How Do Position Encodings Affect Length Generalization? Case Studies On In-Context Function Learning

Di-Nan Lin[1], Yao Jui-Feng[2], Kun-Da Wu[2], Hao Xu[2], Chen-Hsi Huang[2], Hung-Yu Kao[3]

[1]Department of Computer Science and Information Engineering, National Cheng Kung University
[2]Pixel Software, Google
[3]Department of Computer Science, National Tsing Hua University
lindinan934301@gmail.com, hykao@cs.nthu.edu.tw

Abstract

The capability of In-Context Learning (ICL) is crucial for large language models to generalize across a wide range of tasks. By utilizing prompts, these models can accurately predict outcomes for previously unseen tasks without necessitating retraining. However, this generalization ability does not extend to the length of the inputs; the effectiveness of ICL likely diminishes with excessively long inputs, resulting in errors in the generated text. To investigate this issue, we propose a study using a dataset of In-Context functions to understand the operational mechanisms of Transformer models in ICL and length generalization. We generated data using regression and Boolean functions and employed meta-learning techniques to endow the model with ICL capabilities. Our experimental results indicate that position encodings (PEs) can significantly mitigate length generalization issues, with the most effective encoding extending the maximum input length to over eight times that of the original training length. However, further analysis revealed that while PE enhances length generalization, it compromises the model's inherent capabilities, such as its ability to generalize across different data types. Overall, our research illustrates that PEs have a pronounced positive effect on length generalization, though it necessitates a careful trade-off with data generalization performance.

Code — https://github.com/IKMLab/length-generalization-with-icl

Introduction

In recent years, the development of large language models (LLMs) has significantly advanced. Many studies investigate the mysteries of In-Context Learning (ICL) (Brown et al. 2020) with Transformer-based models by carefully controlling the training data (Garg et al. 2022; Akyürek et al. 2023; Zhou et al. 2024a). These studies aim to understand what types of data the model can learn and what algorithms the model can internalize during this process.

According to human experience and intuition, once we have learned an algorithm and understood its application, we can accurately respond to future inputs that the algorithm can solve. We hypothesize that, in the future, LLMs may also achieve this capability through ICL.

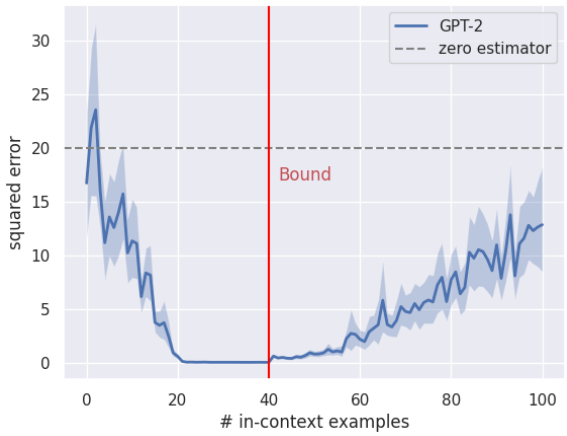


Figure 1: Trained-from-scratch GPT-2 fails on OOD length. We utilize a function dataset to train a GPT-2 from scratch. The red line indicates the maximum number of in-context examples the model has encountered during training. Results show that it is unable to achieve length generalization.

What would happen if model inferences on out-of-distribution (OOD) length? We select linear regression as the test task, setting the dimension of x to 20 and y to 1. Thus, any algorithm requires at least 20 distinct pairs (x, y) to determine the actual weight w. The model's maximum token length is 200; each pair occupies 2 tokens, allowing up to 100 in-context examples. We train the model from scratch, using squared error as a benchmark. During training, the model sees 40 in-context examples before making inferences at maximum length. As shown in Fig. 1, GPT-2 struggles with length extrapolation due to learnable position encoding, which cannot be trained on short and tested on long. In summary, our contributions are as follows.

1. We show that Position Encodings (PEs) are crucial for addressing Length Generalization. The correct PE can sustain the model's ICL capability on OOD lengths.
2. We find that appropriate PE enhances the ability of ICL in different tasks, resulting in higher prediction accuracy.
3. We demonstrate that although PEs enhance Length Generalization capability, they compromise model's inherent abilities, resulting in reduced induction ability.

Related Work

Length Generalization Length generalization primarily examines whether a model can extrapolate to sequences longer than those encountered during training. Previous work has attempted to validate length generalization using various datasets, such as command translation (Lake and Baroni 2018), mathematical addition (Lee et al. 2024; Zhou et al. 2024b,a), parity (Anil et al. 2022; Zhou et al. 2024a), logical reasoning (Zhang et al. 2023), and programming algorithms (Deletang et al. 2023; Zhou et al. 2024a). The common characteristic of these tasks is that the method of solving the problem does not change with the sequence length. In other words, length should not be the primary factor that determines difficulty. However, experimental results indicate that Transformer-based models often fail on OOD lengths. The decoder-only causal Transformer widely used in LLMs has similarly struggled with these tasks.

Most studies (Kazemnejad et al. 2023; Zhou et al. 2024b; McLeish et al. 2024; Wang et al. 2024) consider the key to achieving length generalization in Transformers to lie in PEs. Early decoder-only models, such as GPT-2 (Radford et al. 2019), used learnable PE, which resulted in poor length extrapolation abilities. Consequently, subsequent research has focused on improving PE methods to enhance length generalization.

Position Encodings In the self-attention mechanism, the concept of distance is inherently absent. To address this, Absolute Position Encoding (APE) (Vaswani et al. 2017) was proposed and added it to token embeddings, thereby incorporating positional information into the input vectors. APE is widely used in various language models, such as BERT (Devlin et al. 2019) and GPT-2. Differs from APE, Relative Position Encoding (RPE) (Shaw, Uszkoreit, and Vaswani 2018) uses a learnable relative position embedding and sums it with the key vectors k_i. T5 (Raffel et al. 2020) employs a simpler method by adding the learnable embedding as a bias term. This additive bias technique is also utilized by ALiBi (Press, Smith, and Lewis 2022) and FIRE (Li et al. 2024), and is commonly referred to as Additive Relative Position Encoding (ARPE).

