fragile, significantly swayed by variables such as random initialization and the training data order.

Our key contributions are summarized as follows:

(i) We demonstrate that the success in length generalization is markedly influenced by position encoding and data format. Through careful selection of these factors, we achieved extrapolation to lengths that are $2.5\times$ longer than those seen during training.

(ii) Our exploration of established data formatting and augmentation techniques indicates that their effectiveness in length generalization is primarily contingent on the choice of position encoding.

(iii) Despite remarkable generalization to lengths $2.5\times$ longer than training, we found this generalization to be fragile and heavily relying on factors like random weight initialization and training data order.

2 Experiments

FIRE enables significantly better length generalization. Figure 3 compares the length generalization capabilities of four positional encodings in the best of 10 trials (See Appendix F.1 for all

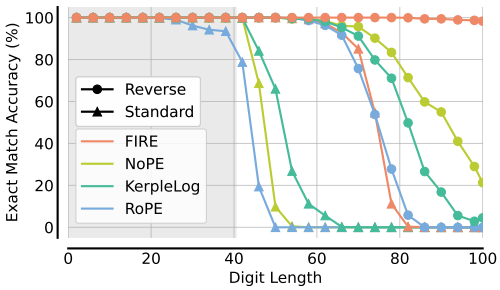


Figure 5: EM accuracy of the standard vs. the reversed format: Consistently with prior studies, the reversed format excels over the standard format across all PEs.

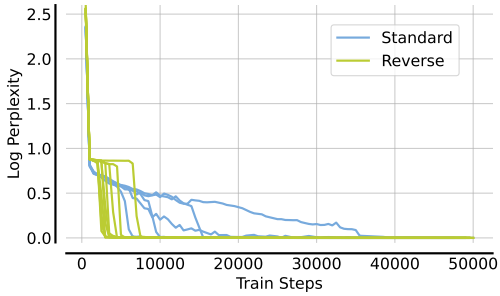


Figure 6: The reversed format shows distinct grokking during training, unlike the gradual enhancement in the standard format. This phenomenon is observed across all PEs (Figure F.3)

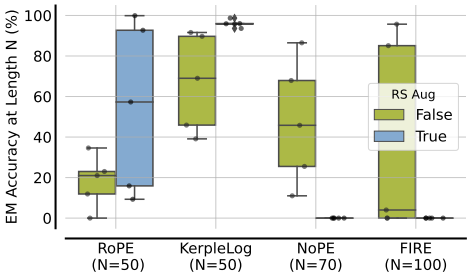


Figure 7: Effects of Random Space Augmentation (RS Aug): Random space augmentation is beneficial for RoPE and KerpleLog; adverse for NoPE and FIRE.

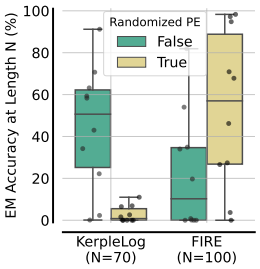


Figure 8: Effects of Randomized PE: Randomized PE enhances FIRE but degrades KerpleLog

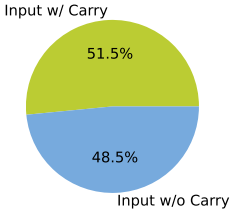


Figure 9: Error Distribution: Errors appear almost equally with and without carry.

trials). Trained exclusively on sequences of lengths 1-40, the best trial of FIRE exhibit near-perfect generalization to sequences up to the length of 100. In contrast, other PEs show a visible degradation in generalization accuracy beyond the sequence length of 60. This finding counters the findings of Kazemnejad et al. (2023) that no positional encoding (NoPE) surpasses complex PE techniques for length generalization. Our findings suggest that a well-designed PE, such as FIRE, is essential for optimal length generalization.

Index hints are crucial. We compare models trained with and without index hints. As shown in Figure 4, index hints significantly enhance length generalization across various PEs, corroborating the findings of Zhou et al. (2024). Notably, without index hints, NoPE and FIRE demonstrate poor in-distribution generalization for 40-digit additions, a marked deviation from their reasonable performance when trained on 10-digits, as shown in Figure F.8(a). Figure G.1 shows that this phenomenon occurs across all random seeds. Conversely, RoPE and KerpleLog exhibit moderate in-distribution generalization but falter in out-of-distribution scenarios. Appendices G.1 and G.2 shows the training loss and test accuracy of these runs.

Analyzing errors in 11-digit additions from models trained on 10-digits revealed a common misalignment issue: the Transformer often adds operands adjacent to the correct ones. An attempt to rectify this by reformatting addition ($A_1B_1, A_2B_2, A_3B_3 = C_1C_2C_3$, with 1 as the least significant bit) failed to improve length generalization, merely shifting the error to adjacent output positions. This highlights the Transformer's inherent limitations in precise position identification.

Standard format vs reversed format. As shown in Figure 5, standard formatting shows limited length generalization in all PEs compared to the reversed format. FIRE excels in length generalization even with the standard format, even matching RoPE in reverse format. However, FIRE's performance (with standard format) declines beyond 60-digit additions, likely due to increased carry propagation challenges exceeding the model's capacity.

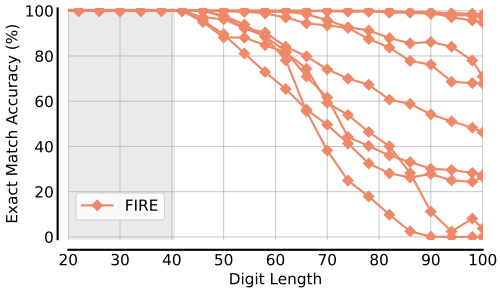


Figure 10: Exact match across 10 trials using FIRE. While transformers can achieve near-perfect accuracy in 100-digit addition, the variance across different random seeds is high.

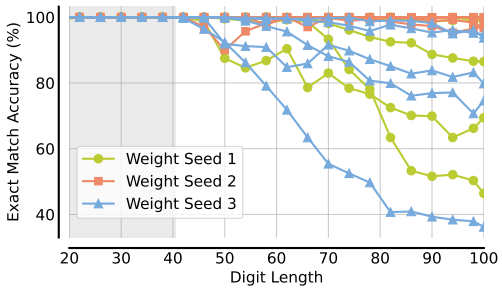


Figure 11: Effects of weight initialization and data input order: 15 models trained on a combination of three weight initialization seeds and five data input order seeds.

Looking at the training loss and training next-token accuracy in both formats also shows interesting differences. As shown in Figures 6 and F.3, the standard format training leads to gradual improvement, whereas reverse format yields a sharp performance transition. This transition, which is a reminiscent of "grokking" phenomenon Power et al. (2022), shows in this case the "Eureka moment" in which the Transformer learns the right addition algorithm.

Random space augmentation and randomized position encoding. Figure 7 reveals divergent impacts of random space augmentation on four PEs. The augmentation's efficacy is notably contingent upon the chosen PE. While Random Spaces marginally enhances RoPE and KerpleLog's performance, it markedly deteriorates NoPE and FIRE. A similar PE-specific pattern is evident in Randomized PE, as Figure 8 demonstrates. Randomized PE significantly degrades KerpleLog's effectiveness, yet it substantially boosts FIRE. See Appendices G.4 and G.5 for training loss and EM accuracy for all trials in each setting.

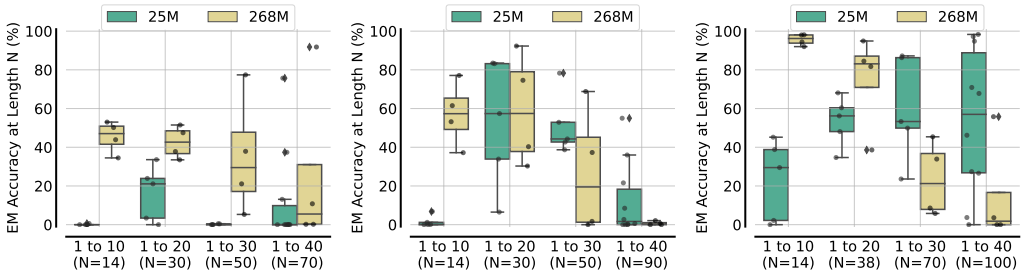
Length generalization is not robust to either weight initialization or training data order. Figure 10 illustrates the varying performance of 10 FIRE trials using identical training data order but distinct weight initializations. Notably, while all trials achieve similar close-to-zero training losses after 10K training steps (Figure F.2) and exhibit perfect in-distribution generalization, their out-of-distribution (OOD) length generalization shows significant variance. Moreover, the length generalization performance fluctuates significantly across training steps (Appendix F.3). This observation contrasts with earlier studies suggesting in-distribution loss as a reliable OOD generalization predictor (Nagarajan et al., 2020).

We further examine 15 unique combinations, resulting from 3 weight initialization seeds and 5 data input orders. As shown in Figure 11, there is significant variance across training data orders even when the weight initialization is constant. Intriguingly, certain weight initializations demonstrate remarkable resilience to changes in data input order. This observation is reminiscent of the Lottery Ticket Hypothesis (Frankle & Carbin, 2018), which posits the existence of a sparse, equally effective sub-network within a larger neural network. Our findings suggest the presence of "fortunate" weight configurations that exhibit robust length generalization, akin to a "lucky weight ticket."

While Anil et al. (2022) also noticed similar in-distribution accuracy but marked differences in OOD behavior on parity tasks, their OOD performance was quite poor across all runs. Moreover, contrary to the findings of Anil et al. (2022) on the impact of hyperparameter variation, our experiments reveal considerable performance fluctuations even with different random seeds. This inconsistency appears unrelated to position encoding (refer to Figure F.1 for different PEs), and is more likely due to variations in random weight initialization and data order.

3 Analysis

Error analysis. In examining Transformers' error characteristics, we classified erroneous predictions into two categories: those with and without carry. Figure 9 shows no significant difference between these categories, thus carry propagation does not majorly impede length generalization.



(a) RoPE (b) NoPE (c) FIRE

Figure 12: Scaling model size inconsistently affects length generalization performance. While consistently enhancing performance in shorter length regimes (1-10, 1-20) across four PEs, this trend does not hold for larger regimes (1-30, 1-40). For instance, larger models outperform smaller ones with RoPE and KerpleLog (Figure F.14), but underperform with NoPE and FIRE. Moreover, increasing model size doesn't noticeably decrease performance variance, suggesting size scaling isn't vital for length generalization.

Additionally, we analyzed the error distribution in 100-digit addition using FIRE, illustrated in Figure F.10. As shown, Figure F.10 indicates an overall uniform error distribution across all indices, despite some individual model checkpoints showing errors at specific positions. Excluding two near-zero accuracy runs, over 90% of errors in incorrect examples are single-digit mistakes, following an exponential distribution. Additional results are shown in Figures F.11 and F.12.

Despite the imperfect calculation, the FIRE model does not show any systematic error. Random errors may stem from phenomena such as attention glitches Liu et al. (2023a). Conversely, other PEs systematically fail to identify the start or end of addition, leading to premature termination.

Scaling model size. The scaling of model size is crucial for improving large language models (Thoppilan et al., 2022; Chowdhery et al., 2023). To assess its effect on length generalization, we contrasted models with 25M and 268M parameters. We find that model size variation has a minor effect on length generalization. Figure 12 shows that larger models slightly improve generalization in short digit regimes (1 to 10 and 1 to 20 digit additions) but yield mixed results in longer regimes. While RoPE and KerpleLog show improvements, NoPE and FIRE experience performance degradation with a larger model, indicating model size may not be the primary factor in length generalization.

The efficacy of length generalization in the 25M model prompted us to explore the capabilities of smaller models. Specifically, we trained models with 2M and 5M parameters. As Figures F.15 and F.16 illustrate, the 2M model's performance deteriorates with longer sequences, indicating limited model capacity as a potential performance bottleneck. Intriguingly, this model outperforms its larger counterparts (5M and 25M models) in tasks involving 1 to 10 digit addition. Furthermore, the 5M model remarkably achieves 80% accuracy in 100 digit addition, trained only on 1 to 40 digit tasks, surpassing the 268M model's performance.

