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

位置編碼如何影響長度泛化性？基於上下文函數學習的案例研究

Author: Di-Nan Lin（林諦南）

Advisor: Prof. Hung-Yu Kao（高宏宇教授）

National Cheng Kung University
Department of Computer Science and Information Engineering
Intelligent Knowledge Management Laboratory

Outline

- **Introduction**
  - Background
  - Motivation
- **Related Work**
  - Length Generalizaiton
  - Position Encodings
  - In-Context Learning
- **Methodology**
- **Experiments**
- **Analysis**
  - Recency Bias
  - Inductive Bias
  - Serial-Position Effect
- **Conclusion**

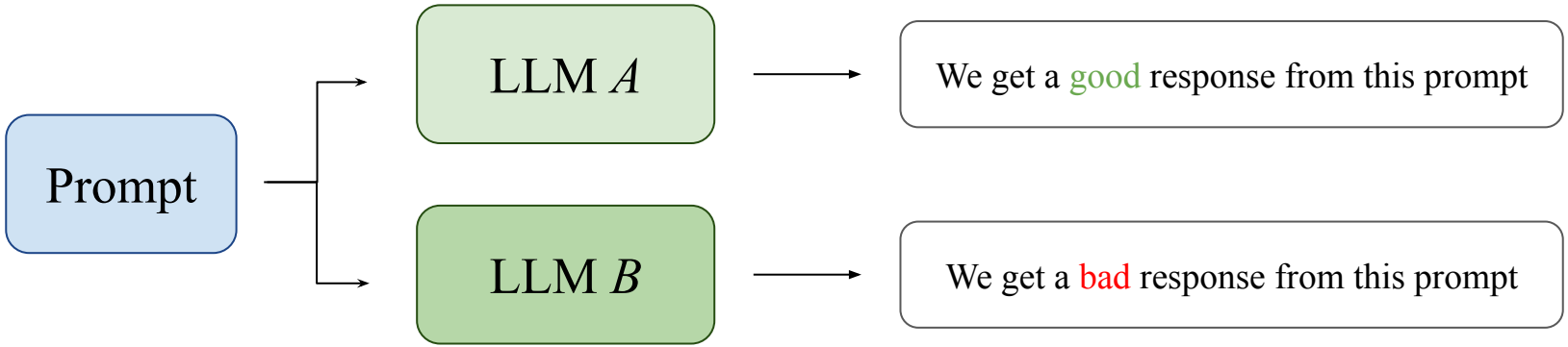
Introduction

- **Backgroud**
  - LLM fails on long context.
- **Motivation**
  - Memorization Capacity
- **Our Work**

LLM is unpredictable

Prompt Engineering = In-Context Learning (ICL)

Prompt Engineering has become a daily work for people nowadays.

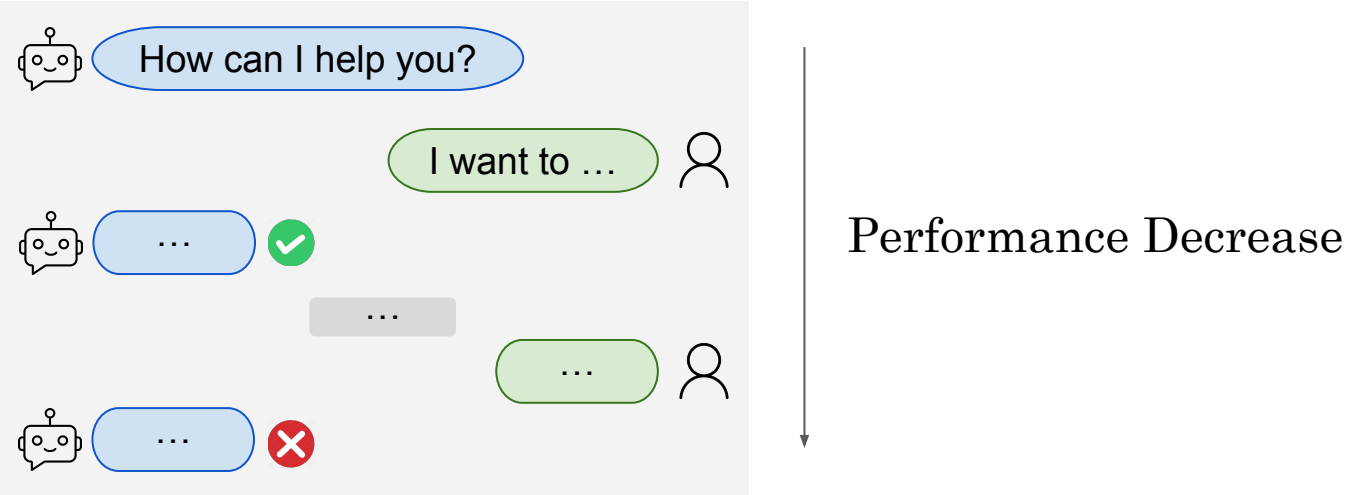


▲ Figure: Unpredictable behavior for the same prompt.

We cannot definitively ascertain why specific prompts enhance performance or why the same prompt may yield suboptimal results with another LLM.

LLM also lost in long context

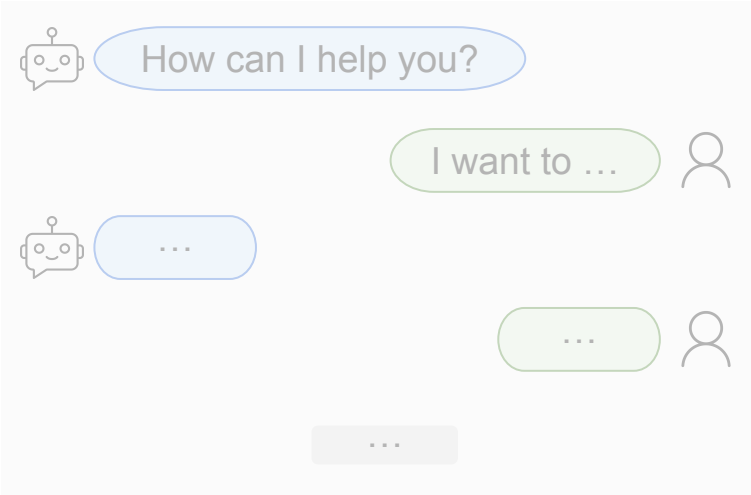
We might have extended conversations on a single topic with an LLM, only to find that it sometimes **forgets or overlooks** certain details.



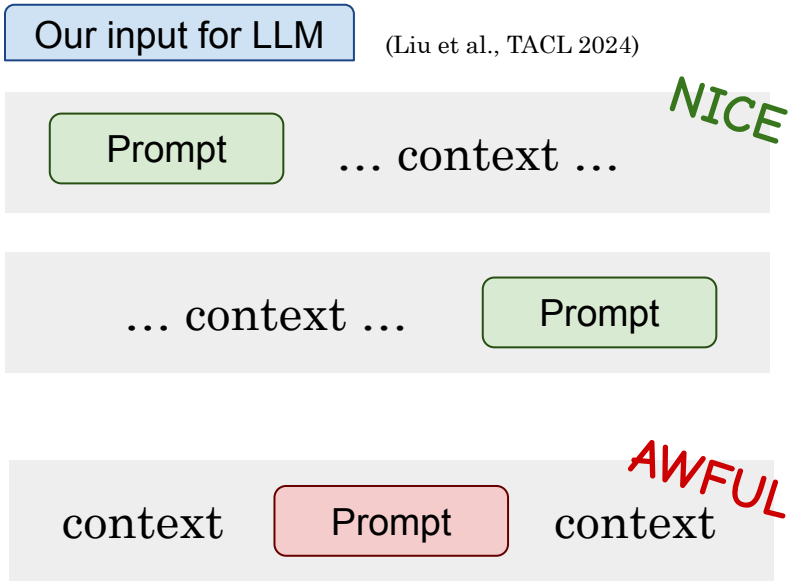
▲ Figure: LLM fails when context get longer.