In addition to these PEs, Rotary Position Encoding (RoPE) (Su et al. 2021) uses a rotation matrix to map the query and key vectors to their respective absolute positions. RoPE is widely used in most of today's LLMs. However, due to its inherent absolute position encoding characteristics and the finite number of θ_j, RoPE is not suitable for length extrapolation. To address this, Positional Interpolation (PI) (Chen et al. 2023) and SuperHot (kaiokendev 2023), which employ a scaling function to adjust the index m based on input length, requiring only minimal fine-tuning to extend context length. Dynamic Neural Tangent Kernel (NTK) (bloc97 2023) was introduced, or called Adjusted Base Frequency (ABF) (Xiong et al. 2024)), which achieves length extrapolation without fine-tuning by adjusting the base frequency θ. YaRN (Peng et al. 2024) combines the advantages of both methods using a ramp function, resulting in improved length extrapolation abilities.

In-Context Function Learning The concept of In-Context Function Learning (Garg et al. 2022; Bhattamishra et al. 2024) involves a model receiving a set of prompt inputs $(x_1, y_1, \ldots, x_n, y_n, x_{\text{query}})$, where it must respond to x_{query} based on the distribution of the previous (x, y) pairs. Specifically, given that $x_i \in \mathbb{R}^d$ is sampled from the distribution D_x and the function $f : \mathbb{R}^d \to \mathbb{R}$ is sampled from the distribution D_F, with y_i obtained through $f(x_i)$, the model's objective is to accurately predict $f(x_{\text{query}})$. In this context, the model is trained to learn the underlying algorithm, a process known as meta learning.

Fig. 2 shows the whole process of In-Context Function Learning. In-Context Function Learning requires reducing the scope of the variables, necessitating training the model from the beginning. This approach differs from typical ICL research, which usually starts with a LLM and utilizes prompts.

Setup for In-Context Function Learning

In this section, we explain the reason for training the model from scratch and demonstrate that removing position encodings is not the optimal approach for achieving length generalization. Additionally, we outline the task definition and describe the model architecture used in the experiments.

Problem Description

How to ensure that input order does not influence the results? Since our data consists of (x, y) pairs input as sequences, the order does not influence our objective, thereby eliminating positional bias. This training approach allows us to evaluate the model's performance under various OOD conditions.

Follow the previous training setup, given that our inputs x and outputs $\hat{y}$ are vectors, we introduce two learnable linear layers to transform the inputs and outputs. Specifically, $L_{\text{input}} : \mathbb{R}^{d_{\text{input}}} \to \mathbb{R}^{d_{\text{model}}}$ for input transformation and $L_{\text{output}} : \mathbb{R}^{d_{\text{model}}} \to \mathbb{R}$ for output transformation. Due to the complexity of high-dimensional x, we employ Curriculum Learning (Bengio et al. 2009) to both the dimensionality of the input x and the number of (x, y) pairs. This makes the learning process more manageable for the model and will accelerate the training process.

Tasks Selection Regression functions (Garg et al. 2022) include 4 In-Context Learning tasks: Linear Regression, Sparse Linear Regression, Decision Tree, and Two-layer ReLU neural networks. Boolean functions (Bhattamishra et al. 2024) include Conjunction, Disjunction, Sparse Disjunction, CNFs, DNFs, Sparse Majority, 0-1 Threshold Functions, Integer Halfspace, Parity, Sparse Parity, and Nearest Neighbor. In our experiments, we test model performance using the definitions of all the aforementioned tasks, except for Sparse Linear Regression, Decision Tree, Two-layer ReLU neural networks, CNFs, DNFs, and Nearest Neighbor. The descriptions of the tasks are as follows:

- *Linear Regression*: We define $F = \{f \mid f(x) = w^\top x\}$, where $w \in \mathbb{R}^d$ and d is the input dimension. Both x and w are sampled from a Gaussian distribution $\mathcal{N}(0, I_d)$.

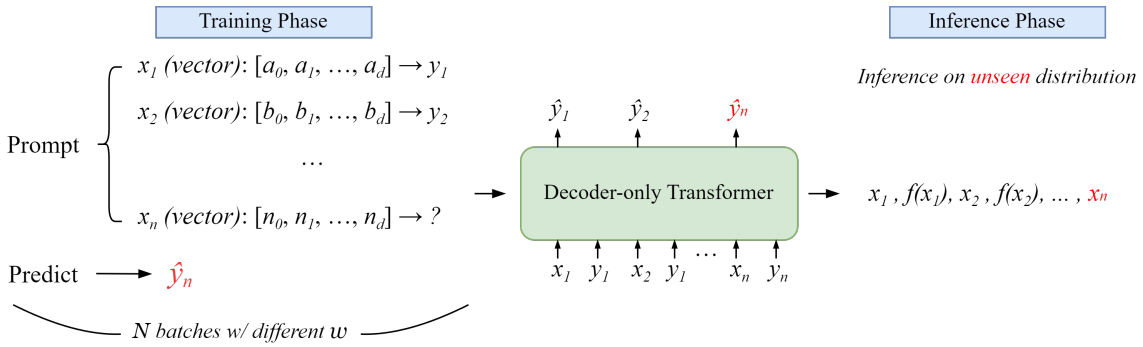


Figure 2: The overview of In-Context Function Learning. In the training phase, we sample random input-output pairs and train the model to predict $\hat{y_n}$. In the inference phase, we evaluate the model on new and unseen input-output pairs.