4 Conclusion

Length generalization in Transformers has been a long-standing challenge. We evaluate the ability of Transformers to generalize to longer test sequences using the decimal addition task. Through extensive experiments, we find that there is no inherent limitation in Transformers' design preventing effective length generalization. Instead, the missing ingredient is the right combination of data format and position encoding. We show that Transformers can achieve almost perfect generalization on sequences up to $2.5\times$ the training length, given appropriate data formatting and position encoding.

Our thorough empirical analysis of common length generalization techniques reveals a significant dependency between the type of position encoding and the data format. This underscores the importance of synergizing data format with model architecture for optimal generalization. Despite these advancements, robust length generalization in Transformers remains elusive, even with meticulously finetuned regularization hyperparameters.

REFERENCES

Emmanuel Abbe, Samy Bengio, Aryo Lotfi, and Kevin Rizk. Generalization on the unseen, logic reasoning and degree curriculum. *arXiv preprint arXiv:2301.13105*, 2023.

Cem Anil, Yuhuai Wu, Anders Andreassen, Aitor Lewkowycz, Vedant Misra, Vinay Ramasesh, Ambrose Slone, Guy Gur-Ari, Ethan Dyer, and Behnam Neyshabur. Exploring length generalization in large language models. *Advances in Neural Information Processing Systems*, 35:38546–38556, 2022.

Pranjal Awasthi and Anupam Gupta. Improving length-generalization in transformers via task hinting. *arXiv preprint arXiv:2310.00726*, 2023.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

Shouyuan Chen, Sherman Wong, Liangjian Chen, and Yuandong Tian. Extending context window of large language models via positional interpolation. *arXiv preprint arXiv:2306.15595*, 2023.

Ta-Chung Chi, Ting-Han Fan, Peter J Ramadge, and Alexander Rudnicky. Kerple: Kernelized relative positional embedding for length extrapolation. *Advances in Neural Information Processing Systems*, 35:8386–8399, 2022.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240): 1–113, 2023.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc V Le, and Ruslan Salakhutdinov. Transformer-xl: Attentive language models beyond a fixed-length context. *arXiv preprint arXiv:1901.02860*, 2019.

Gregoire Deletang, Anian Ruoss, Jordi Grau-Moya, Tim Genewein, Li Kevin Wenliang, Elliot Catt, Chris Cundy, Marcus Hutter, Shane Legg, Joel Veness, and Pedro A Ortega. Neural networks and the chomsky hierarchy. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=WbxHAzkeQcn.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Shaoxiong Duan and Yining Shi. 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.

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

Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. *arXiv preprint arXiv:1803.03635*, 2018.

Gemini, Rohan Anil, Sebastian Borgeaud, Yonghui Wu, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, et al. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*, 2023.

Alex Graves, Greg Wayne, Malcolm Reynolds, Tim Harley, Ivo Danihelka, Agnieszka Grabska-Barwinska, Sergio Gomez Colmenarejo, Edward Grefenstette, Tiago Ramalho, John P. Agapiou, Adrià Puigdomènech Badia, Karl Moritz Hermann, Yori Zwols, Georg Ostrovski, Adam Cain, Helen King, Christopher Summerfield, Phil Blunsom, Koray Kavukcuoglu, and Demis Hassabis. Hybrid computing using a neural network with dynamic external memory. *Nat.*, 538 (7626):471–476, 2016. doi: 10.1038/NATURE20101. URL https://doi.org/10.1038/nature20101.

Adi Haviv, Ori Ram, Ofir Press, Peter Izsak, and Omer Levy. Transformer language models without positional encodings still learn positional information. *arXiv preprint arXiv:2203.16634*, 2022.

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.

Samy Jelassi, Stéphane d'Ascoli, Carles Domingo-Enrich, Yuhuai Wu, Yuanzhi Li, and François Charton. Length generalization in arithmetic transformers. *arXiv preprint arXiv:2306.15400*, 2023.

Amirhossein Kazemnejad, Inkit Padhi, Karthikeyan Natesan, Payel Das, and Siva Reddy. The impact of positional encoding on length generalization in transformers. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL `https://openreview.net/forum?id=Drrl2gcjzl`.

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

Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, et al. Solving quantitative reasoning problems with language models. *Advances in Neural Information Processing Systems*, 35:3843–3857, 2022.

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. In *The Twelfth International Conference on Learning Representations*, 2024. URL `https://openreview.net/forum?id=rR03qFesqk`.

Yujia Li, David Choi, Junyoung Chung, Nate Kushman, Julian Schrittwieser, Rémi Leblond, Tom Eccles, James Keeling, Felix Gimeno, Agustin Dal Lago, et al. Competition-level code generation with alphacode. *Science*, 378(6624):1092–1097, 2022.

Bingbin Liu, Jordan T Ash, Surbhi Goel, Akshay Krishnamurthy, and Cyril Zhang. Exposing attention glitches with flip-flop language modeling. *arXiv preprint arXiv:2306.00946*, 2023a.

Bingbin Liu, Jordan T. Ash, Surbhi Goel, Akshay Krishnamurthy, and Cyril Zhang. Transformers learn shortcuts to automata. In *The Eleventh International Conference on Learning Representations*, 2023b. URL `https://openreview.net/forum?id=De4FYqjFueZ`.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

Mohammad Motamedi, Nikolay Sakharnykh, and Tim Kaldewey. A data-centric approach for training deep neural networks with less data. *arXiv preprint arXiv:2110.03613*, 2021.

Vaishnavh Nagarajan, Anders Andreassen, and Behnam Neyshabur. Understanding the failure modes of out-of-distribution generalization. *arXiv preprint arXiv:2010.15775*, 2020.

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

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

Alethea Power, Yuri Burda, Harri Edwards, Igor Babuschkin, and Vedant Misra. Grokking: Generalization beyond overfitting on small algorithmic datasets. *arXiv preprint arXiv:2201.02177*, 2022.

Ofir Press, Noah Smith, and Mike Lewis. Train short, test long: Attention with linear biases enables input length extrapolation. In *International Conference on Learning Representations*, 2022. URL `https://openreview.net/forum?id=R8sQPpGCv0`.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research*, 21(1):5485–5551, 2020.

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.

Avi Schwarzschild, Eitan Borgnia, Arjun Gupta, Furong Huang, Uzi Vishkin, Micah Goldblum, and Tom Goldstein. Can you learn an algorithm? generalizing from easy to hard problems with recurrent networks. *Advances in Neural Information Processing Systems*, 34:6695–6706, 2021.

Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. Self-attention with relative position representations. *arXiv preprint arXiv:1803.02155*, 2018.

Noam Shazeer. Glu variants improve transformer. *arXiv preprint arXiv:2002.05202*, 2020.

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.

David R So, Wojciech Mańke, Hanxiao Liu, Zihang Dai, Noam Shazeer, and Quoc V Le. Primer: Searching for efficient transformers for language modeling. *arXiv preprint arXiv:2109.08668*, 2021.

Jianlin Su. Rectified rotary position embeddings. `https://github.com/bojone/rerope`, 2023.

Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024.

Romal Thoppilan, Daniel De Freitas, Jamie Hall, Noam Shazeer, Apoorv Kulshreshtha, Heng-Tze Cheng, Alicia Jin, Taylor Bos, Leslie Baker, Yu Du, et al. Lamda: Language models for dialog applications. *arXiv preprint arXiv:2201.08239*, 2022.

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

Yuhuai Wu, Albert Qiaochu Jiang, Wenda Li, Markus Rabe, Charles Staats, Mateja Jamnik, and Christian Szegedy. Autoformalization with large language models. *Advances in Neural Information Processing Systems*, 35:32353–32368, 2022.

Yi Zhang, Arturs Backurs, Sébastien Bubeck, Ronen Eldan, Suriya Gunasekar, and Tal Wagner. Unveiling transformers with lego: a synthetic reasoning task. *arXiv preprint arXiv:2206.04301*, 2022.

Hattie Zhou, Azade Nova, Hugo Larochelle, Aaron Courville, Behnam Neyshabur, and Hanie Sedghi. Teaching algorithmic reasoning via in-context learning. *arXiv preprint arXiv:2211.09066*, 2022.

Hattie Zhou, Arwen Bradley, Etai Littwin, Noam Razin, Omid Saremi, Joshua M. Susskind, Samy Bengio, and Preetum Nakkiran. What algorithms can transformers learn? a study in length generalization. In *The Twelfth International Conference on Learning Representations*, 2024. URL `https://openreview.net/forum?id=AssIuHnmHX`.

A POSITION ENCODING AND DATA FORMATS

Recently proposed improvements in architectural design, notably in position encoding (Shen et al., 2023; Kazemnejad et al., 2023; Ruoss et al., 2023) and attention mechanisms (Dubois et al., 2019; Duan & Shi, 2023), aim to address the challenge of length generalization in arithmetic computations with Transformers. However, the effectiveness of such modifications is often constrained, either due to their overly ad-hoc nature or their poor performance on longer sequences. Although scaling the size of models and datasets has been recognized as a generally effective strategy to improve performance, prior research (Brown et al., 2020; Anil et al., 2022) suggests that relying solely on scale might not be sufficient for handling test sequences that are longer than training. Concurrently, with the rising focus on data-centric AI (Motamedi et al., 2021), recent work has investigated refining the data format to enhance the learning efficacy of existing Transformer models. In this section, we review some of the most common position encodings (Appendix A.1) and relevant data formats (Appendix A.2)

A.1 POSITION ENCODING FOR LENGTH GENERALIZATION

The inability of transformers to extrapolate to longer sequences has been primarily attributed to position encoding (PE; Shaw et al., 2018). In this section, we review existing positional encoding approaches with an emphasis on their length generalization abilities.

Absolute Positional Encoding (APE). APE enhances Transformer models with positional information by attaching a positional vector $\boldsymbol{p}_i$ to each position i. This is achieved through a predefined sinusoidal function (Vaswani et al., 2017) or a learnable approach (Devlin et al., 2018). Then, the vector $\boldsymbol{p}_i$ is combined with the token embedding $\boldsymbol{e}_i$ before entering the transformer's first layer. Although straightforward, APE often struggles with generalizing to longer sequences, as observed in both NLP (Press et al., 2022) and algorithmic tasks (Kazemnejad et al., 2023).

Additive Relative Positional Encoding (RPE). Shaw et al. (2018) pioneered the additive RPEs, diverging from standard input-level integration by modifying keys and, optionally, values in each attention layer. This concept was advanced by T5, which employed scalar biases to directly affect pre-softmax attention logits, a method noted for its simplicity yet criticized for limited efficiency and positional differentiation in long sequences (Raffel et al., 2020; Press et al., 2022). Later approaches such as Alibi (Press et al., 2022), Kerple (Chi et al., 2022) and FIRE (Li et al., 2024) build on the idea of learned additive bias, proposing different functions to model the scalar bias as a function of the key- and query-indices. Most pre-softmax attention logits of additive RPEs can be generally written as (Li et al., 2024):

$$\boldsymbol{A}_{\mathrm{RPE}}(\boldsymbol{X}) = \boldsymbol{X}\boldsymbol{W}_Q(\boldsymbol{X}\boldsymbol{W}_K)^\top + \boldsymbol{B}, \tag{A.1}$$

where $\boldsymbol{X}$, $\boldsymbol{W}_Q$, $\boldsymbol{W}_K$ denote the input and weight matrices for queries and keys. The bias matrix $\boldsymbol{B} \in \mathbb{R}^{n\times n}$ is induced by the **position encoding function** $b : \mathbb{N}^{*2} \to \mathbb{R}$, with its (i, j)-th entry defined as $b(i, j)$. Instances of $b(i, j)$ include:

- T5 (Raffel et al., 2020): $b(i, j) = r_{min}\{i-j, K\}$, where K is a hyperparameter and r_i are learned scalars.
- Alibi (Press et al., 2022): $b(i, j) = -r\,|i - j|$, where $r > 0$ is a hyperparameter.
- KerpleLog (Chi et al., 2022): $b(i, j) = -r_1 \log(1 + r_2|i - j|)$, where $r_1, r_2 > 0$ are learnable scalars.
- FIRE (Li et al., 2024): $b(i, j) = f_\theta\left(\frac{\psi(i-j)}{\psi(\max\{L,i\})}\right)$, where $f_\theta : \mathbb{R} \to \mathbb{R}$ is a learnable MLP parameterized by θ, $\psi : \mathbb{N} \to \mathbb{R}_+$ is $\psi(x) = log\,(cx + 1)$ and $c > 0, L > 0$ are learnable scalars.