LLM also lost in long context

We might have extended conversations on a single topic with an LLM, only to find that it sometimes **forgets or overlooks** certain details.



▲ Figure: LLM fails when context get longer.



Why LLM cannot perform ICL in long context?

Intuition

Similar to humans, LLM have a **limited memorization capacity**.

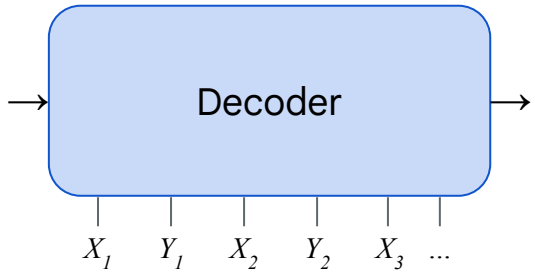
Model tend to forget because its parameters only sufficient to memorize these info.

Set up an experiment

Train from scratch

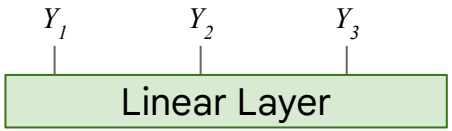
NER Dataset (CoNLL 2003)

EU<b-org> rejects<o> German<b-misc>
call<o> to<o> boycott<o> British<b-misc>
lamb<o> .<o>



Output labels

3, 0, 7, 0, 0, 0, 7, 0, 0

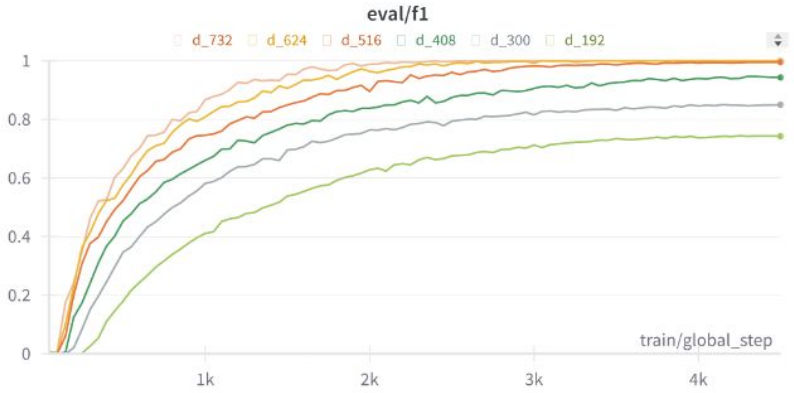


Validating the hypothesis of memorizatoin capacity

Dataset	
Data avg length	21.19
Total samples	1400

We let

Train set = Valid set



hidden size	f1-score
d_192	0.74
d_300	0.85
d_408	0.94
d_516	0.99
d_624	0.99
d_732	1

▲ Figure: Overview of the results. [Thesis p.2]

Validating the hypothesis of memorizatoin capacity

Dataset

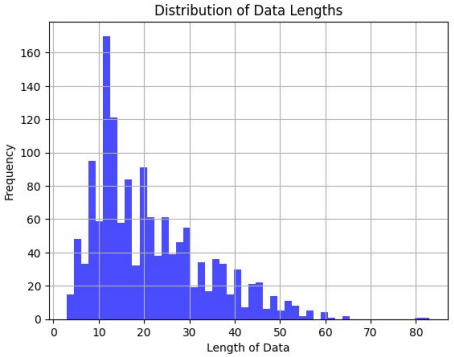
Data avg length	21.19
Total samples	1400

We let

Train set = Test set (concat all data)

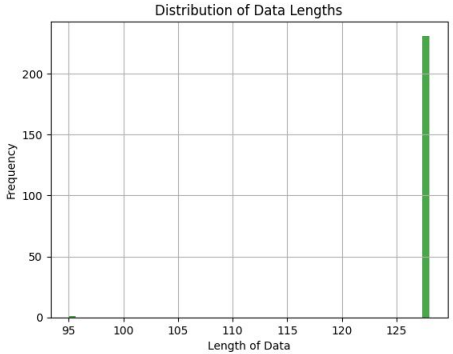
but has longer length.

Train on

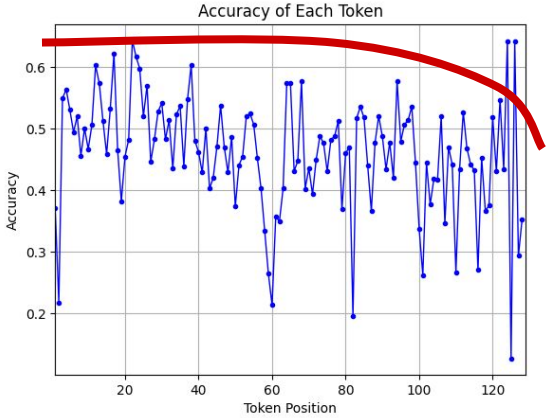


▲ Train set

Test on



▲ Test set



▲ Figure: Results. [Thesis p.3]

What makes the experiment fail?

1. In the NER task, words are **not independent**, and mutual influence during concatenation may have caused a decline in accuracy.
2. Training data mainly consists of short sentences. The uneven data length distribution may hurt the model and **let some positions underfitting**.

Find a dataset without any positional bias (order does not influence results).

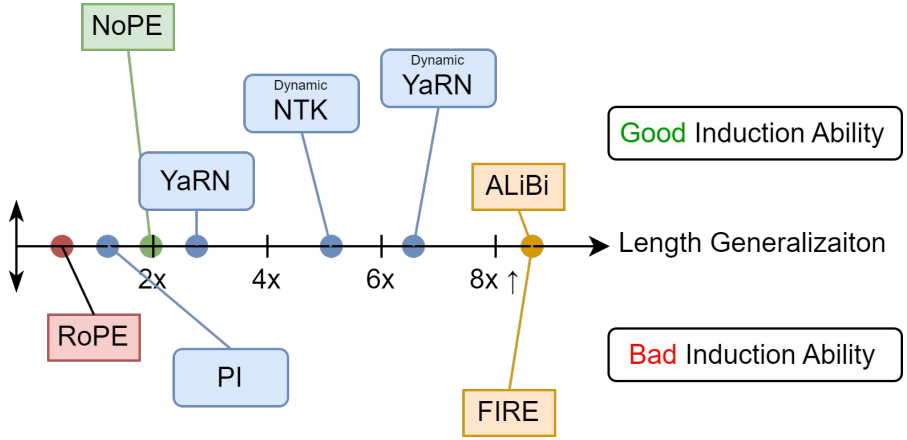
→ Function Dataset

The key to achieving length generalization in LLM?

→ Position Encodings (PEs)

Overall, our contributions are:

1. PEs sustain model's ICL capability on out-of-distribution (OOD) lengths.
2. PEs enhance ICL capability across different tasks and increase accuracy.
3. PEs might reduce Induction Ability, which is a trade-off between length.



▲ Figure: Different performance between PEs. [Thesis p.4]

Length Generalization:

Model can extrapolate to longer text.