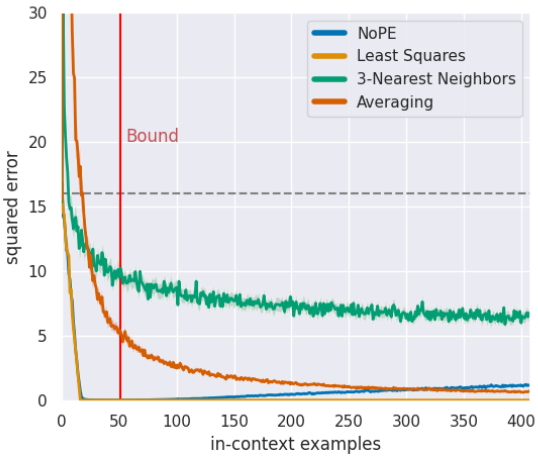


Figure 3: NoPE can alleviate but not solve the issue.

- *Conjunction*: The task is simply an $and(\wedge)$ operation of some subsets. For example, when input dimension $d = 10$, one possible input X could be $x_2 \wedge \neg x_4 \wedge x_8$. The output will be 1 only if when x_2, x_8 are 1 and x_4 is 0.
- *Disjunction*: Similar to Conjunction, Disjunction change the operation to $or(\vee)$.
- *Sparse Majority*: The task will output the majority number by selecting a subset of dimensions. For example, if the subset is $\{x_1, x_2, x_4, x_5\}$, then the output will be 1 if more than two dimensions have value 1 and be 0 otherwise.
- *0-1 Threshold Functions*: The input space is changed from $X_d = \{0, 1\}^d$ to $X_d = \{-1, 1\}^d$. Formally, the function can be define as f : sign $\left(\sum_{i=1}^{d} w_i x_i - b\right)$, where sign $\in \{-, +\}$, $w_i \in \{-1, 0, 1\}$, $b \in \{-k, \cdots, k-1, k\}$, and k is hyperparameter. We set $k = 3$ for this task.
- *Integer Halfspace*: Similar to 0-1 Threshold Functions. The task has a modification version of the function, which is f : sign $\left(\sum_{i=1}^{d} w_i x_i - 0.5\right)$.
- *Parity*: The task is a XOR$(\oplus)$ of some subsets. For example, a parity input could be $x_2 \oplus x_4 \oplus x_8$. The output will be 1 if the input has an odd number of 1s.

To prevent overfitting on the training data, we resample new x and w at each step, ensuring that the model always learns from fresh data. However, this introduces a challenge: unlike Linear Regression, the dataset for Boolean functions is finite. Considering the nature of Boolean tasks, each dimension in x can be represented in three ways: x_{d_i}, $\neg x_{d_i}$, and not selected. Consequently, for a dimension $d = 16$, there are 3^{16} possible combinations, with additional variations due to different permutations. Therefore, we believe that under these conditions, the model will not suffer from memorization issues.

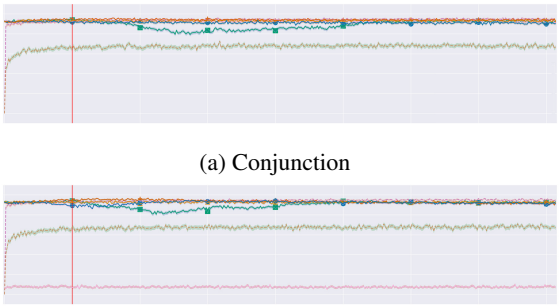
Model architecture We perform experiments utilizing a decoder-only Transformer architecture, following the design principles outlined in Llama 2 (Touvron et al. 2023). The model has 16 layers, 8 attention heads, a hidden size of 256, and a feedforward hidden size of 1024. The resultant model consists of 24.98 million parameters.

For the comparison of PE methods, our experiments include NoPE, ALiBi, FIRE, and Dynamic YaRN. In the FIRE model, we set the MLP dimension to 32. For Dynamic YaRN, the base frequency is set to 10,000 and the scaling factor to 8.

Previous study (Panwar, Ahuja, and Goyal 2024) similarly observed that models are unable to generalize on longer sequences and suggested that removing Position Encoding (NoPE) could address GPT-2's failure with OOD lengths. Consequently, we replicated the previous experiment setup, omitting the model's PE. Our findings indicated that while NoPE mitigates the problem, errors begin to emerge when the sequence length reaches twice that of the training data, as shown in Fig. 3. This prompted us to investigate whether alternative PEs might produce different outcomes.

Experiments

In this section, we will compare the performance of the 4 types of PEs on the 7 tasks. Based on the results of the characteristics of PEs, we categorize the task into 3 types:



(a) Conjunction

(b) Disjunction

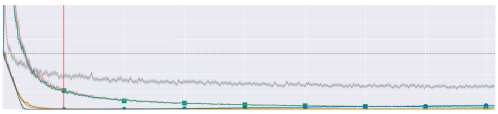
Figure 4: Easy to memorize. The figure includes the task baseline and four types of PEs. Specifically, the blue line represents NoPE, the green line represents Dynamic YaRN, the red line represents ALiBi, and the orange line represents FIRE. In the Conjunction and Disjunction tasks, since they are relatively simple, all PEs exhibit similar performance.

- **Easy to memorize:** This indicates that the model can answer correctly without needing in-context examples due to the task's simplicity or the model's ability to memorize all the rules.
- **Demonstrate ICL capability:** This indicates that the model can learn the algorithm from training tasks and correctly predict answers when encountering unseen data.
- **Hard to solve:** This indicates that the task is too difficult, and the model is unable to utilize In-Context Function Learning to learn and answer the tasks.

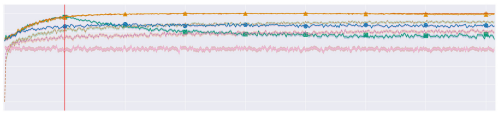
In the main experiment, we focus on testing the ability of PE to extrapolate. The results we report are inferences from previously unseen data, and the data for each dimension are the average of 1280 sequences.

Easy to memorize (Fig. 4): For Conjunction and Disjunction, since we perform an and($\wedge$) operation on each dimension (or bit) of the input x, most of the outputs are 0 for Conjunction and 1 for Disjunction. This explains why the Null Classifier performs exceptionally well. Following the original method (Bhattamishra et al. 2024), we ensure that 30% of the outputs in the training data are 1 for Conjunction and 0 for Disjunction. Despite this, the task remains relatively simple, resulting in the model achieving high accuracy when predicting the initial x_{query}. Consequently, we believe that the effect of PE on length extrapolation ability is relatively minor in this task.