Additional background on additive RPEs is provided in Appendix D.1

Rotary Positional Encoding (RoPE). RoPE (Su et al., 2024) encodes position information in attention logits through rotational encoding of query and key vectors based on their relative positions. Despite being simple and effective, RoPE exhibits limited length generalization (Press et al., 2022; Kazemnejad et al., 2023). While extensions like Position Interpolation Chen et al. (2023); Peng et al. (2023); Su (2023) enhance RoPE's context length, they do not necessarily improve length generalization on algorithmic tasks where learning the underlying algorithm is crucial.

No Positional Encoding (NoPE). While encoder-only Transformers (e.g., BERT (Devlin et al., 2018)) are permutation equivariant without positional encodings, decoder-only counterparts with causal attention, as shown by Haviv et al. (2022), acquire positional understanding autonomously, even without explicit PE. Interestingly, recent findings by Kazemnejad et al. (2023) further reveal that a model without PE outperforms those with specialized PEs on simple algorithmic tasks.

Randomized Position Encoding. Ruoss et al. (2023) introduced Randomized PE to enhance existing PEs by randomly sampling encodings from a range exceeding test-time lengths while preserving the order. Transformers trained this way adapt to larger positional encodings, effectively eliminating OOD position encodings during testing.

A.2 Data Formats

Data format plays a pivotal role in enhancing Transformers' length generalization capabilities, primarily by transforming the data into a format that could be more easily learned. We give an overview of the existing techniques below.

Reversed Format. Computing addition in an algorithmic way (as taught in elementary school) requires starting with the least significant digit (LSD) and proceeds to the most significant digit (MSD). This sequence contrasts with the standard printed format ($A_3A_2A_1 + B_3B_2B_1 = C_3C_2C_1$, where A_1 and B_1 are the LSDs, which is not ideally suited for autoregressive models due to their outputting the MSD first. However, the reversed format ($A_1A_2A_3 + B_1B_2B_3 = C_1C_2C_3$) aligns better with these the natural order of computing the digits. It simplifies the learning task to a function that depends only on the two corresponding operand digits and the carry from the previous step (Lee et al., 2023; Zhou et al., 2024; Shen et al., 2023).

Index Hints. Zhou et al. (2024) introduced "index hints" in both the query and response of arithmetic tasks. For example, $42 + 39 = 81$ is represented as $a4b2 + a3b9 = a8b1$ during training and inference, enabling transformers to execute indexing via induction heads (Olsson et al., 2022).

Random Space Augmentation. Shen et al. (2023) explored the impact of random spacing between digits in addition, aiming to disrupt the model's reliance on absolute positional information. Their results show successful generalization from 10-digit to 11-digit addition, but falters with longer sequences.

Figure 2 lists the position encodings and data formats used in some of the most related work to ours.

B A Recipe for Length Generalization in Decimal Addition

The task of decimal addition is composed of two critical subtasks: (a) the identification of the right operands to add; and (b) the summation of these operands with the preceding carry. While the summation step ((b)) is relatively easier because it has a finite set of possible inputs, the primary generalization challenge lies in the operand identification ((a)), where precise positional access is crucial.

Our best model, which leads to the results in Figure 1, uses the following combination:

1. **FIRE position encodings** (Li et al., 2024): We believe that FIRE position encodings are helpful for length generalization because they are more expressive than other PEs, as shown by Li et al. (2024).
2. **Randomized position encodings** (Ruoss et al., 2023): We believe that randomized position encodings are crucial to avoid overfitting on the position indices and index differences that were seen during training.
3. **Reversed format**: The reversed format makes it easier for the model to *decompose* the long computation to local, "markovian", steps that depend only on the single previous step.
4. **Index hints** (Zhou et al., 2024): We believe that index hints are useful because they ease the task of *operand identification* (discussed in (a)), of matching the right operands to add at a certain step.

We ablate each of these decisions and some other alternative choices in Section 2.

C RELATED WORK

Length generalization remains a significant challenge in neural networks, underscored by substantial research (Graves et al., 2016; Hupkes et al., 2020; Schwarzschild et al., 2021; Zhang et al., 2022; Deletang et al., 2023; Dziri et al., 2023). Despite their advanced reasoning capabilities, Transformer-based large language models (LLMs) (Thoppilan et al., 2022; Chowdhery et al., 2023) struggle with processing sequences beyond their training scope Anil et al. (2022). Enhancements in length generalization, especially in the addition task, primarily focus on two areas: refining positional encoding and optimizing data format.

Position Encoding for Length Generalization The inability of Transformers to extrapolate to longer sequences has been primarily attributed to Position Encoding (PE) Shaw et al. (2018). Various studies have suggested alternatives, such as relative positional encodings, which focus on the relative distances between tokens (Dai et al., 2019), the implementation of randomized position encoding (Ruoss et al., 2023), or the adoption of weighted attention mechanisms in place of position embeddings (Press et al., 2022; Raffel et al., 2020; Chi et al., 2022; Li et al., 2024). These approaches have shown promise in natural language processing (NLP). However, Kazemnejad et al. (2023) found that omitting position encoding entirely yields better results for algorithmic tasks. In contrast, our experiments indicate that an effectively designed PE, such as the FIRE, is crucial for achieving optimal length generalization (Figure 3). Moreover, we show that a synergistic approach to consider both PE and data design markedly enhances length generalization capabilities.

Data format for Length Generalization A range of heuristic-based data formatting methods have been introduced, particularly for pretrained LLMs. These methods, including the use of scratchpads and the chain of thoughts approach, aim to facilitate arithmetic learning either through in-context learning or fine-tuning Anil et al. (2022); Zhou et al. (2022). Conversely, there is a body of research focused on Transformers trained from scratch. This research indicates that employing techniques such as reversed formatting and scratch pads can significantly boost length generalization performance Shen et al. (2023); Lee et al. (2023). Furthermore, it has been observed that both the data distribution and the sampling strategies can profoundly influence generalization Lee et al. (2023). Awasthi & Gupta (2023) further demonstrates the benefits of incorporating a simpler auxiliary task (e.g., identifying the successor element) in supporting the primary task (e.g., sorting). In contrast, Jelassi et al. (2023) finds that train set priming enables length generalization for a encoder-only Transformer model. In contrast, our good length generalization performance achieved with naive random sampling approach suggesting that sophisticated data sampling might be redundant.

D POSITIONAL ENCODING

D.1 ADDITIVE RELATIVE POSITIONAL ENCODING (RPE)

Shaw et al. (2018) pioneered additive RPE by integrating position encodings into the attention layer's key, and optionally the value, rather than the input. This concept was further simplified in T5 (Raffel et al., 2020), where the vector representations of relative positions are simplified to scalar biases added to pre-softmax attention logits. Subsequent advancements in additive RPE, aimed at enhancing length generalization and computational efficiency, include notable methods like Alibi (Press et al., 2022), Kerple (Chi et al., 2022), and FIRE (Li et al., 2024). A commonality among these methods is the unified computation formula for pre-softmax attention logits, as outlined by Li et al. (2024):

$$\boldsymbol{A}_{\mathrm{RPE}}(\boldsymbol{X}) = \boldsymbol{X}\boldsymbol{W}_Q(\boldsymbol{X}\boldsymbol{W}_K)^\top + \boldsymbol{B}, \tag{D.1}$$

where the bias matrix $\boldsymbol{B} \in \mathbb{R}^{n\times n}$ is induced by the **position encoding function** $b : \mathbb{N}^{*2} \to \mathbb{R}$, has its (i, j)-th entry defined as $b(i, j)$. Variations in b's formulations and parameterizations give rise to diverse RPE variants.

- T5 (Raffel et al., 2020): T5's RPE segments relative distances into distinct buckets with a logarithmic scale, each associated with a unique parameter. With $K + 1$ buckets and a

pre-defined distance L_1, the attention bias is calculated as (assuming $K+1$ is even)

$$b(i,j) = \begin{cases} r_{i-j} & 0 \le i-j < \frac{K+1}{2} \\ r_{\frac{K+1}{2}+\lfloor \frac{K+1}{2}\log\left(\frac{2(i-j)}{K+1}\right)/\log\left(\frac{2L_1}{K+1}\right)\rfloor} & \frac{K+1}{2} \le i-j < L_1 \\ r_K & i-j \ge L_1 \end{cases}. \tag{D.2}$$

- Alibi (Press et al., 2022): $b(i,j) = -r|i-j|$, where $r > 0$ is a hyper-parameter.
- Kerple (Chi et al., 2022): $b(i,j) = -r_1 \log(1 + r_2|i-j|)$ (logarithmic variant) or $-r_1|i-j|^{r_2}$ (power variant), where $r_1, r_2 > 0$ are learnable scalars.
- FIRE (Li et al., 2024): $b(i,j) = f_\theta\left(\frac{\psi(i-j)}{\psi(\max\{L,i\})}\right)$, where $f_\theta : \mathbb{R} \to \mathbb{R}$ is a learnable MLP parameterized by θ, $\psi : \mathbb{N} \to \mathbb{R}_+$ is monotonically increasing and $L > 0$ is a learnable scalar.

E IMPLEMENTATION DETAILS

E.1 DATA GENERATION

As shown in Figure 2, we adopt the reversed format with index hints as our default data format. During training, we randomly sample a consecutive index hints from a pre-defined ordered index set with 102 distinct symbols, thereby enhancing the learning of hint sequences and their order. At inference, the same hint sampling strategy is applied to questions, prompting the model for answers.

To generate addition examples, we opt for a naive random sampling approach instead of structured data sampling Lee et al. (2023), as our analysis indicates that carry operations are not a major hindrance to length generalization (See Figure 9). Our approach involves uniformly selecting the number's length from 1 to the maximum training length, followed by independent sampling of two operands based on this length, with an additional zero padding to accommodate potential carry-induced extra digits. For training, datasets comprising 30M, 40M, 60M, and 120M examples are generated for number lengths 1-40, 1-30, 1-20, and 1-10, respectively. In contrast, the test set consists of 1,000 examples per digit length.

E.2 TRAINING DETAILS

Our base model, following Zhou et al. (2024), is a 25M parameter Transformer featuring 6 blocks, a 512 hidden size, a feedforward layer with a hidden dimension of 2048 using GeGLU activation (Shazeer, 2020), and an 8-head attention mechanism. We also adopt RMSNorm, integrating both PreNorm and PostNorm layers, following the Primer architecture (So et al., 2021). Additionally, our preliminary investigations underscore the significance of employing causal language modeling when applying the index hint technique. Conversely, attempts to leverage prefix language modeling paired with bidirectional attention in model inputs consistently falter in length generalization. Our three other model variants with size [2M, 5M, 268M] consist of [2, 4, 16] blocks, a [256, 256, 1024] hidden size, a feedforward layer with a hidden dimension of [1024, 1024, 4096], and a [4, 4, 16]-head attention mechanism, respectively.

In our implementation of FIRE Li et al. (2024), we employ layerwise sharing of attention bias across all attention blocks to enhance training efficiency. The paraterization of FIRE consists of a 2-layer MLP with a 32-unit hidden layer, utilizing ReLU activation.

We use the AdamW optimizer (Loshchilov & Hutter, 2017) to train the model with a weight decay value of 0.1 and dropout rate of 0.0. The learning rate schedule incorporates an initial 500-step linear warm-up, followed by a cosine decay, starting at 3e-4. We train the model with sequence packing, a batch size of 128, and a sequence length of 2048, over 50,000 steps. We use greedy decoding to generate the model output during evaluation. We summarize the hyperparameters in Table E.1.