Induction Ability:

Model can utilize induction heads.

Related Work

- **Length Generalization**
  - Computational complexity
- **Position Encodings**
  - The key to extrapolate
- **In-Context Learning**
  - Learning to Learn

Length Generalization

Goal

Teach (train) model how to add (**learn an algorithm**).

$14 + 32 \rightarrow 55$

$123 + 654 \rightarrow 777$

Problem

Carry operation is not easy:

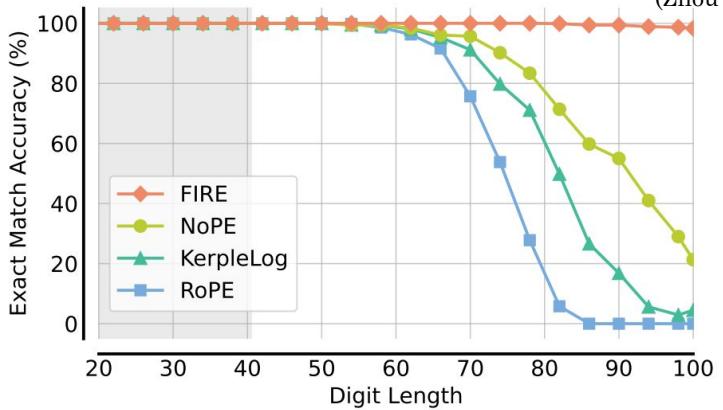
$5678 + 9012 \rightarrow ?????$

The digit length is a challenge as well.

Results

*Current SoTA can only generalize to lengths 2.5x longer than training set.

(Zhou et al., ICLR 2024)



Length Generalization

▼ Table: Examples of the tasks for length generalization. [Thesis p.6]

Task	Example Input	Example Output
Classical Length Generalization Dataset		
SCAN (Lake et al., 2018)	jump left and walk	LTURN JUMP WALK
Mathematical and Reasoning Tasks		
Addition (Anil et al., 2022)	659 + 103 =	762
Parity (Anil et al., 2022)	0 1 1 0 1	1 (odd number of 1s)
LEGO (Zhang et al., 2023)	*a=+1; b=-a; e=+b; d=-f; f=+e. f =*	1
Programming		
Stack (Deletang et al., 2023)	*abbaa* POP PUSH *a* POP	*abba*
Sort (Deletang et al., 2023)	4 12 3 7	3 4 7 12

Length Generalization

▼ Table: Examples of the tasks for length generalization. [Thesis p.6]

Task	Example Input	Example Output
	Classical Length Generalization Dataset	
Lake et al., 2018 SCAN	jump left and walk	LTURN JUMP WALK
	Mathematical and Reasoning Tasks	
Addition	659 + 103 =	762
Anil et al., 2022 Parity	0 1 1 0 1	1 (odd number of 1s)
Zhang et al., 2023 LEGO	*a=+1; b=-a; e=+b; d=-f; f=+e. f =*	1
	Programming	
Stack	*abbaa* POP PUSH *a* POP	*abba*
Deletang et al., 2023 Sort	4 12 3 7	3 4 7 12

Length Generalization

▼ Table: Examples of the tasks for length generalization. [Thesis p.6]

Task	Example Input	Example Output
Classical Length Generalization Dataset		
SCAN (Lake et al., 2018)	jump left and walk	LTURN JUMP WALK
Mathematical and Reasoning Tasks		
Addition	659 + 103 =	762
Parity (Anil et al., 2022)	0 1 1 0 1	1 (odd number of 1s)
LEGO (Zhang et al., 2023)	*a=+1; b=-a; e=+b; d=-f; f=+e. f =*	1
Programming		
Stack (Deletang et al., 2023)	*abbaa* POP PUSH *a* POP	*abba*
Sort	4 12 3 7	3 4 7 12

Why Transformer struggles with OOD length?

1. You need lots of GPUs to train a long context LLM due to the **computational complexity** of self-attention is $O(n^2 \cdot d)$

2. You use learnable position encoding like what GPT-2 does, and it is impossible to do length generalization.

3. Data distribution? Architecture constraints? It's still a research question. But most think the key is **Position Encoding**.

Position Encodings (PEs)

Absolute Position Encoding (APE): (Vaswani et al., 2017)

- Learnable position encoding (BERT, GPT-2)
- Sinusoidal Position Encoding (Vanilla Transformer):

$$PE_{(\text{pos},\ 2i)} = \sin\left(\frac{\text{pos}}{10000^{\frac{2i}{d}}}\right)$$

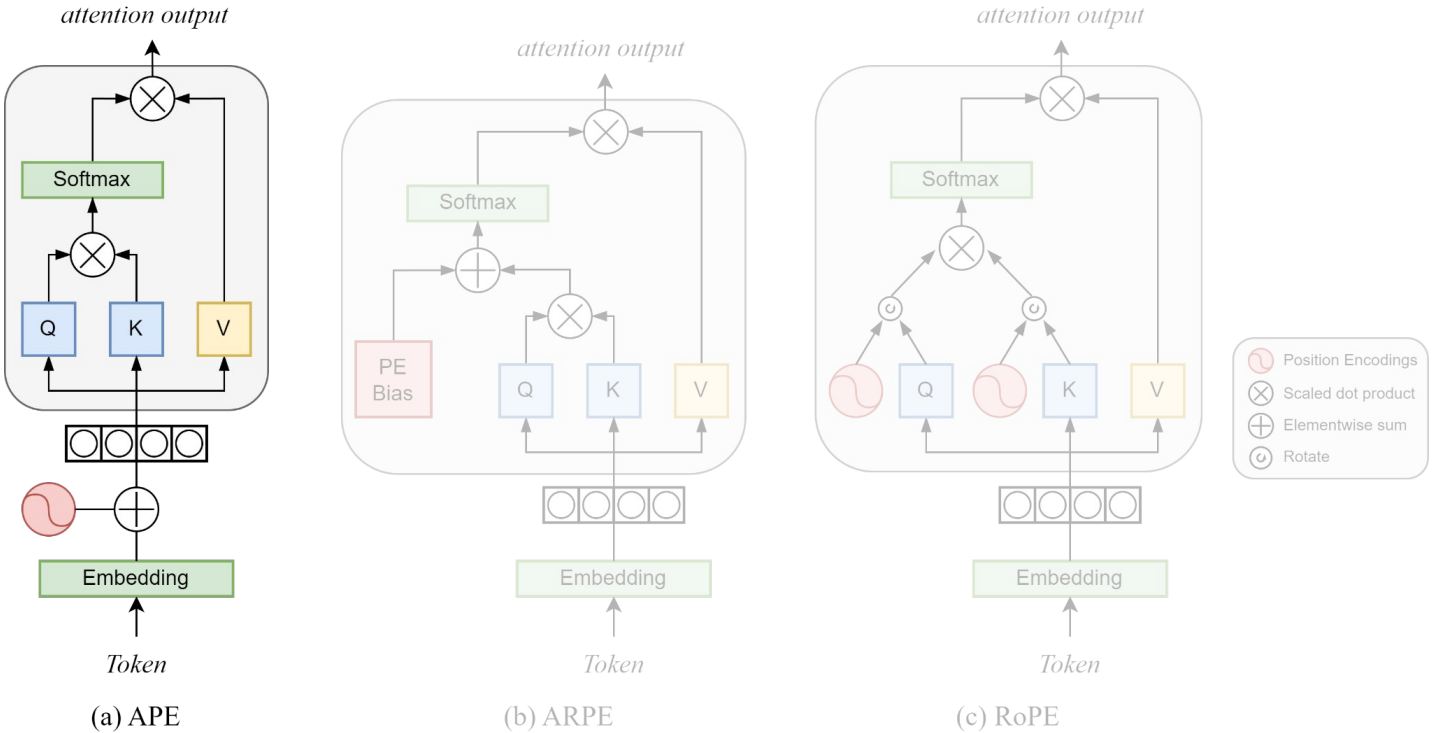
$$PE_{(\text{pos},\ 2i+1)} = \cos\left(\frac{\text{pos}}{10000^{\frac{2i}{d}}}\right)$$

Normalize the range

can keep the value within [-1, 1].

where $2i$, $2i+1$ is the dimension of the position embedding, d is the model hidden dimension, and 10000 is the wavelengths form a geometric progression.