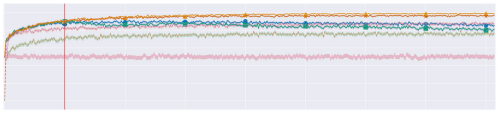
Demonstrate ICL capability (Fig. 5): For Linear Regression, the behavior of the Transformer is similar to that of the least squares algorithm, suggesting that the algorithm learned by the Transformer closely resembles least squares (Akyürek et al. 2023). However, as the OOD length increases, changes begin to manifest. Without PE, the squared error of the Transformer starts to rise at approximately twice the model length and continues to increase monotonically. In contrast, the model shows the ability of extrapolation with PEs.



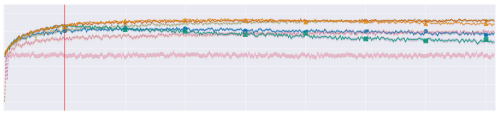
(a) Linear Regression



(b) Sparse Majority



(c) 0-1 Threshold Functions



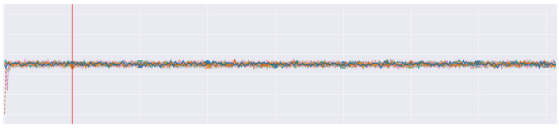
(d) Integer Halfspace

Figure 5: Demonstrate ICL capability. In tasks (a) to (d), ALiBi and FIRE exhibit the most stable performance (lower squared errors). While other PEs yield comparable results in Linear Regression, their performance declines in other tasks with a drop in accuracy.

For Integer Halfspace and Sparse Majority, it is evident that adding PE enhances the Transformer's performance. Sparse Majority is defined as outputting the majority value (0 or 1) in the specified dimensions. With a sufficient number of in-context examples, the model's accuracy can approach 100%. Therefore, we can see that the performance of ARPE still increases when the position exceeds the model length. Overall, FIRE performs slightly better than ALiBi in this task, and Dynamic YaRN no longer performs the capability of length generalization.

Conjunction and Disjunction can be considered specific instances of 0-1 Threshold Functions. Compared to these two, the label distribution of 0-1 Threshold Functions is more balanced, allowing us to observe more generalizable results. For in-distribution performance, NoPE remains the poorest performer. However, for OOD length, the accuracy of FIRE's accuracy is impacted, which is an unexpected outcome.

Hard to solve (Fig. 6): Parity is considered a particularly challenging task in previous research (Anil et al. 2022; Zhou et al. 2024a). Even with the application of Meta Learning, the Transformer still performs at a level similar to random guessing. Teaching sequence (Bhattamishra et al. 2024) may enhance the Transformer's performance. However, as our focus is on PE and the performance of OOD length, we do not utilize additional training techniques.



(a) Parity

Figure 6: Hard to solve. The Parity task is inherently difficult, leading all baselines and PEs to yield results close to random guessing.

Discussion

In this section, we will extend the main experiment by posing several questions and conducting observations to answer the question: "How do PEs affect length generalization?"

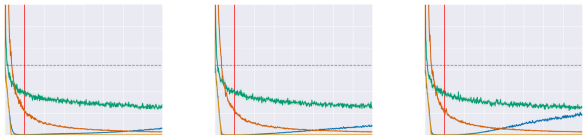
Does Increasing Model Size Enhance Performance? With the recent development of LLMs, there has been a trend to increase model sizes by adding more parameters to enhance performance. In this context, the concept of "Emergent Ability" has been introduced (Wei et al. 2022), emphasizing that a model's abilities might be underestimated when the number of parameters is below certain thresholds. Once these thresholds are surpassed, new capabilities can suddenly emerge. Although some past experimental studies have claimed that the ability for length generalization does not significantly improve with model scaling (Anil et al. 2022; Zhou et al. 2024b), we conducted tests on the performance of Linear Regression across three types of PEs to determine whether increasing the number of parameters enhances the model's length extrapolation abilities.

We categorize the models based on three parameter sizes: 12M, 25M, and 84M, with corresponding hidden sizes of 256, 256, and 512, respectively. The models have 8, 16, and 16 layers, and the number of attention heads were 4, 8, and 8, respectively. All other model parameters and training methods are kept constant.

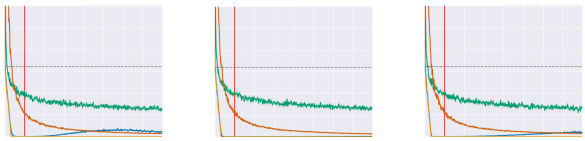
The results are shown in Fig 7. Given that our data and tasks are less complex than language learning, models with insufficient parameter sizes may show incomplete ICL capabilities, resulting in weaker length extrapolation abilities. However, when the model is sufficiently large, its performance tends to stabilize. Thus, based on the results of our primary experiments, we believe that the parameter sizes used are adequate, and the experimental data remains valuable for reference.

Can recency bias explain why Transformer fails? Recency bias is a well-known issue in sequence models, where the model tends to be influenced more by recent states, with the importance of earlier states diminishing. Since the learning objective of decoder-only Transformers shares characteristics with sequence models, it is likely that we encountered recency bias in our main experiments.