Table E.1: Hyperparameters Summary for Length Generalization

Hyperparameter	**Value**
Language Model Type	Causal
Activation Functions	GeGLU
Normalization Layer	RMSNorm
Normalization Type	PreNorm and PostNorm
Optimizer	AdamW
Training Steps	50,000
Batch size	128
Weight Decay	0.1
Dropout	0.0
Learning Rate (LR)	0.0003
LR Warmup Steps	500
LR Cooldown (Begin, End)	(500, 50,000)
Warmup Schedule	Linear (from 0 to LR)
Cooldown Schedule	Cosine Decay (from LR to 0.1LR)
Training Sequence Length	2048
Evaluation	Greedy

F Additional Results

F.1 Training Loss and Sequence Exact Match Accuracy of Reverse Format with Index Hint trained up to 40

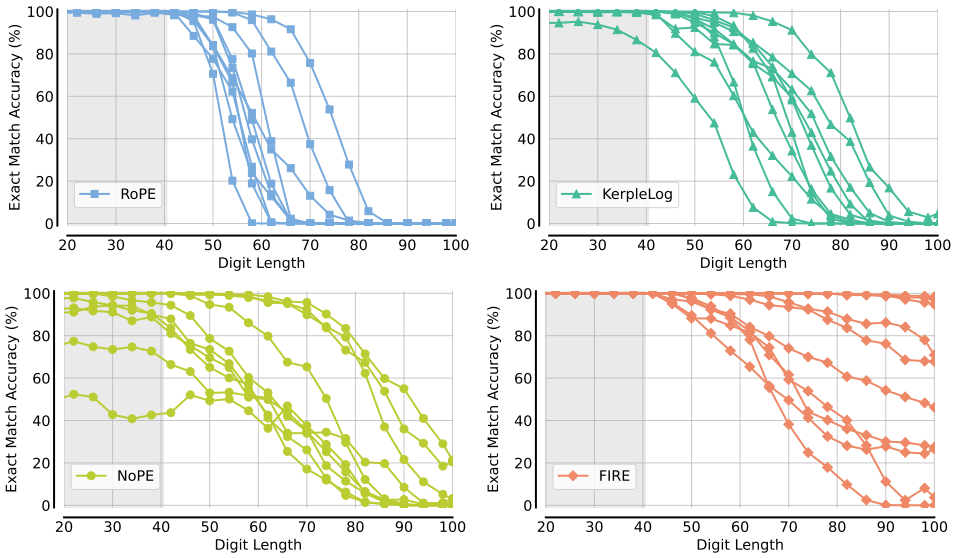


Figure F.1: Exact match accuracy on 20 to 100 digit addition of all 10 trials trained on up to 40-digit addition with index hint and reverse format using four different position encodings.

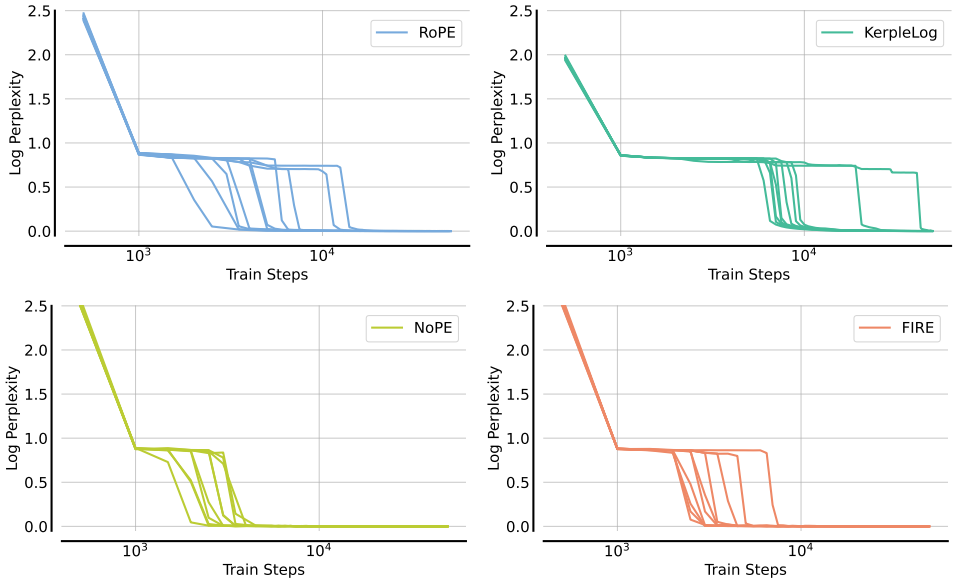
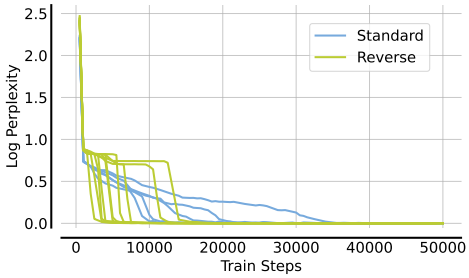
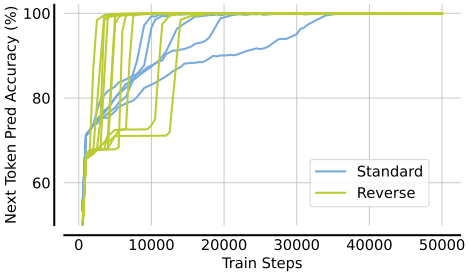


Figure F.2: Training loss over 10 trials in reverse formats. Despite similar nearly 0 log perplexity losses across runs after 10K training steps, different runs exhibit very different length generalization.

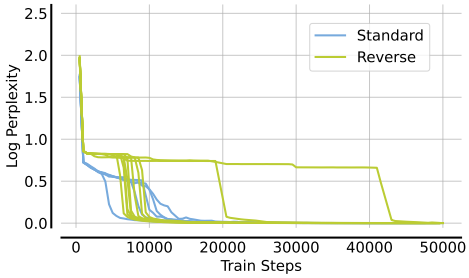
F.2 Training Loss and Next-token Prediction Accuracy of Standard and Reverse Format



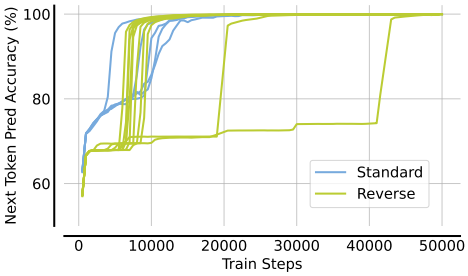
(a) Training Loss using RoPE



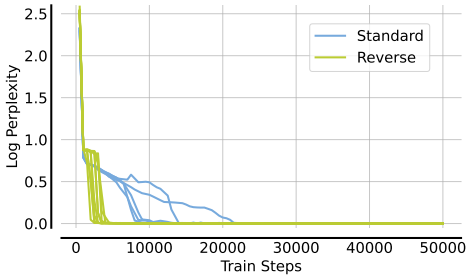
(b) Next-token Prediction Accuracy using RoPE



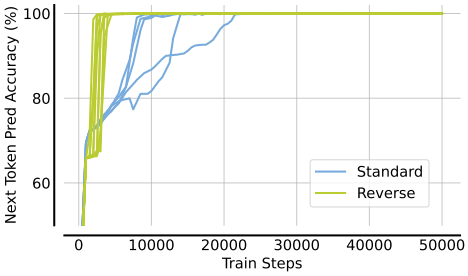
(c) Training Loss using KerpleLog



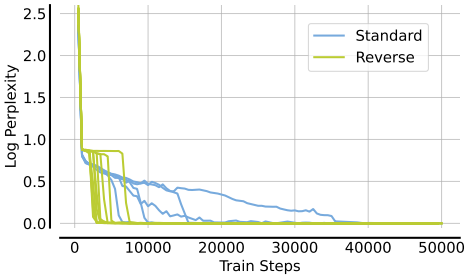
(d) Next-token Prediction Accuracy using KerpleLog



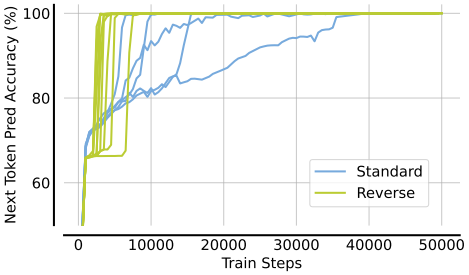
(e) Training Loss using NoPE



(f) Next-token Prediction Accuracy using NoPE



(g) Training Loss using FIRE



(h) Next-token Prediction Accuracy using FIRE

Figure F.3: Training log perplexity and next-token prediction accuracy over 10 trials in standard versus reverse formats using RoPE, KerpleLog, NoPE and FIRE. Reverse format shows distinct grokking during training, unlike the gradual enhancement in standard format.

F.3 The evolution of EM Accuracy during training in reverse format using 4 PEs

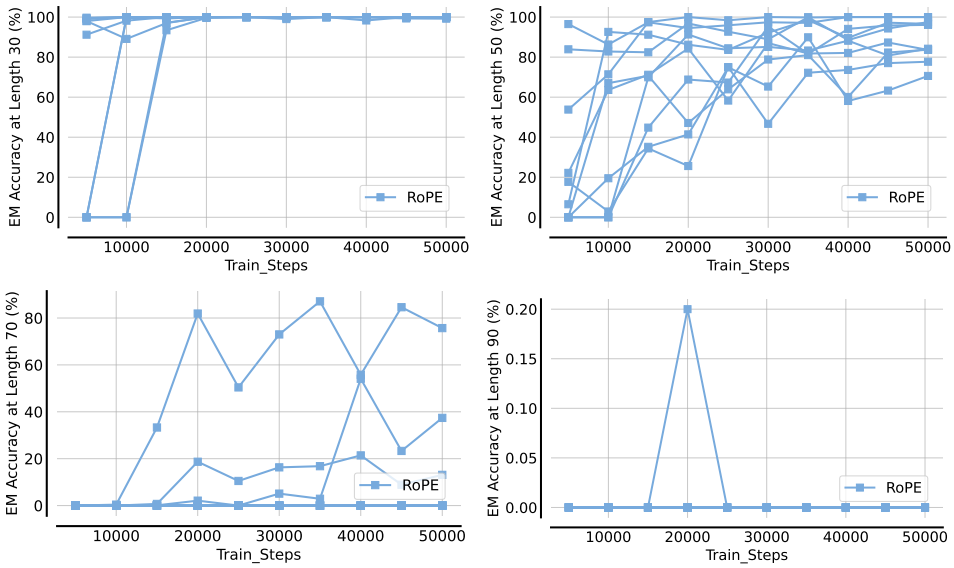


Figure F.4: Exact match accuracy on $[30, 50, 70, 90]$ digit addition of all 10 trials trained on up to 40-digit addition with index hint and reverse format using RoPE.

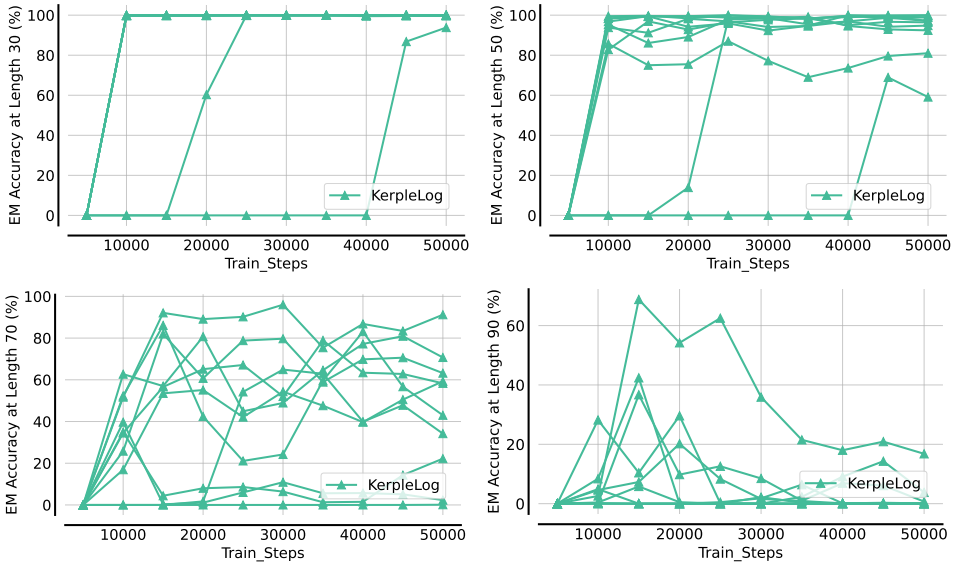


Figure F.5: Exact match accuracy on $[30, 50, 70, 90]$ digit addition of all 10 trials trained on up to 40-digit addition with index hint and reverse format using KerpleLog.