Absolute Position Encoding (APE)



▲ Figure: Overview of the results. [Thesis p.7]

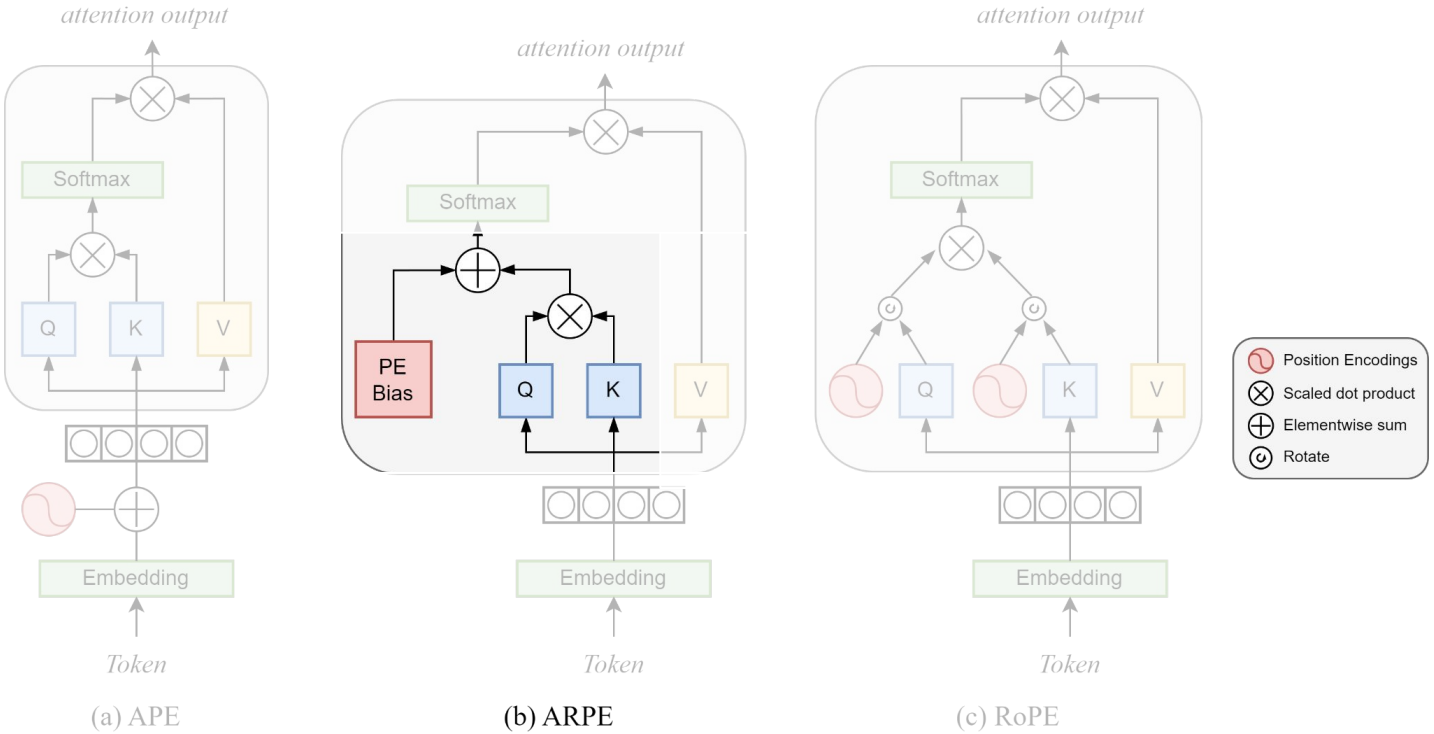
Additive Relative Position Encoding (ARPE)

Overall, ARPE can be formulated as follow equation:

$$\alpha_{\text{RPE}} = \mathbf{q}_i \mathbf{k}_i^\top + b(i,j),$$

- T5 bias: $b(i,j) = r(i,j,K)$, where r is learnable and K is a hyperparameter.
- ALiBi: $b(i,j) = m|j-i|$, where a head-specific slope function.
- FIRE: $b(i,j) = f_\theta\left(\frac{\psi(i-j)}{\psi(\max\{L,i\})}\right)$, where f_θ is a learnable MLP function, $\psi : \mathbb{N} \to \mathbb{R}_+$ is monotonically increasing, and L > 0 is a learnable scalar.

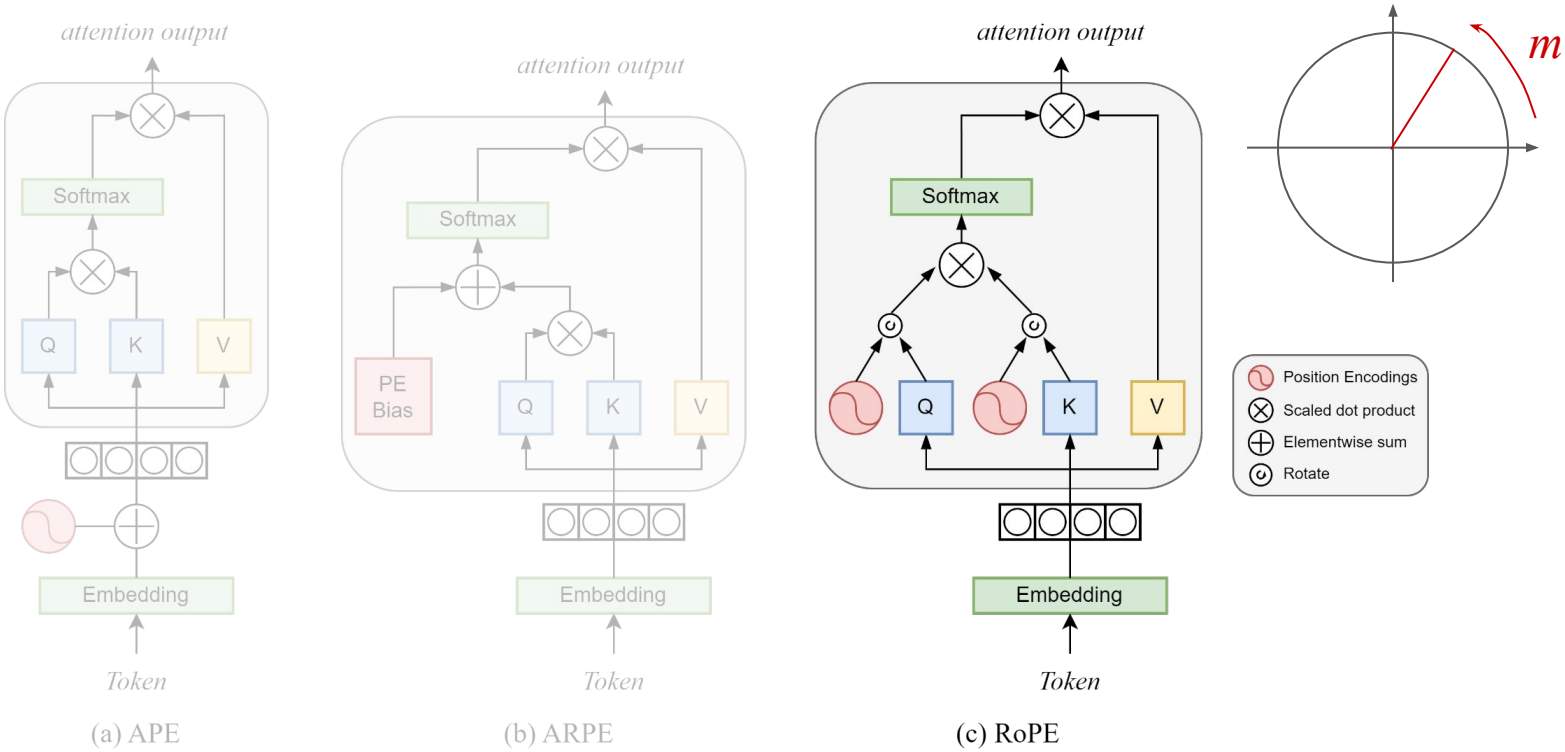
Additive Relative Position Encoding (ARPE)