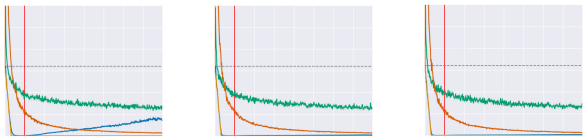
Another potential source of recency bias is RPE. Some RPE designs, such as ALiBi, cause the model to preferentially focus on recent tokens. We can observe this by analyzing the attention weights during inference. Because our data is more like a set but is input sequentially, each token should



(a) NoPE-12M (b) NoPE-25M (c) NoPE-84M



(d) FIRE-12M (e) FIRE-25M (f) FIRE-84M



(g) YaRN-12M (h) YaRN-25M (i) YaRN-84M

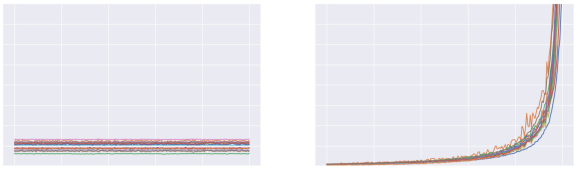
Figure 7: Blue line represents PEs, while other colors represent baselines. It is surprising that NoPE results in decreased performance when scaling. Unlike the standard and large models, the small model could not perform on par with the least squares algorithm. When the model is sufficiently large, its performance tends to stabilize.

theoretically have the same importance. If the model's attention weights favor recent inputs, it indicates the presence of recency bias in the model.

Since ALiBi incorporates a significant distance decay in its PE design, we compared it with NoPE. We averaged the attention weights of the 8 heads in each layer and displayed all the results on a single graph. In the Linear Regression task (Fig. 8) and Sparse Majority task (Fig. 9), NoPE distributed attention evenly across all tokens, suggesting that for the Transformer, all tokens in this task are equally important. In contrast, ALiBi exhibited a clear recency bias, focusing more on nearby tokens. Surprisingly, despite this bias, ALiBi performed better in our experiments (see Fig 5), which contradicts our initial hypothesis.

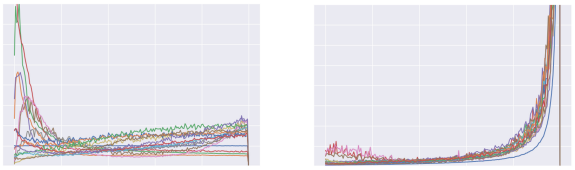
From the above observations, we conclude that recency bias does not explain why some PEs produce errors at OOD lengths. Instead, when the data is simple and each token's importance is uniform, recency bias appears to support length generalization.

Does inductive heads exist in In-Context Function Learning? A language model's ability to perform ICL can be attributed to the presence of induction heads, a mechanism described by Olsson et al. (Olsson et al. 2022). These heads enable the model to identify and leverage repeated token sequences through prefix matching. An effective model should learn from the training data, deduce a set of rules or algorithms, and then use these heads to infer new test data. In their experiments, they observed that induction heads func-



(a) NoPE (b) ALiBi

Figure 8: NoPE's and ALiBi's average attention weight for each layer. We compare the results of Linear Regression task. Note that we have normalized the token position, in which far left is the farthest token and far right is the most recent token.



(a) NoPE (b) ALiBi

Figure 9: NoPE's and ALiBi's Average attention weight for each layer on Majority task. Results show recency bias does not explain why some PEs produce errors at OOD lengths.

tion by recalling tokens. Specifically, if a combination such as $[Token_a]$ $[Token_b]$ appears in the context, then the next time $[Token_a]$ is encountered, the induction heads are likely to prompt the model to output $[Token_b]$.

Inspired by Olsson et al., we hypothesize that since the goal of In-Context Function Learning is to enable the model to learn and choose the most appropriate algorithm, it should also possess induction heads. To test this hypothesis, we design two observational experiments to investigate whether different PEs influence the function of induction heads.

We inserted multiple sets of repeated in-context examples during inference: we selected k data points to form a group and repeatedly fed these groups into the model as input, aiming to trigger the induction heads' function. Since merely examining accuracy or error rates did not yield insights, we utilized attention weights to illustrate the results. We set $k = 50$, corresponding to the maximum length of the model. We then repeated this set of data 4 times, resulting in 200 in-context examples for testing.

Given that our input is structured as $P = (x_1, y_1, \ldots, x_k, y_k, x_1, y_1, \ldots, x_k)$, the prediction target for the model, x_{query}, is x_k. Based on the experimental results by Olsson et al., x_k allocates its attention primarily to y_k. Furthermore, our findings indicate that x_k scarcely allocates attention to any x_i. Consequently, when reporting attention weights, we consider only the y_i tokens.

Our observations with NoPE suggest that Transformers inherently rely on previously encountered information to make predictions. This behavior is evident even in the absence of PE, demonstrating the model's reliance on induction heads to process repeated sequences. Otherwise, in scenarios where test data does not repeat, the model's attention weights would reflect the outcomes illustrated in Fig. 10.

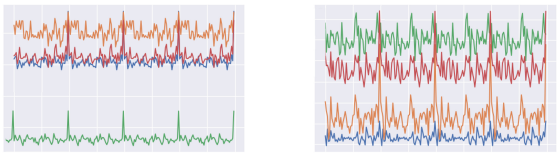


Figure 10: NoPE's attention weight for multiple sets of repeated in-context examples on Linear Regression task. The left side includes heads No.0 to No.3, and the right side includes heads No.4 to No.7.

Conversely, with the inclusion of PE, this phenomenon becomes less evident (Fig 11). We do not observe significant changes in the attention weights at all positions where y_k appears. This makes it unclear whether the model is responding based on previously encountered (x, y) pairs. Therefore, we conducted a second experiment using repeated (x, y) pairs to further investigate this behavior.

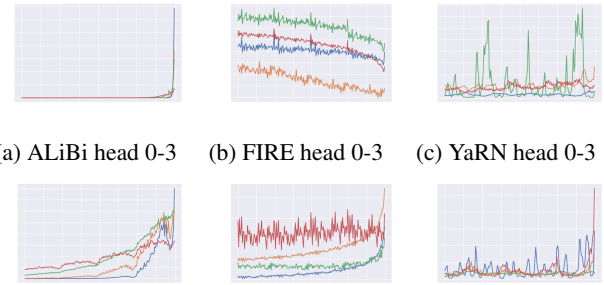
We continuously input the same data during inference: we randomly generated a new test data point and repeatedly fed it into the model until the length limit was reached. If the model is utilizing induction heads, its accuracy should significantly improve starting from the second input onward.