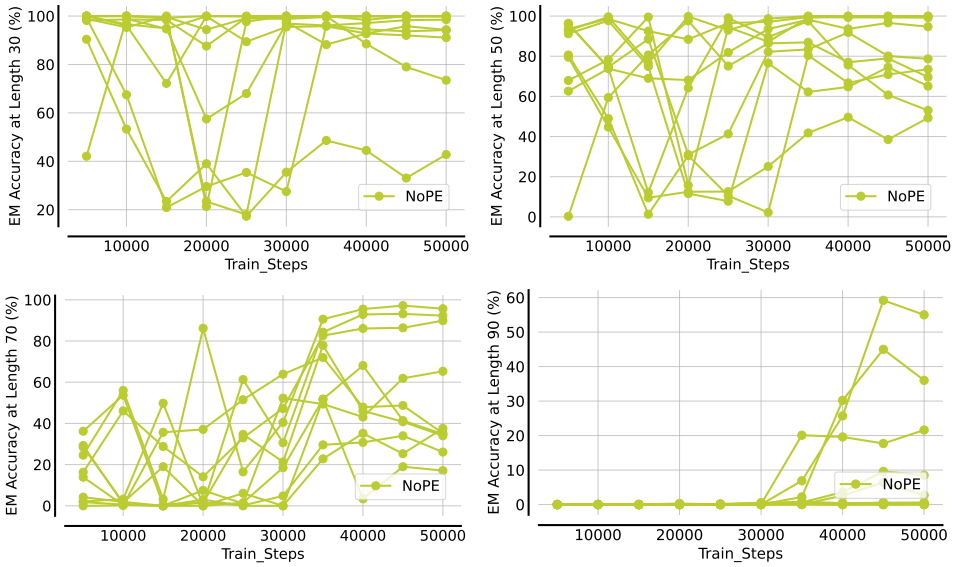


Figure F.6: Exact match accuracy on $[30, 50, 70, 90]$ digit addition of all 10 trials trained on up to 40-digit addition with index hint and reverse format using NoPE.

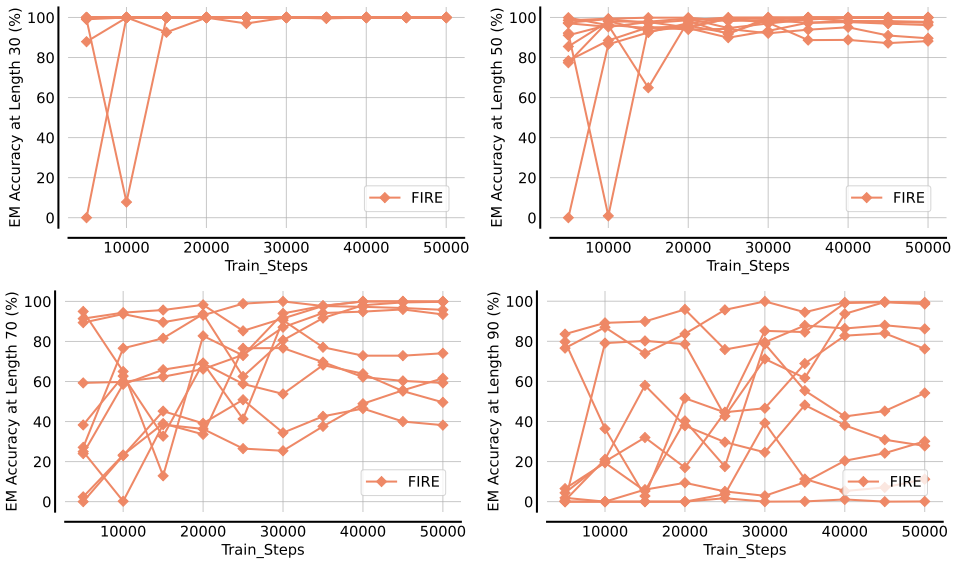
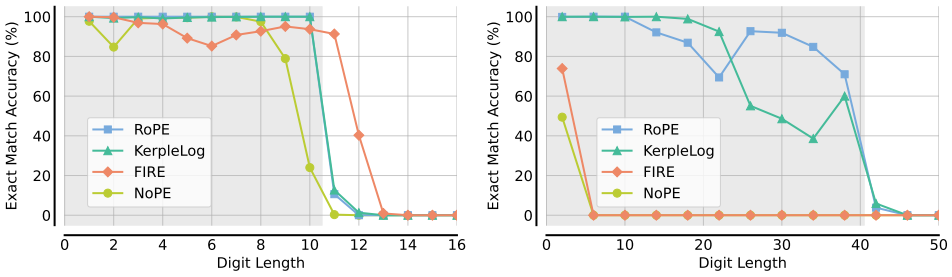


Figure F.7: Exact match accuracy on $[30, 50, 70, 90]$ digit addition of all 10 trials trained on up to 40-digit addition with index hint and reverse format using FIRE.

F.4 Effect of Index Hint



(a) Models trained up to 10-digit addition (b) Models trained up to 40-digit addition

Figure F.8: Best sequence exact match accuracy over five trials without index hint, trained upto length 10. All position encoding methods fail to generalize beyond trivial lengths and struggle with in-distribution generalization, highlighting the crucial role of index hints in length generalization. See the performance of each run in Appendix G.2 and Appendix G.1 for trained up to 10-digit and 40-digit addition.

F.5 Source of Variance

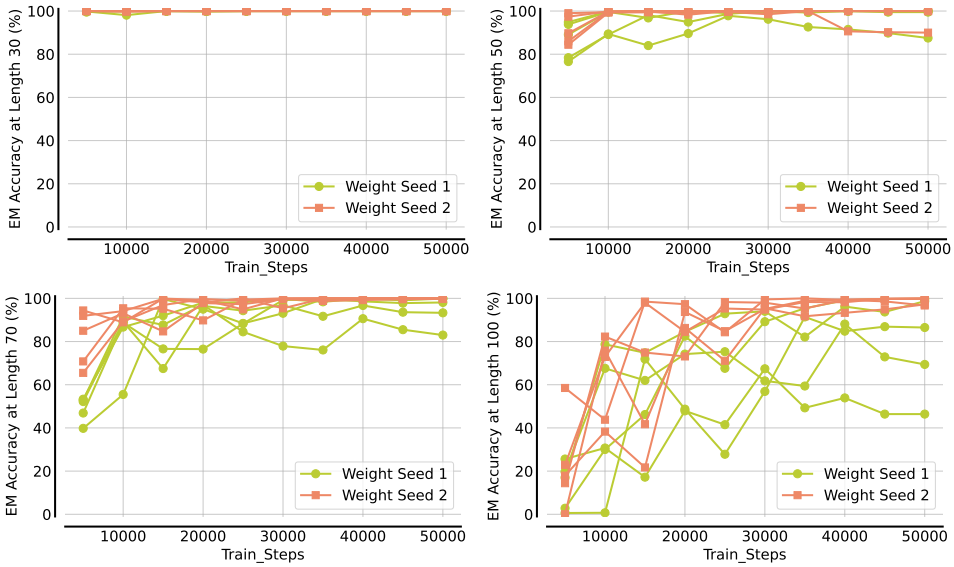


Figure F.9: Exact match accuracy on $[30, 50, 70, 100]$ digit addition of all 10 trials (2 weight seeds x 5 data seeds) trained on up to 40-digit addition with index hint and reverse format using FIRE.

F.6 Error Analysis

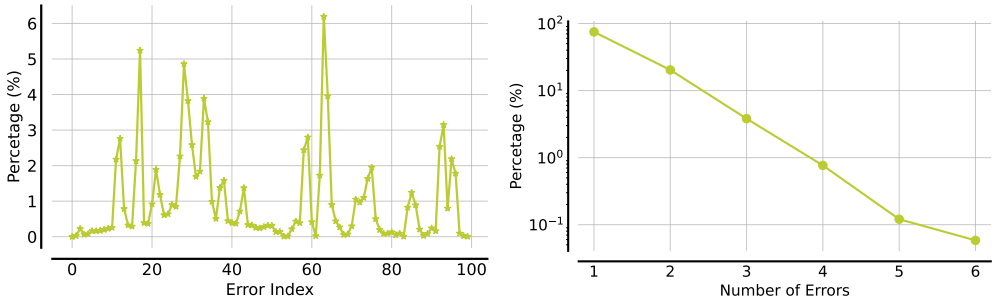


Figure F.10: (Left) Average error position distribution over 10 runs, showing a broad error spread across all positions. Specific checkpoints exhibit a propensity for errors at certain positions (refer to Figure F.12). (Right) Notably, in successful generalizations, more than 90% of errors are confined to single-digit inaccuracies, exhibiting an exponential distribution.

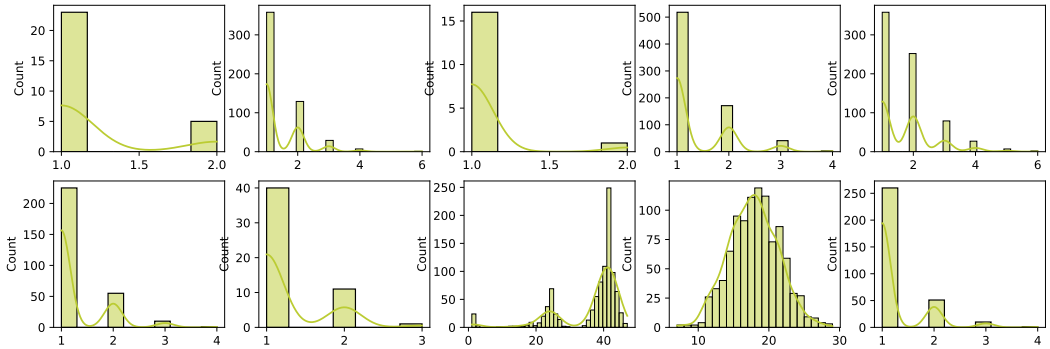


Figure F.11: Error count distribution

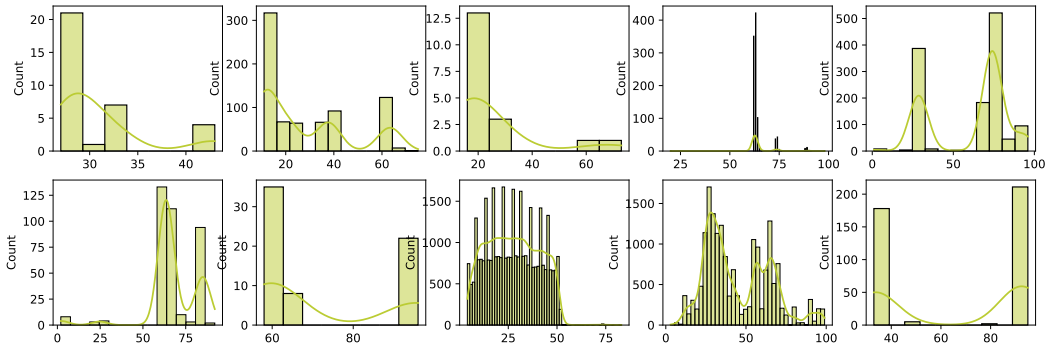
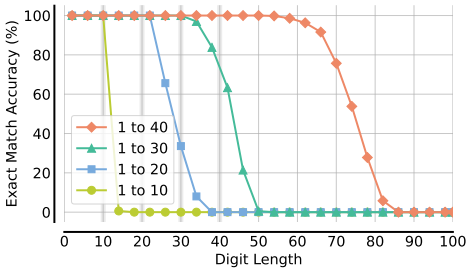
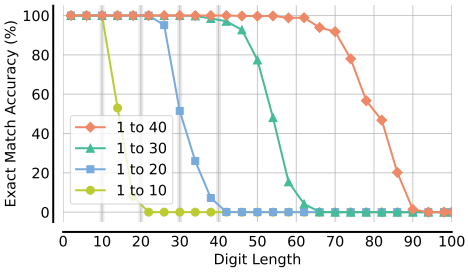