▲ Figure: Overview of the results. [Thesis p.7]

Rotary Position Encoding (RoPE)

When token at position m



▲ Figure: Overview of the results. [Thesis p.7]

List of PEs

▼ Table: List of PEs and their differences. [Thesis p.8]

PE		Learnable	Operation	Extrapolatable
APE	APE	✓	Additive	✗
	Sinusoidal	✗	Additive	✓
RPE	RPE	✓	Additive	✓
	T5 Bias	✓	Additive	✓
	ALiBi	✗	Additive	✓
	FIRE	✓	Additive	✓
	RoPE	✗	Multiplicative	✓

Shaw et al., 2018

Raffel et al., 2020

Press et al., 2022

Li et al., 2024

Su et al., 2021

In-Context Learning with Meta Learning

$1 + 1 = 2$; $2 + 1 = 3$; $3 + 2 = 5$; $4 + 5 = ?$

In-Context Learning with Meta Learning

$1 + 1 = 2$; $2 + 1 = 3$; $3 + 2 = 5$; $4 + 5 = ?$

$(1 , 1) = 2$

$(2 , 1) = 3$

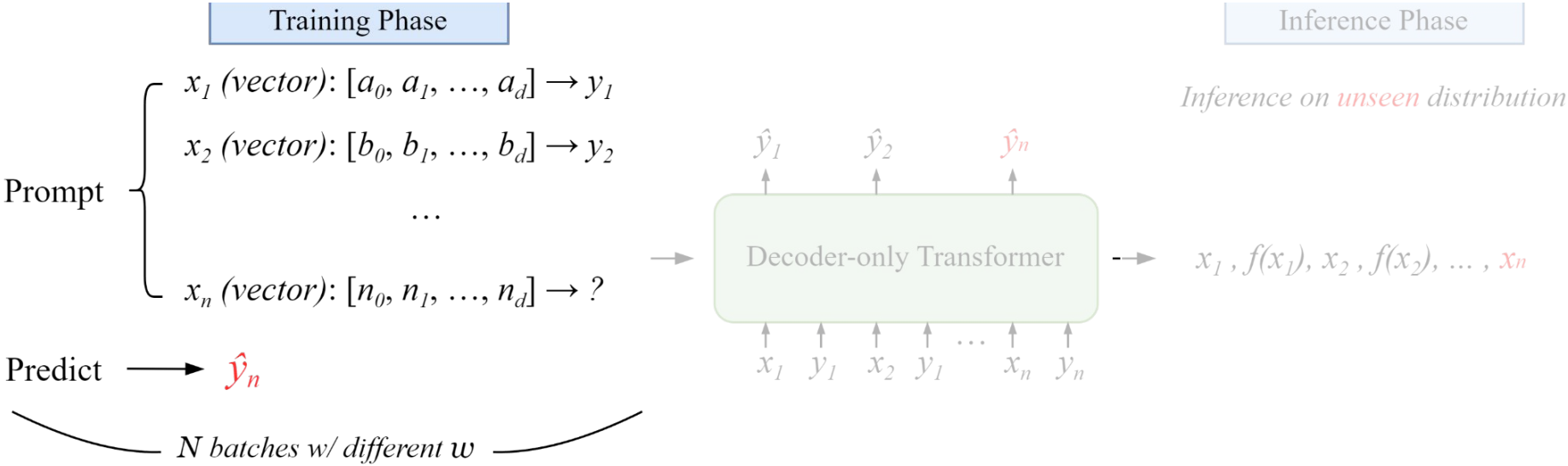
$(3 , 2) = 5$

$(4 , 5) = ?$

We want model

Learning to learn an algorithm!

In-Context Meta Learning



▲ Figure: Overview of In-Context Meta Learning setup [Thesis p.12]

Learning to learn Linear Regression and Boolean functions

(Garg et al., NIPS 2022) (Bhattamishra et al., ICLR 2024)

Summary of Related Work

Length Generalization is hard, but maybe a good Position Encoding can solve the problem.

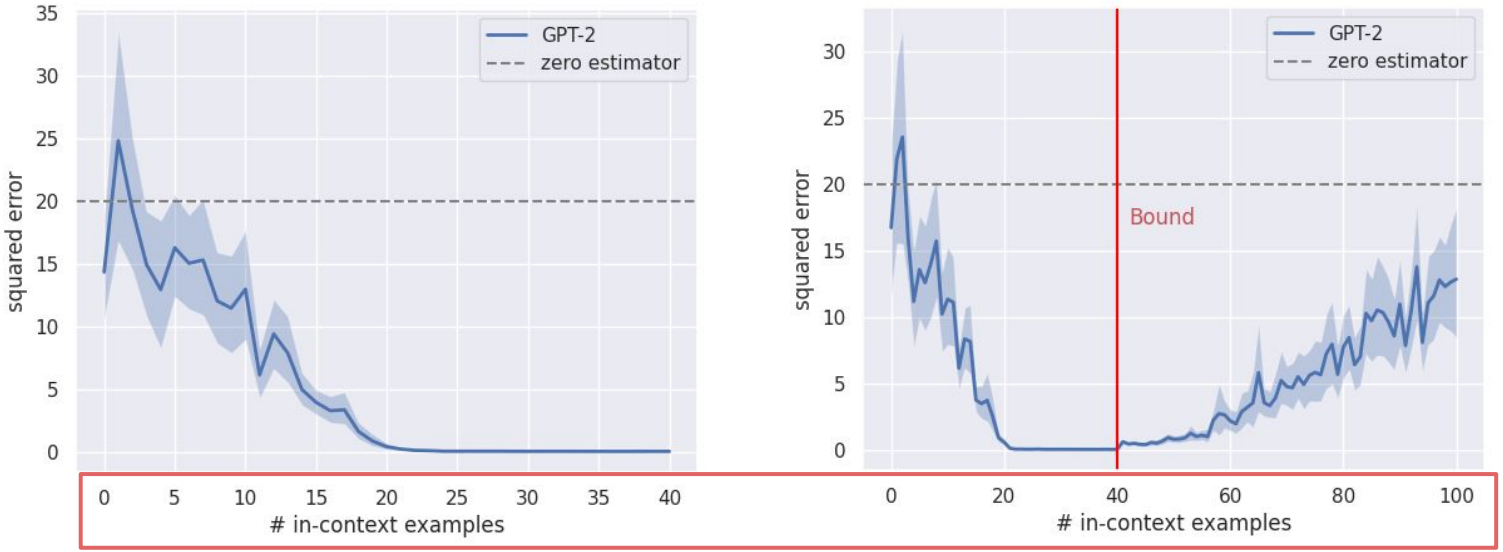
To make sure data without any positional bias, we use In-Context Meta Learning with 2 function classes.