In the second experiment (Fig 12), unexpectedly, only NoPE and Dynamic YaRN did not produce errors due to the repetition of "non-informative" data. ALiBi, however, accumulates error rates as the distance increases, while FIRE exhibits changes within the model's maximum sequence length. We speculate that this is a form of overfitting to positional information that occurs when the Transformer is trained with PE.

Based on our observations, we conclude that In-Context Function Learning leverages the mechanisms of induction heads to handle duplicated data without failing to predict answers due to the algorithm's inability to solve for w. However, while incorporating PEs improves the model's length extrapolation capability, it appears to diminish the effectiveness of these mechanisms, suggesting a trade-off between the two.

Does serial-position effect exist in In-Context Function Learning? The serial-position effect is a psychological term introduced by Hermann Ebbinghaus (Ebbinghaus 2013). It describes the tendency for humans to remember items at the beginning (primacy effect) or at the end (recency effect) of a list more easily. Recent experiments "Lost in the Middle" (Liu et al. 2024) discovered that LLMs might also display this effect, particularly during long context ICL. Therefore, we aim to simulate the testing methods of the serial-position effect to observe whether different PEs produce varying results.

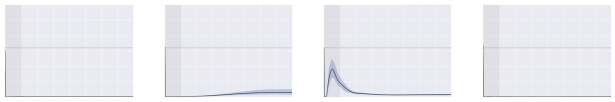
We first want to highlight the differences between In-Context Function Learning and general LLMs concerning



(a) ALiBi head 0-3 (b) FIRE head 0-3 (c) YaRN head 0-3

(d) ALiBi head 4-7 (e) FIRE head 4-7 (f) YaRN head 4-7

Figure 11: Attention weight for multiple sets of repeated in-context examples results with 3 different PEs.



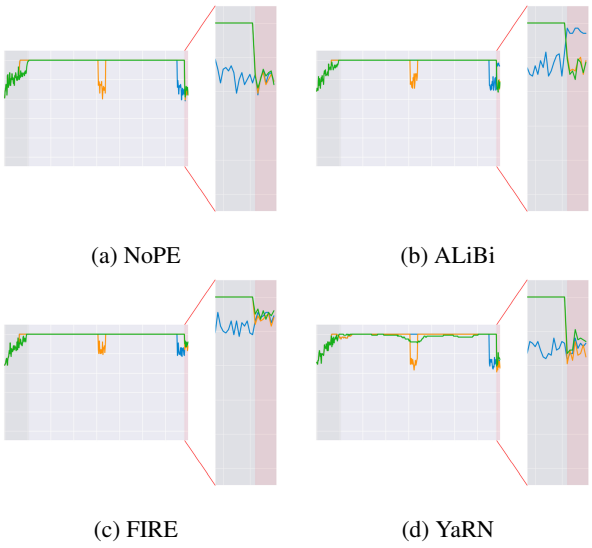
(a) NoPE (b) ALiBi (c) FIRE (d) YaRN

Figure 12: Duplicated data attack on NoPE, ALiBi, FIRE, and Dynamic YaRN.

training data. Our models are trained on data that lacks positional bias to ensure that the length extrapolation experiments are influenced solely by the PE variable, without any external interference. Consequently, unlike LLMs whose behavior may be altered by variations in training data, our experimental variables are limited to the test data and PE.

Building on the previous settings, we conducted a serial-position effect experiment. In the serial-position effect experiment, the input prompt P was altered to $(instruction + answer + noise + target)$, and the input dimension x is 16, meaning the model requires 16 in-context examples to calculate w. Consequently, we set 32 in-context examples in the $instruction$. Following this, we generated $noise$ by repeatedly duplicating the last piece of data in the $instruction$. After creating k pieces of $noise$, we appended the $target$, which included 8 in-context examples. $answer$ comprises a sequence of 16 in-context examples, with the last 8 examples being the actual $target$. By changing the position of the $answer$, we seek to evaluate the model's performance.

Our focus is on the $target$ block marked with a red background. According to the serial-position effect, $target$ items at the beginning (highlighted in green) and the end (highlighted in blue) typically achieve the highest accuracy, while those in the middle (highlighted in orange) are more prone to being forgotten, leading to lower accuracy. Using NoPE as the baseline, we found that, due to the self-attention mechanism lacking the concept of distance and the absence of implicit positional information from the training data, NoPE demonstrated similar performance across all three $target$ positions. This indicates that without explicit positional cues, the model's accuracy is evenly distributed regardless of the $answer$'s position.



(a) NoPE (b) ALiBi

(c) FIRE (d) YaRN

Figure 13: Lost in the Middle test on Sparse Majority. Changing the position of $answer$ on different PEs.

In terms of our findings (Fig. 13), apart from ALiBi demonstrating a clear recency effect, we observed that most PEs did not exhibit the "Lost in the Middle" issue. We attribute ALiBi's recency effect to its use of a position mask during computation, which directs the model to focus more on nearby tokens. This results in higher accuracy at the end (blue lines) compared to other positions. The other PEs did not show either a primacy or a recency effect. Therefore, we conclude that this serial-position effect is independent of the PE. Given that most modern LLMs use RoPE-based PEs, if a model is found to have issues related to the serial-position effect, it is more likely due to bias introduced during the training data phase.

Limitations and Conclusion

This study examines the impact of PEs on Length Generalization in ICL. To remove positional bias, we use datasets from regression and boolean functions. Results show NoPE is suboptimal, often reducing performance, while ARPE best balances Length Generalization and accuracy.

Our analysis reveals recency bias does not explain why decoder-only models struggle with OOD lengths. Models with NoPE develop inductive heads that aid ICL, but other PEs disrupt this, weakening induction capability. Lastly, we explored the 'Lost in the Middle' phenomenon, finding that except for ALiBi's recency effect, PEs do not induce primacy or recency biases.

While we hope these findings offer language modeling more comprehensive metrics for evaluating PEs and enhance understanding of their strengths and weaknesses, our experimental scope is limited to self-attention-based auto-regressive models. Additionally, variability in function classes and text characteristics presents further challenges. Thus, several research questions remain unanswered before we can truly achieve Length Generalization.