Figure F.12: Error position distribution (FIRE)

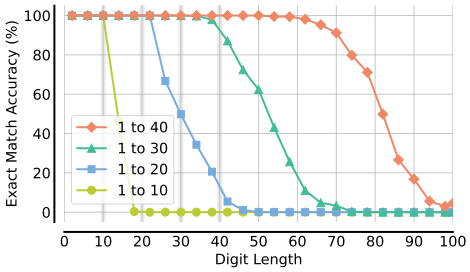
F.7 Training Digit Length Effect



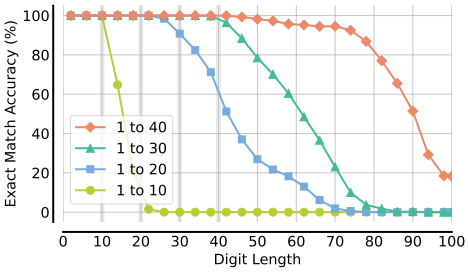
(a) EM accuracy of a 25M model using RoPE



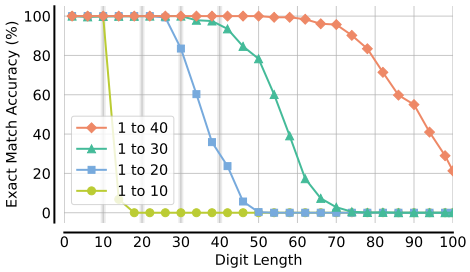
(b) EM accuracy of a 268M model using RoPE



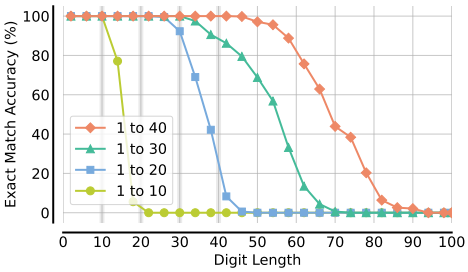
(c) EM accuracy of a 25M model using KerpleLog



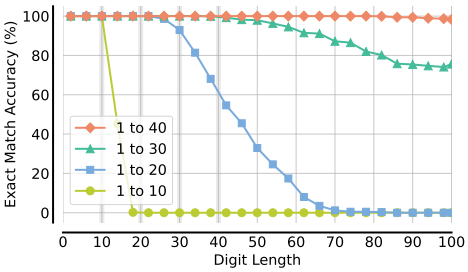
(d) EM accuracy of a 268M model using KerpleLog



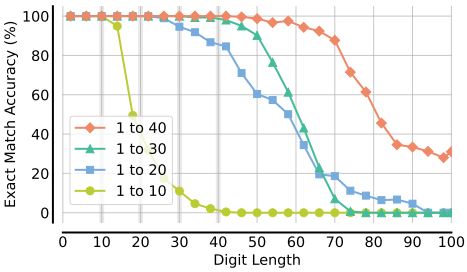
(e) EM accuracy of a 25M model using NoPE



(f) EM accuracy of a 268M model using NoPE



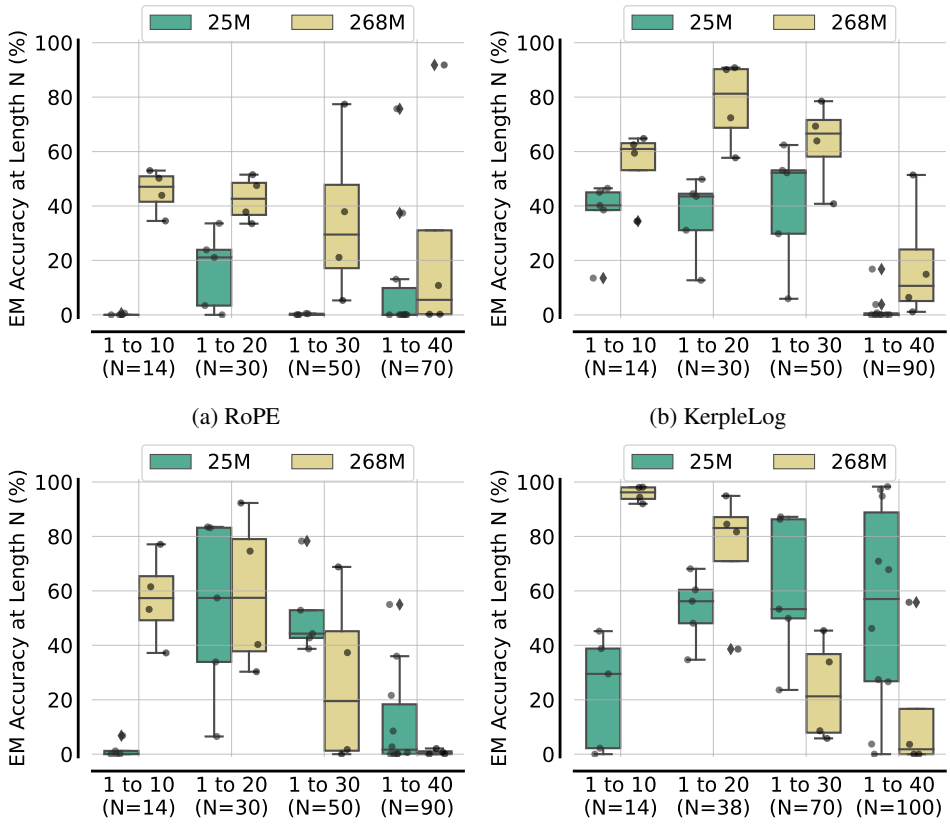
(g) EM accuracy of a 25M model using FIRE



(h) EM accuracy of a 268M model using FIRE

Figure F.13: Best sequence exact match accuracy of 5 trials with two model sizes (i.e., 25M and 268M), trained on up to 10, 20, 30 and 40 digit length using 4 PEs.

F.8 Model Size Effect



(a) RoPE (b) KerpleLog

(c) NoPE (d) FIRE

Figure F.14: Scaling model size inconsistently affects length generalization performance. While consistently enhancing performance in shorter length regimes (1-10, 1-20) across four position encodings, this trend does not hold for larger regimes (1-30, 1-40). For instance, larger models outperform smaller ones with RoPE and KerpleLog encodings, but underperform with NoPE and FIRE. Moreover, increasing model size doesn't noticeably decrease performance variance, suggesting size scaling isn't vital for length generalization.

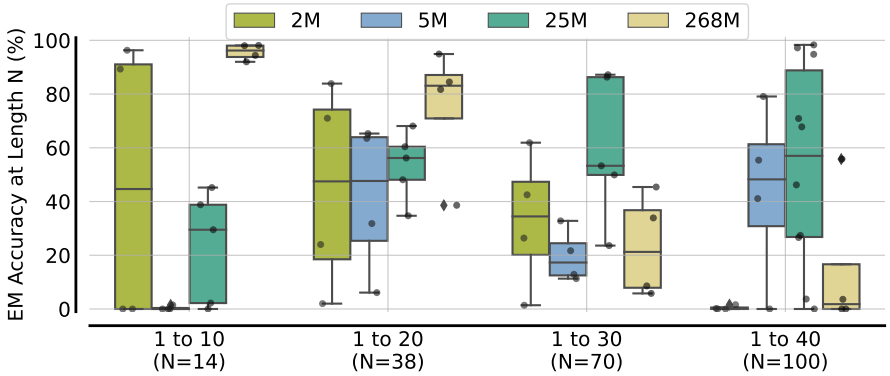
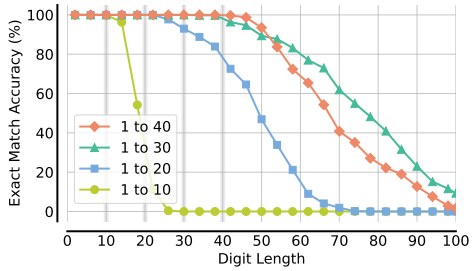
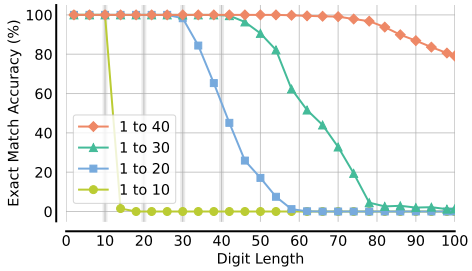


Figure F.15: Effect of different model sizes with FIRE as the position encoding.

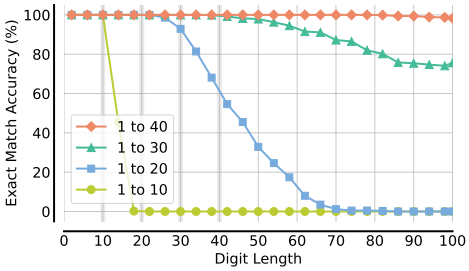
F.9 FIRE RELATED SCALING



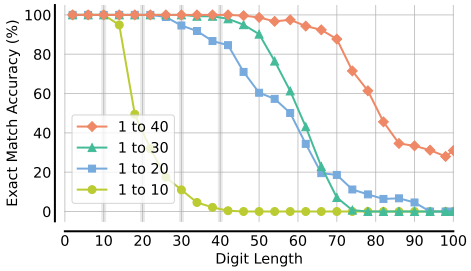
(a) EM accuracy of a 2M model using FIRE



(b) EM accuracy of a 5M model using FIRE



(c) EM accuracy of a 25M model using FIRE



(d) EM accuracy of a 268M model using FIRE

Figure F.16: Best sequence exact match accuracy of 5 trials with four model sizes (i.e., 2M, 5M, 25M and 268M), trained on up to 10, 20, 30 and 40 digit length using **FIRE**.

F.10 HYPERPARAMETER STUDY

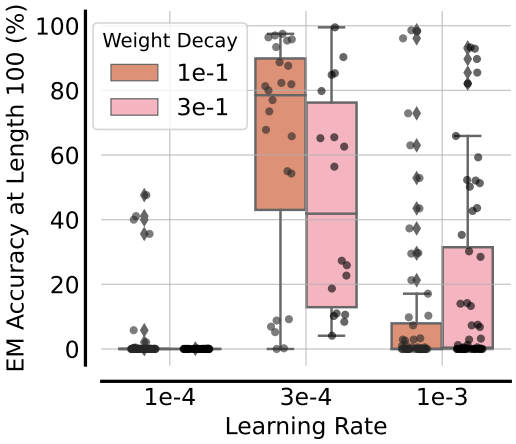


Figure F.17: Sequence exact match accuracy for test digit length 100, trained on digit lengths 1-40. 3e-4 seems to be the optimal learning rate.

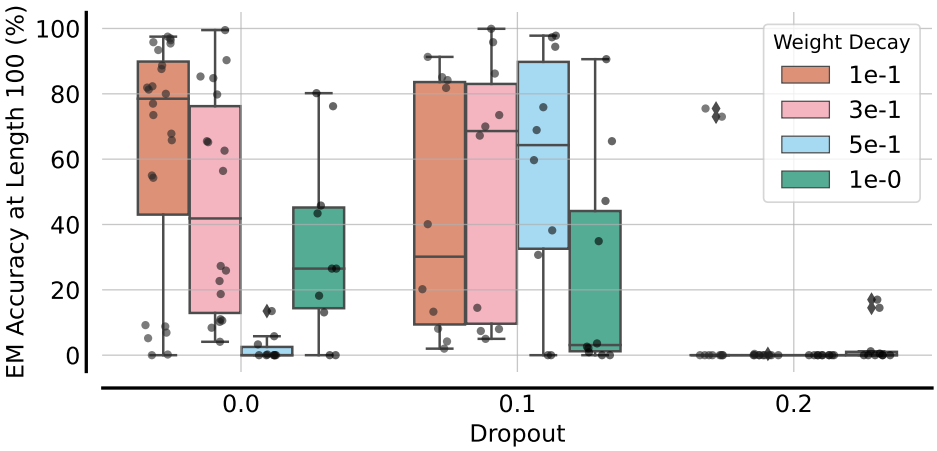


Figure F.18: Sequence exact match accuracy for test digit length 100, trained on digit lengths 1-40. A higher dropout rate markedly impedes length generalization, whereas a lower rate shows negligible impact.