Methodology

- **Preliminary Experiments**
  - In-Context Meta Learning
- **Task Description**
  - Regression functions
  - Boolean functions
- **Other Setups**
  - Hyperparams & Baseline

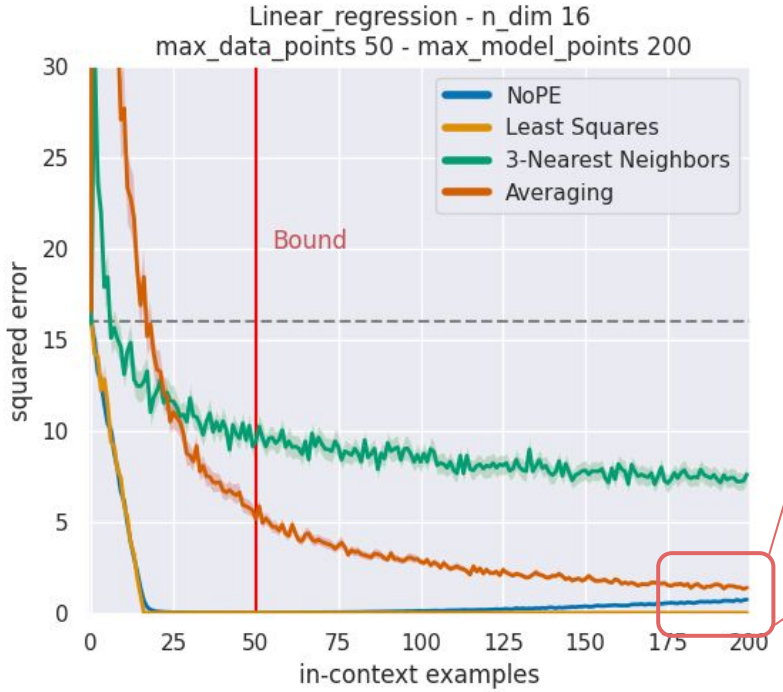
Preliminary: Model fails on OOD Length

We already know that GPT-2 fails because of learnable PE, which is an APE.

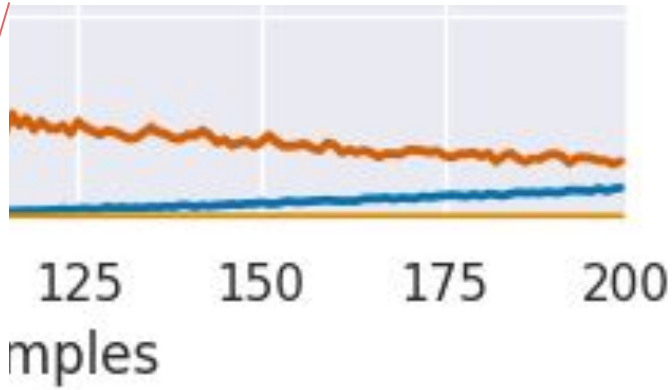


▲ Figure: GPT-2 fails on OOD length. [Thesis p.14]

Preliminary: No Position Encoding (NoPE) also fails



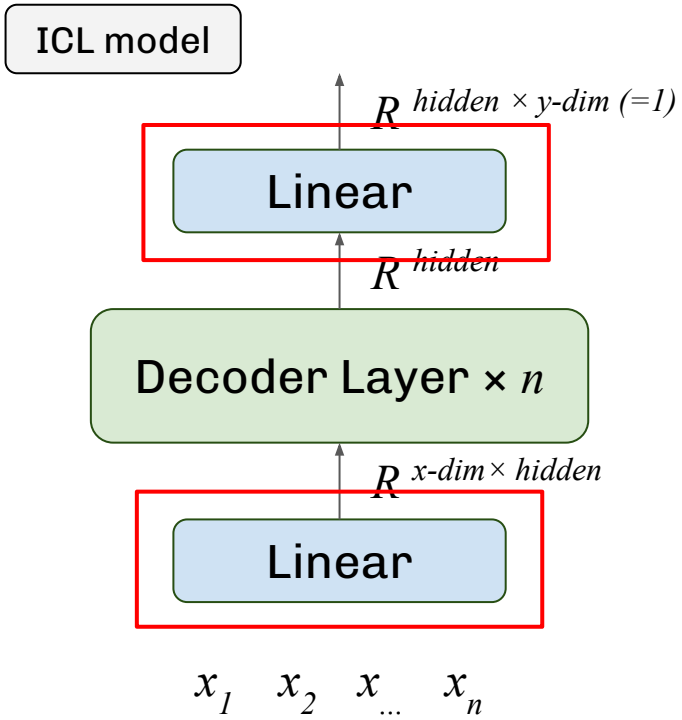
Blue line should be completely flat.

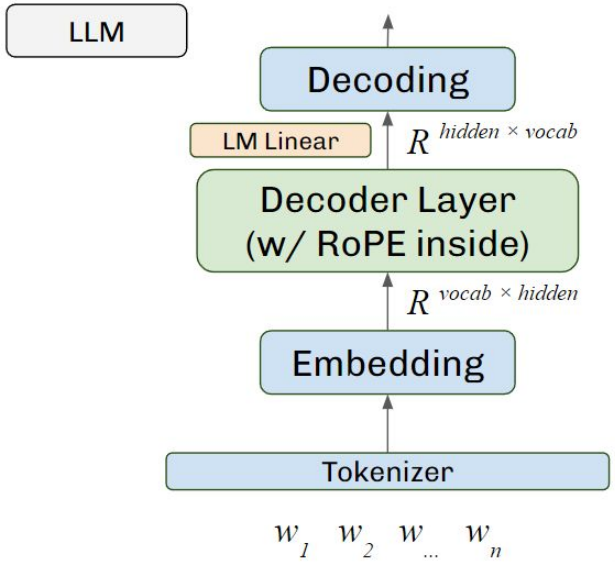


But our data is irrelevant to position.

▲ Figure: NoPE is not perfect. [Thesis p.15]

Model Architecture





▲ Figure: LLMs usually have to tokenize words and generate outputs by decoding.

Task Description

(Garg et al., NIPS 2022) (Bhattamishra et al., ICLR 2024)

▼ Table: List of our function classes.