Acknowledgements

We sincerely thank the reviewers for their thorough reading of our manuscript and for providing insightful feedback that helped us identify and address weaknesses in our work. Their comments greatly improved the clarity and rigor of our study. We are also deeply grateful to Yun-Yu Hu and Runn Prasoprat for their valuable suggestions on experimental design. Additionally, we would like to extend our heartfelt thanks to Ying-Jia Lin, Zhi-Quan Feng, and Yu-Xiang Hong for their guidance in discussing the experimental results and assisting in the manuscript preparation.

References

Akyürek, E.; Schuurmans, D.; Andreas, J.; Ma, T.; and Zhou, D. 2023. What learning algorithm is in-context learning? Investigations with linear models. In *The Eleventh International Conference on Learning Representations*.

Anil, C.; Wu, Y.; Andreassen, A. J.; Lewkowycz, A.; Misra, V.; Ramasesh, V. V.; Slone, A.; Gur-Ari, G.; Dyer, E.; and Neyshabur, B. 2022. Exploring Length Generalization in Large Language Models. In Oh, A. H.; Agarwal, A.; Belgrave, D.; and Cho, K., eds., *Advances in Neural Information Processing Systems*.

Bengio, Y.; Louradour, J.; Collobert, R.; and Weston, J. 2009. Curriculum learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, 41–48. New York, NY, USA: Association for Computing Machinery. ISBN 9781605585161.

Bhattamishra, S.; Patel, A.; Blunsom, P.; and Kanade, V. 2024. Understanding In-Context Learning in Transformers and LLMs by Learning to Learn Discrete Functions. In *The Twelfth International Conference on Learning Representations*.

bloc97. 2023. NTK-Aware Scaled RoPE allows LLaMA models to have extended (8k+) context size without any fine-tuning and minimal perplexity degradation. https://www.reddit.com/r/LocalLLaMA/comments/14lz7j5/ntkaware_scaled_rope_allows_llama_models_to_have/.

Brown, T.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J. D.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; Agarwal, S.; Herbert-Voss, A.; Krueger, G.; Henighan, T.; Child, R.; Ramesh, A.; Ziegler, D.; Wu, J.; Winter, C.; Hesse, C.; Chen, M.; Sigler, E.; Litwin, M.; Gray, S.; Chess, B.; Clark, J.; Berner, C.; McCandlish, S.; Radford, A.; Sutskever, I.; and Amodei, D. 2020. Language Models are Few-Shot Learners. In Larochelle, H.; Ranzato, M.; Hadsell, R.; Balcan, M.; and Lin, H., eds., *Advances in Neural Information Processing Systems*, volume 33, 1877–1901. Curran Associates, Inc.

Chen, S.; Wong, S.; Chen, L.; and Tian, Y. 2023. Extending Context Window of Large Language Models via Positional Interpolation. arXiv:2306.15595.

Deletang, G.; Ruoss, A.; Grau-Moya, J.; Genewein, T.; Wenliang, L. K.; Catt, E.; Cundy, C.; Hutter, M.; Legg, S.; Veness, J.; and Ortega, P. A. 2023. Neural Networks and the Chomsky Hierarchy. In *The Eleventh International Conference on Learning Representations*.

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In Burstein, J.; Doran, C.; and Solorio, T., eds., *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, 4171–4186. Minneapolis, Minnesota: Association for Computational Linguistics.

Ebbinghaus, H. 2013. Memory: A contribution to experimental psychology. *Annals of neurosciences*, 20(4): 155.

Garg, S.; Tsipras, D.; Liang, P.; and Valiant, G. 2022. What Can Transformers Learn In-Context? A Case Study of Simple Function Classes. In Oh, A. H.; Agarwal, A.; Belgrave, D.; and Cho, K., eds., *Advances in Neural Information Processing Systems*.

kaiokendev. 2023. Extending Context is Hard but not Impossible. https://kaiokendev.github.io/context. Accessed: 2023-06-20.

Kazemnejad, A.; Padhi, I.; Natesan Ramamurthy, K.; Das, P.; and Reddy, S. 2023. The Impact of Positional Encoding on Length Generalization in Transformers. In Oh, A.; Naumann, T.; Globerson, A.; Saenko, K.; Hardt, M.; and Levine, S., eds., *Advances in Neural Information Processing Systems*, volume 36, 24892–24928. Curran Associates, Inc.

Lake, B.; and Baroni, M. 2018. Generalization without Systematicity: On the Compositional Skills of Sequence-to-Sequence Recurrent Networks. In Dy, J.; and Krause, A., eds., *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, 2873–2882. PMLR.

Lee, N.; Sreenivasan, K.; Lee, J. D.; Lee, K.; and Papailiopoulos, D. 2024. Teaching Arithmetic to Small Transformers. In *The Twelfth International Conference on Learning Representations*.

Li, S.; You, C.; Guruganesh, G.; Ainslie, J.; Ontanon, S.; Zaheer, M.; Sanghai, S.; Yang, Y.; Kumar, S.; and Bhojanapalli, S. 2024. Functional Interpolation for Relative Positions improves Long Context Transformers. In *The Twelfth International Conference on Learning Representations*.

Liu, N. F.; Lin, K.; Hewitt, J.; Paranjape, A.; Bevilacqua, M.; Petroni, F.; and Liang, P. 2024. Lost in the Middle: How Language Models Use Long Contexts. *Transactions of the Association for Computational Linguistics*, 12: 157–173.

McLeish, S.; Bansal, A.; Stein, A.; Jain, N.; Kirchenbauer, J.; Bartoldson, B. R.; Kailkhura, B.; Bhatele, A.; Geiping, J.; Schwarzschild, A.; and Goldstein, T. 2024. Transformers Can Do Arithmetic with the Right Embeddings. *arXiv preprint arXiv:2405.17399*.