G Training Loss and Sequence Exact Match Accuracy

G.1 Reverse Format without Index Hint trained up to 40-digit addition

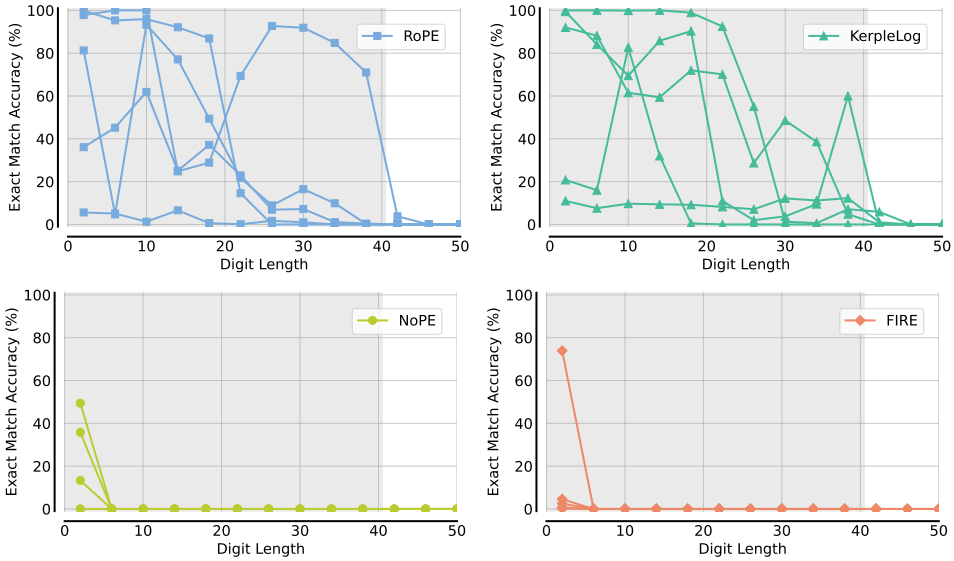


Figure G.1: Exact match accuracy on 20 to 100 digit addition of all 10 trials trained on up to 40-digit addition with index hint and reverse format using four different position encodings.

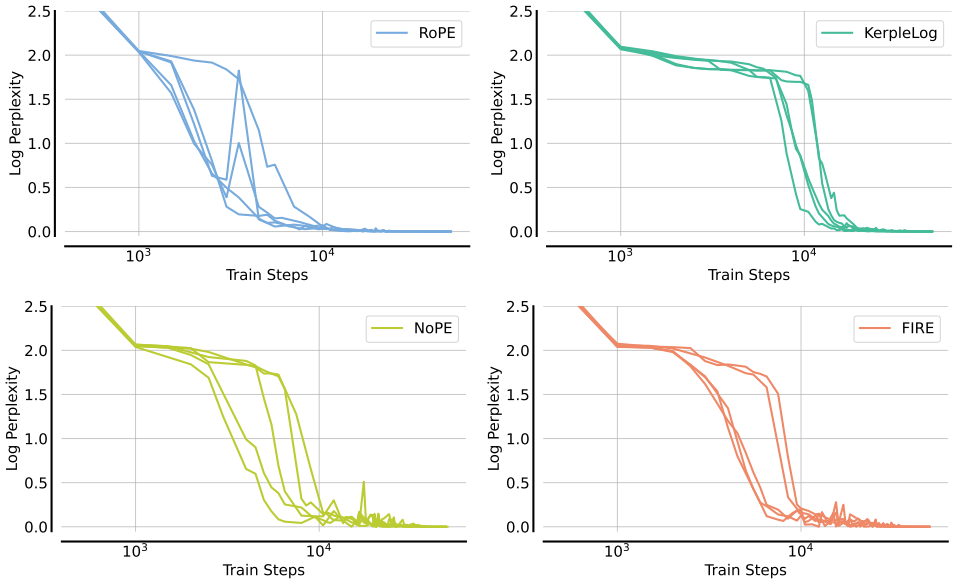


Figure G.2: Training loss over 10 trials in reverse formats. Despite similar nearly 0 log perplexity losses across runs after 10K training steps, different runs exhibit very different length generalization.

G.2 Reverse Format without Index Hint trained up to 10-digit addition

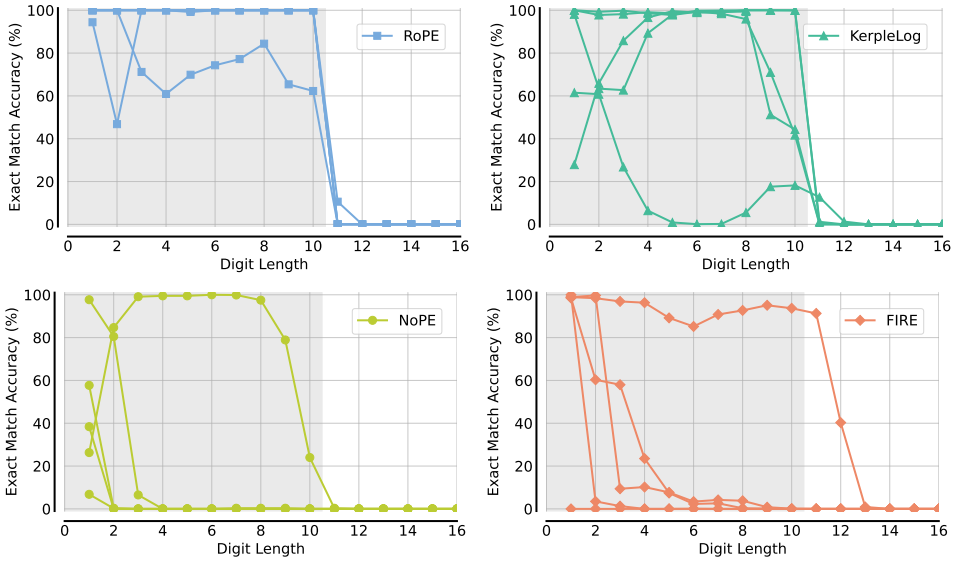


Figure G.3: Exact match accuracy on 20 to 100 digit addition of all 10 trials trained on up to 10-digit addition with index hint and reverse format using four different position encodings.

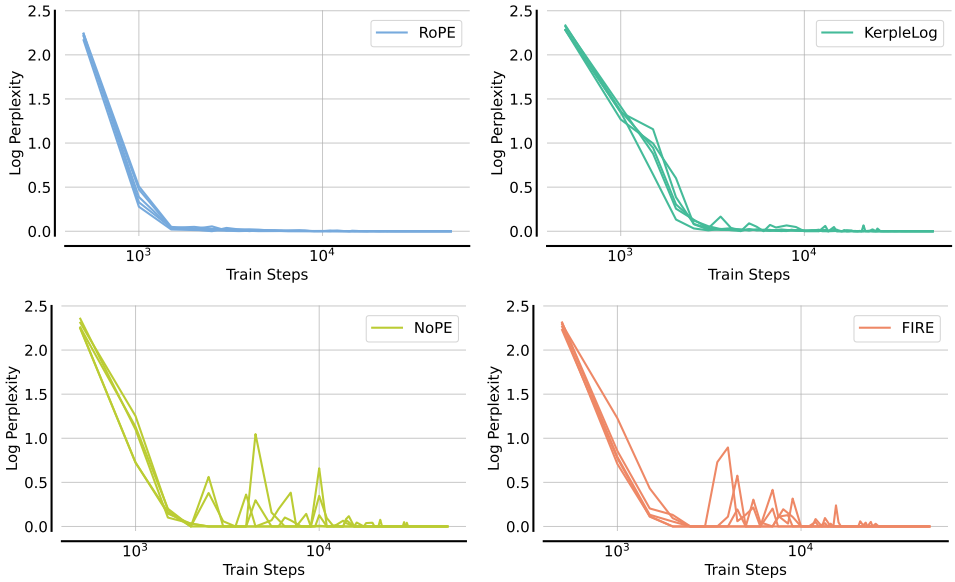


Figure G.4: Training loss over 10 trials in reverse formats. Despite similar nearly 0 log perplexity losses across runs after 10K training steps, different runs exhibit very different length generalization.

G.3 STANDARD FORMAT WITH INDEX HINT TRAINED UP TO 40

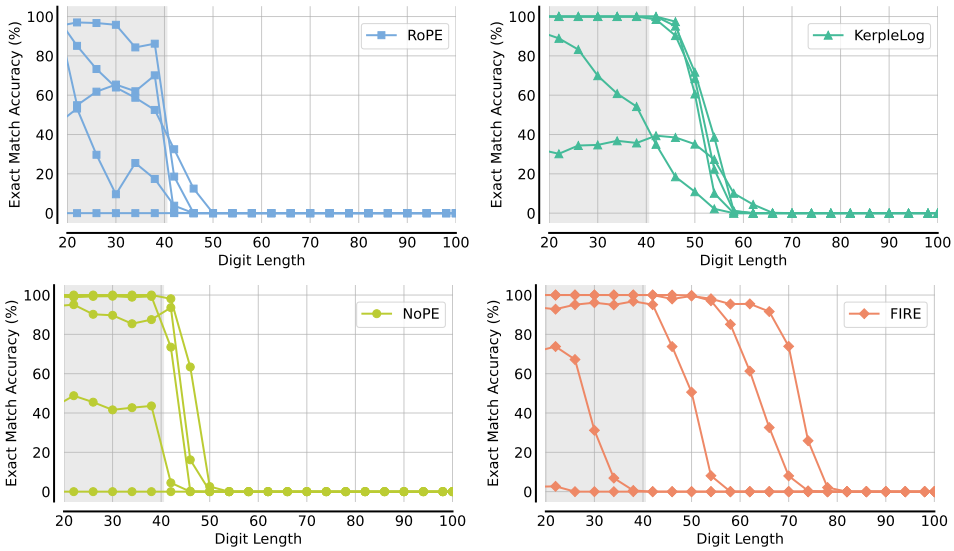


Figure G.5: Exact match accuracy on 20 to 100 digit addition of all 10 trials trained on up to 40-digit addition with index hint and reverse format using four different position encodings.

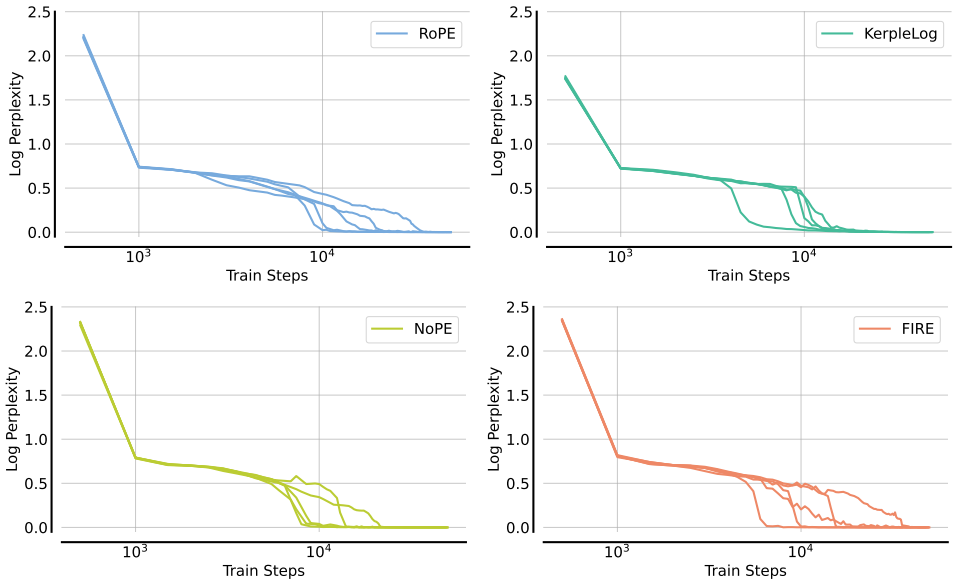


Figure G.6: Training loss over 10 trials in reverse formats. Despite similar nearly 0 log perplexity losses across runs after 10K training steps, different runs exhibit very different length generalization.