Task Name		Example x	Example w	Example y
Regression	Linear	$[0.1, -1.0, 0.23]$	$[1.0, -0.1, 0.4]$	0.92
	Sparse Linear	$[0.1, -1.0, 0.23]$	$[0, 0.1, 0]$	-1.0
Boolean	Conjunction	$[0, 1, 1, 0, 0]$	$x_2 \wedge x_3 \wedge \neg x_4$	1
	Disjunction	$[0, 1, 1, 0, 0]$	$x_1 \vee \neg x_2 \vee x_4$	0
	Sparse Disjunction	$[0, 1, 1, 0, 0]$	$x_4 \vee x_5$	0
	CNFs	$[0, 1, 1, 0, 0]$	$(x_1 \wedge x_2) \vee x_4$	0
	DNFs	$[0, 1, 1, 0, 0]$	$x_1 \wedge (\neg x_2 \vee x_5)$	0
	Sparse Majority	$[0, 1, 1, 0, 0]$	$\neg x_1, x_2, x_4$	1
	0-1 Threshold Functions	$[-1, 1, 1, -1, -1]$	$\text{sign}\left(\sum_{i=1}^{d} w_i x_i - b\right)$	3
	Integer Halfspace	$[-1, 1, 1, -1, -1]$	$\text{sign}\left(\sum_{i=1}^{d} w_i x_i - 0.5\right)$	-0.5
	Parity	$[0, 1, 1, 0, 0]$	$x_1 \oplus \neg x_2 \oplus x_3$	1
	Sparse Parity	$[0, 1, 1, 0, 0]$	$x_4 \wedge \neg x_5$	1

10

Hyperparameters Settings

▼ Table: Hyperparameters summary [Thesis p.19]

Data Generation	
input x dimension	16
output y dimension	1
In-Context Examples for Training	50
In-Context Examples for Inference	400
Sampler for Linear Regression	$\mathcal{N}(0, I_d)$
Sampler for Boolean Functions	$\{0, 1\}$

Training Arguments	
Batch Size	64
Training Steps	50,000
Weight Decay	0.0
Dropout	0.0
Optimizer	AdamW
Learning Rate (LR)	3e-4
LR Warmup Ratio	0.03
LR Scheduler	Linear

Baseline for Linear Regression

Given a set of data points, or prompt, $P = (x_1, y_1, \ldots, x_n, y_n, x_{\text{query}})$, we define that each baseline model M should solve for $\hat{w}$ and predict the output $\hat{y}$ with the target input x_{query}, i.e., $M(P) = \hat{w}x_{\text{query}}$. Note that $x \in \mathbb{R}^d$ and $y \in \mathbb{R}$.

Least squares: Minimize $\|y - \hat{w}x\|_2$. This estimator needs d samples to solve $\hat{w}$.

Averaging estimator: This estimator computes $\hat{w} = \frac{1}{n}\sum_{i=1}^{n} x_i y_i$.

Nearest neighbors: This estimator computes $\hat{y} = \frac{1}{n}\sum_{i \in S} y_i$, where S is the set of indices of n nearest neighbors. We set $n = 3$.

Baseline for Boolean Functions

Null classifier: This estimator will only predict 0 for each input.

Averaging estimator: Same estimator as in Linear Regression.

Nearest neighbors: Same estimator as in Linear Regression. For most tasks, we set $n = 3$, except for Conjunction, Disjunction, and Majority tasks, where we set $n = 1$.

Summary of Methodology

We use 8 PEs NoPE | ALiBi | FIRE | RoPE | PI | Dynamic NTK | YaRN | Dynamic YaRN

And 12 function classes Linear Regression | Sparse Linear Regression | Conjunction | Disjunction | Sparse Disjunction | CNFs | DNFs | Sparse Majoriy | 0-1 Threshold Functions | Integer Halfspace | Parity | Sparse Parity

To see how PEs affect Length Generalization

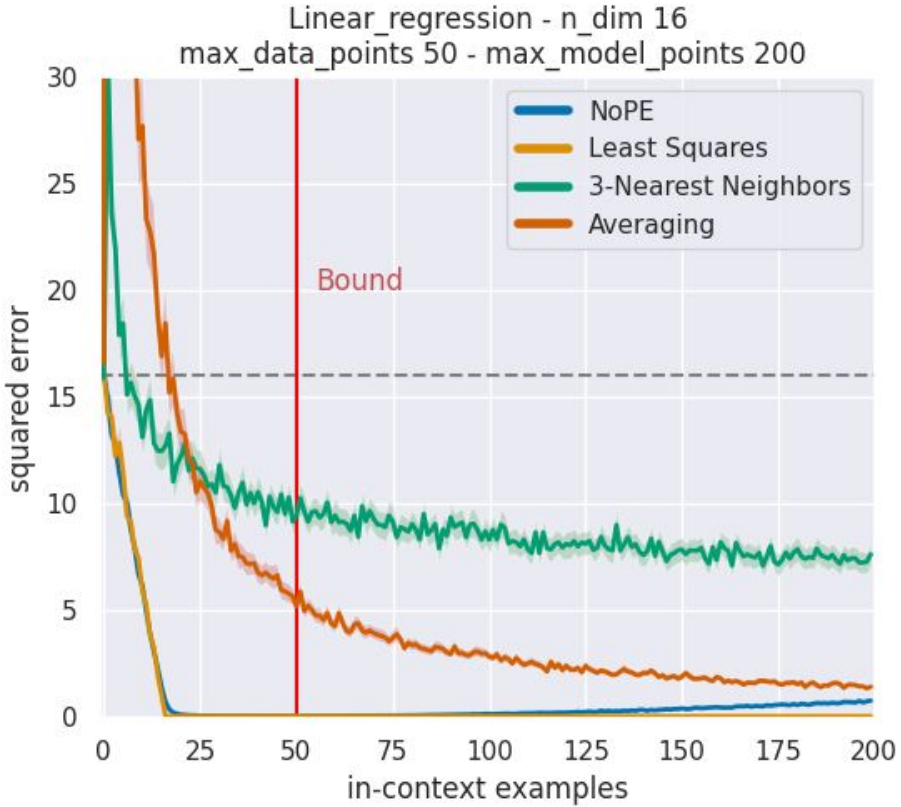
while ensuring that no other variables influence our experiments.

Experiments

- **Easy to Memorize**
- **Demonstrate ICL capability**
- **Hard to solve**

We categorize the task characteristics into 3 types:

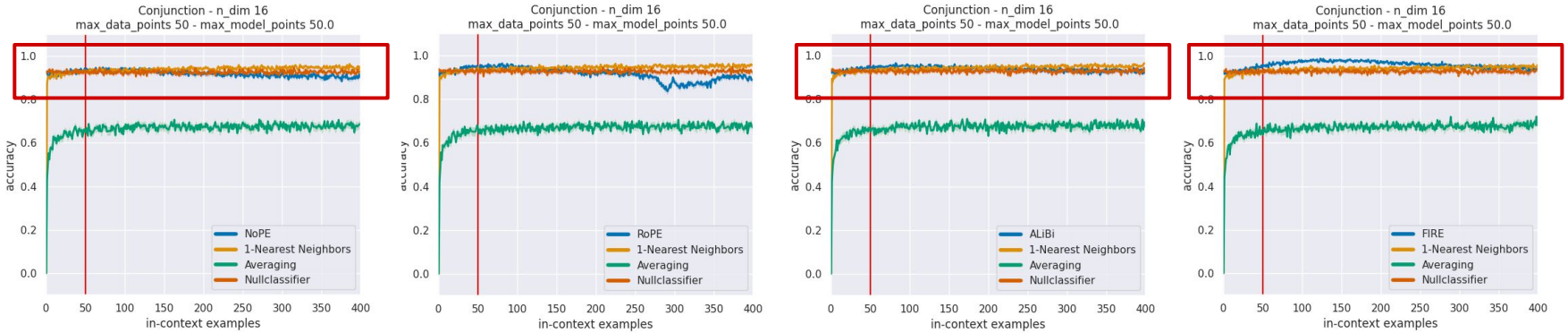
- **Easy to Memorize**: Model can answer correctly **without needing in-context examples**.
- **Demonstrate ICL capability**: Model can **learn algorithm** from training tasks and predict answers when facing unseen data.
- **Hard to solve**: Model is unable to utilize In-Context Meta Learning to learn and answer.