Olsson, C.; Elhage, N.; Nanda, N.; Joseph, N.; DasSarma, N.; Henighan, T.; Mann, B.; Askell, A.; Bai, Y.; Chen, A.; Conerly, T.; Drain, D.; Ganguli, D.; Hatfield-Dodds, Z.; Hernandez, D.; Johnston, S.; Jones, A.; Kernion, J.; Lovitt, L.; Ndousse, K.; Amodei, D.; Brown, T.; Clark, J.; Kaplan, J.; McCandlish, S.; and Olah, C. 2022. In-context Learning and Induction Heads. *Transformer Circuits*

Thread. Https://transformer-circuits.pub/2022/in-context-learning-and-induction-heads/index.html.

Panwar, M.; Ahuja, K.; and Goyal, N. 2024. In-Context Learning through the Bayesian Prism. In *The Twelfth International Conference on Learning Representations*.

Peng, B.; Quesnelle, J.; Fan, H.; and Shippole, E. 2024. YaRN: Efficient Context Window Extension of Large Language Models. In *The Twelfth International Conference on Learning Representations*.

Press, O.; Smith, N.; and Lewis, M. 2022. Train Short, Test Long: Attention with Linear Biases Enables Input Length Extrapolation. In *International Conference on Learning Representations*.

Radford, A.; Wu, J.; Child, R.; Luan, D.; Amodei, D.; and Sutskever, I. 2019. Language Models are Unsupervised Multitask Learners.

Raffel, C.; Shazeer, N.; Roberts, A.; Lee, K.; Narang, S.; Matena, M.; Zhou, Y.; Li, W.; and Liu, P. J. 2020. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *Journal of Machine Learning Research*, 21(140): 1–67.

Shaw, P.; Uszkoreit, J.; and Vaswani, A. 2018. Self-Attention with Relative Position Representations. In Walker, M.; Ji, H.; and Stent, A., eds., *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, 464–468. New Orleans, Louisiana: Association for Computational Linguistics.

Su, J.; Lu, Y.; Pan, S.; Wen, B.; and Liu, Y. 2021. RoFormer: Enhanced Transformer with Rotary Position Embedding. arXiv:2104.09864.

Touvron, H.; Martin, L.; Stone, K.; Albert, P.; Almahairi, A.; Babaei, Y.; Bashlykov, N.; Batra, S.; Bhargava, P.; Bhosale, S.; Bikel, D.; Blecher, L.; Ferrer, C. C.; Chen, M.; Cucurull, G.; Esiobu, D.; Fernandes, J.; Fu, J.; Fu, W.; Fuller, B.; Gao, C.; Goswami, V.; Goyal, N.; Hartshorn, A.; Hosseini, S.; Hou, R.; Inan, H.; Kardas, M.; Kerkez, V.; Khabsa, M.; Kloumann, I.; Korenev, A.; Koura, P. S.; Lachaux, M.-A.; Lavril, T.; Lee, J.; Liskovich, D.; Lu, Y.; Mao, Y.; Martinet, X.; Mihaylov, T.; Mishra, P.; Molybog, I.; Nie, Y.; Poulton, A.; Reizenstein, J.; Rungta, R.; Saladi, K.; Schelten, A.; Silva, R.; Smith, E. M.; Subramanian, R.; Tan, X. E.; Tang, B.; Taylor, R.; Williams, A.; Kuan, J. X.; Xu, P.; Yan, Z.; Zarov, I.; Zhang, Y.; Fan, A.; Kambadur, M.; Narang, S.; Rodriguez, A.; Stojnic, R.; Edunov, S.; and Scialom, T. 2023. Llama 2: Open Foundation and Fine-Tuned Chat Models. arXiv:2307.09288.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, L. u.; and Polosukhin, I. 2017. Attention is All you Need. In Guyon, I.; Luxburg, U. V.; Bengio, S.; Wallach, H.; Fergus, R.; Vishwanathan, S.; and Garnett, R., eds., *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Wang, J.; Ji, T.; Wu, Y.; Yan, H.; Gui, T.; Zhang, Q.; Huang, X.; and Wang, X. 2024. Length Generalization of Causal Transformers without Position Encoding. arXiv:2404.12224.

Wei, J.; Tay, Y.; Bommasani, R.; Raffel, C.; Zoph, B.; Borgeaud, S.; Yogatama, D.; Bosma, M.; Zhou, D.; Metzler, D.; Chi, E. H.; Hashimoto, T.; Vinyals, O.; Liang, P.; Dean, J.; and Fedus, W. 2022. Emergent Abilities of Large Language Models. *Transactions on Machine Learning Research*. Survey Certification.

Xiong, W.; Liu, J.; Molybog, I.; Zhang, H.; Bhargava, P.; Hou, R.; Martin, L.; Rungta, R.; Sankararaman, K. A.; Oguz, B.; Khabsa, M.; Fang, H.; Mehdad, Y.; Narang, S.; Malik, K.; Fan, A.; Bhosale, S.; Edunov, S.; Lewis, M.; Wang, S.; and Ma, H. 2024. Effective Long-Context Scaling of Foundation Models. In Duh, K.; Gomez, H.; and Bethard, S., eds., *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, 4643–4663. Mexico City, Mexico: Association for Computational Linguistics.

Zhang, Y.; Backurs, A.; Bubeck, S.; Eldan, R.; Gunasekar, S.; and Wagner, T. 2023. Unveiling Transformers with LEGO: A Synthetic Reasoning Task.

Zhou, H.; Bradley, A.; Littwin, E.; Razin, N.; Saremi, O.; Susskind, J. M.; Bengio, S.; and Nakkiran, P. 2024a. What Algorithms can Transformers Learn? A Study in Length Generalization. In *The Twelfth International Conference on Learning Representations*.

Zhou, Y.; Alon, U.; Chen, X.; Wang, X.; Agarwal, R.; and Zhou, D. 2024b. Transformers Can Achieve Length Generalization But Not Robustly. In *ICLR 2024 Workshop on Mathematical and Empirical Understanding of Foundation Models*.