G.4 Random Space Augmentation with Reverse Format with Index Hint trained up to 40-digit addition

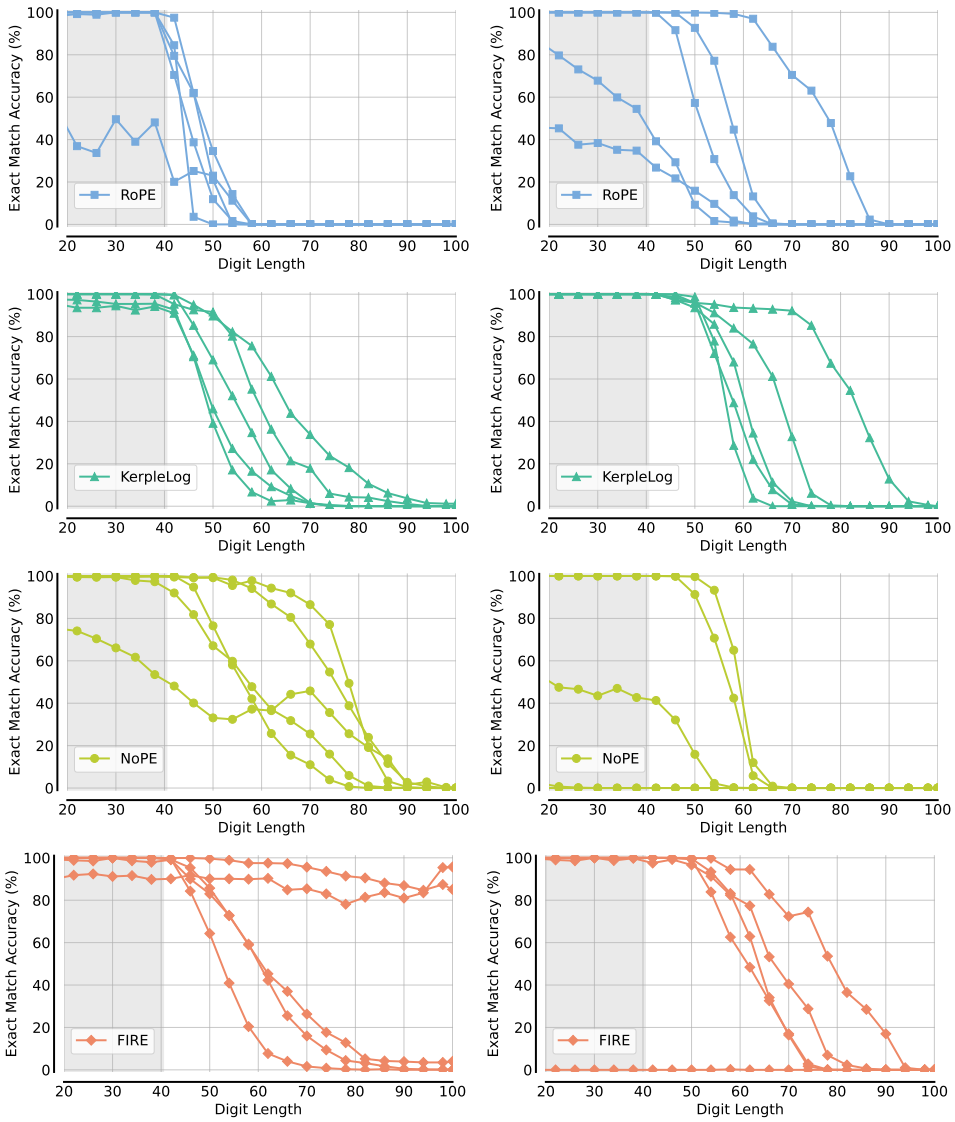


Figure G.7: (Left) With Random Space Augmentation. (Right) Without Random Space Augmentation. Exact match accuracy on 20 to 100 digit addition of all 10 trials trained on up to 40-digit addition with index hint and reverse format using four different position encodings.

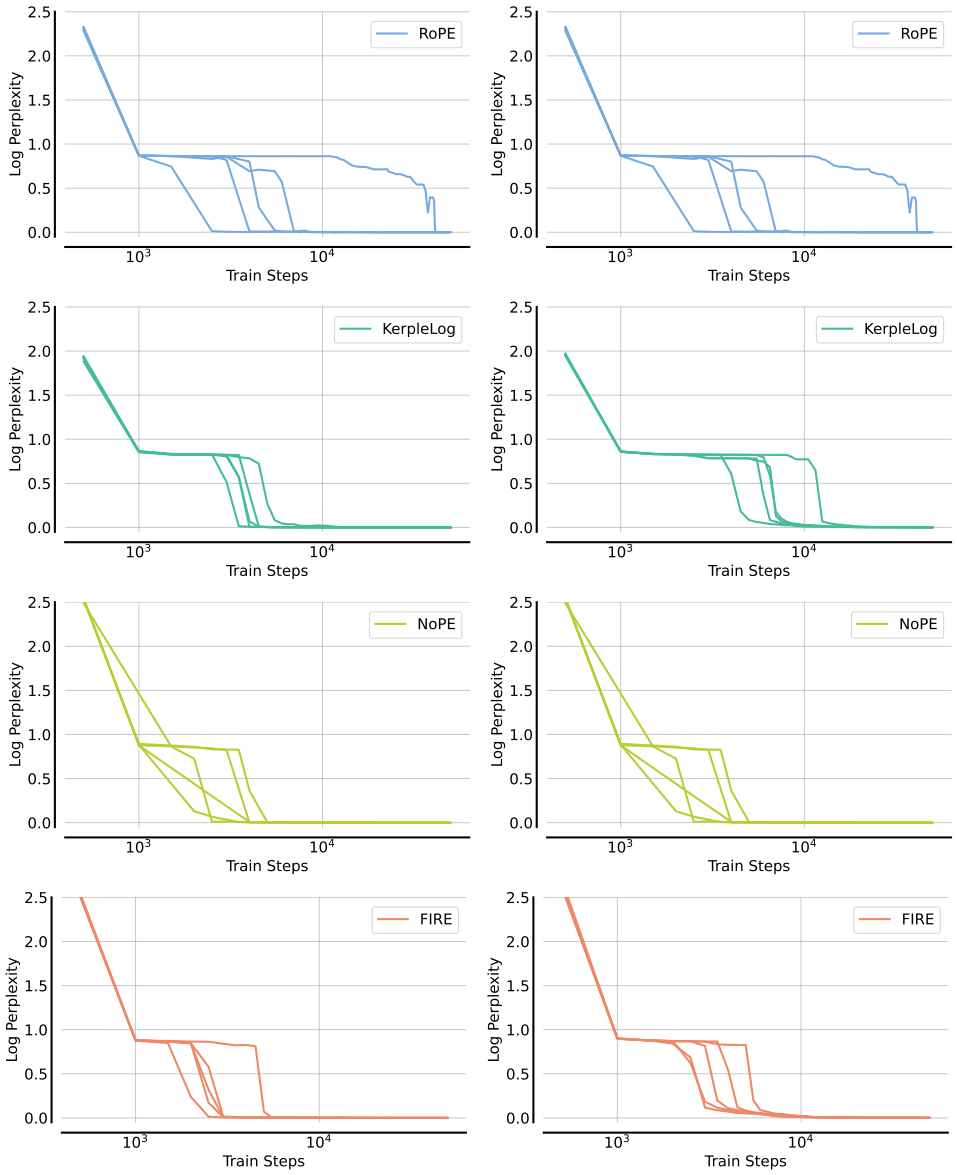


Figure G.8: (Left) Without Random Space Augmentation. (Right) With Random Space Augmentation. Training loss over 10 trials in reverse formats. Despite similar nearly 0 log perplexity losses across runs after 10K training steps, different runs exhibit very different length generalization.

G.5 Randomized Position Encoding with Reverse Format with Index Hint trained up to 40-digit addition

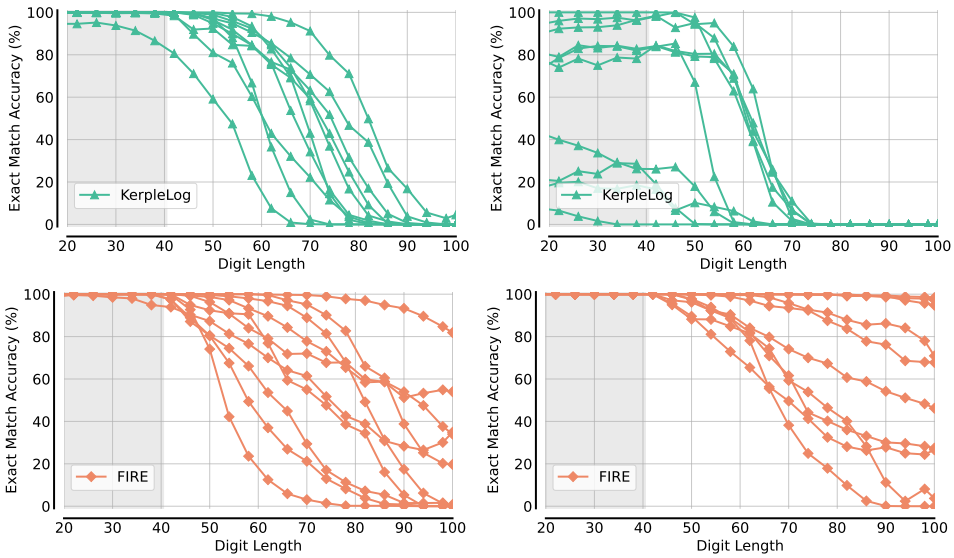


Figure G.9: (Left) Without Randomized Position Encoding (Right) With Randomized Position Encoding. Exact match accuracy on 20 to 100 digit addition of all 10 trials trained on up to 40-digit addition with index hint and reverse format using four different position encodings.

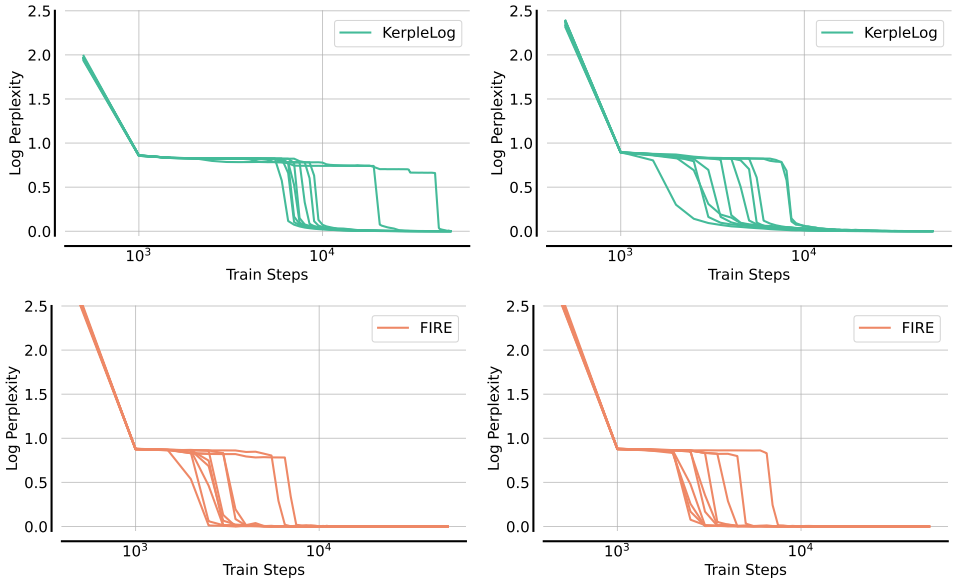


Figure G.10: (Left) Without Randomized Position Encoding (Right) With Randomized Position Encoding. Training loss over 10 trials in reverse formats.