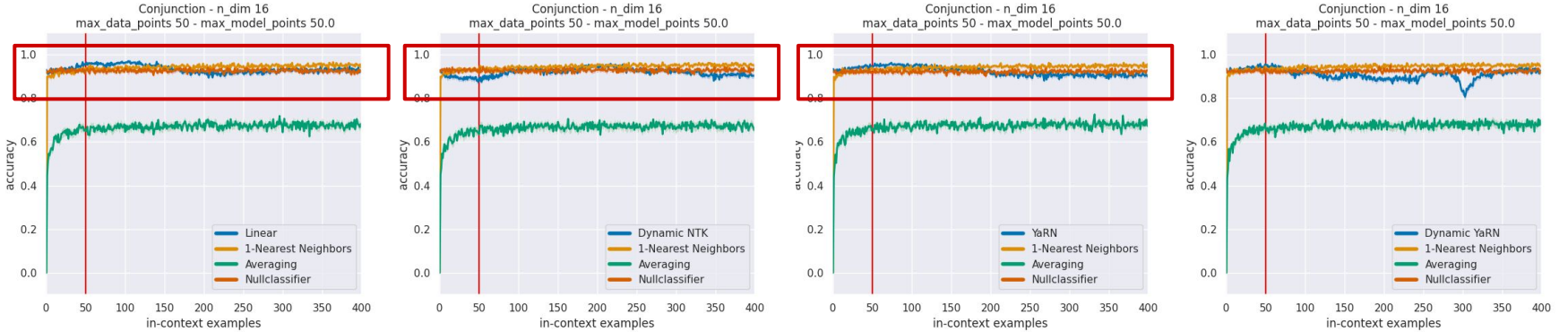
PE
(We will focus on it)

Baseline

Easy to Memorize: Conjunction

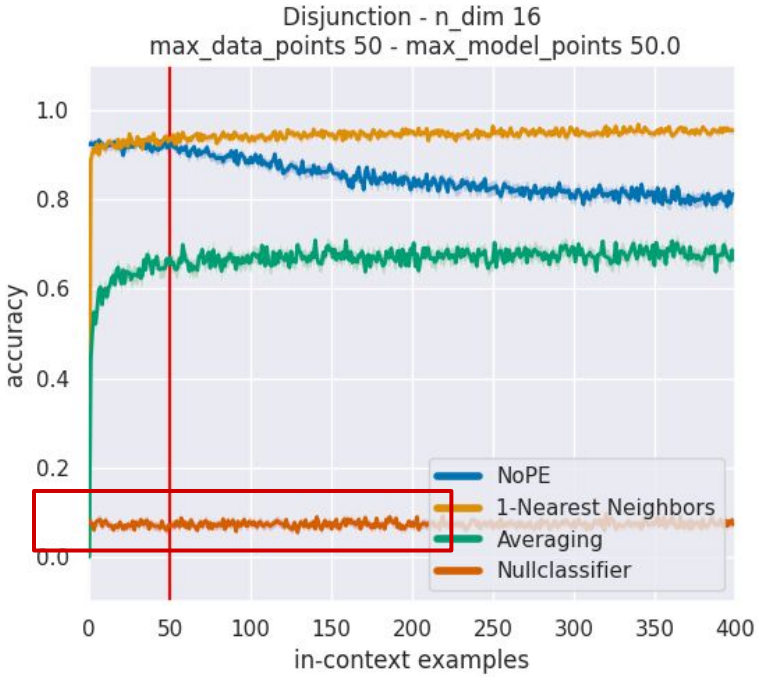


(a) NoPE (b) RoPE (c) ALiBi (d) FIRE

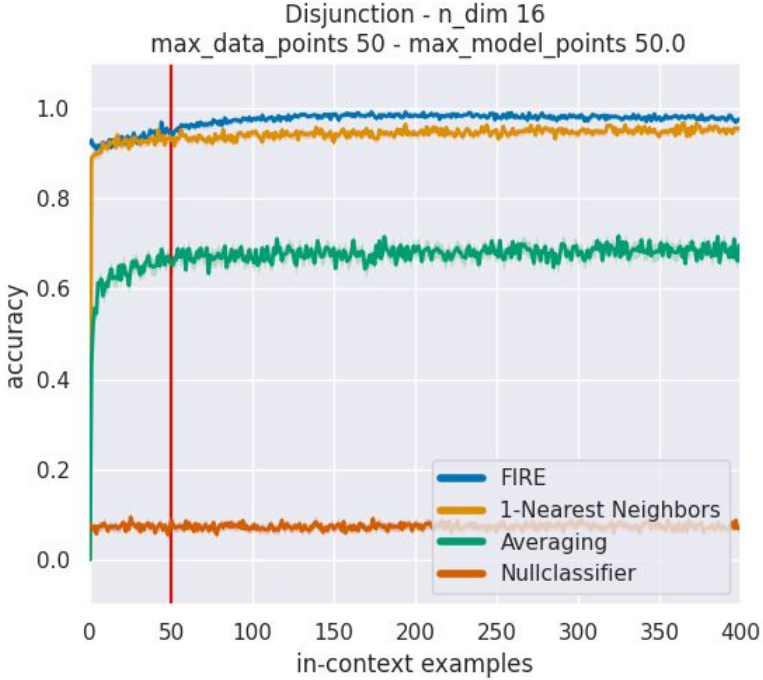


(e) PI (Linear) (f) Dynamic NTK (g) YaRN (h) Dynamic YaRN

Easy to Memorize: Disjunction

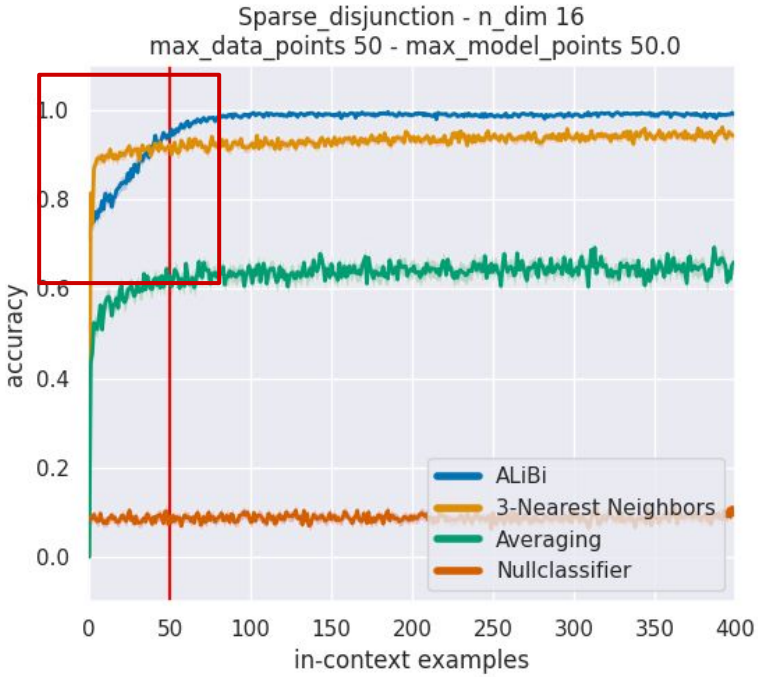


(a) NoPE

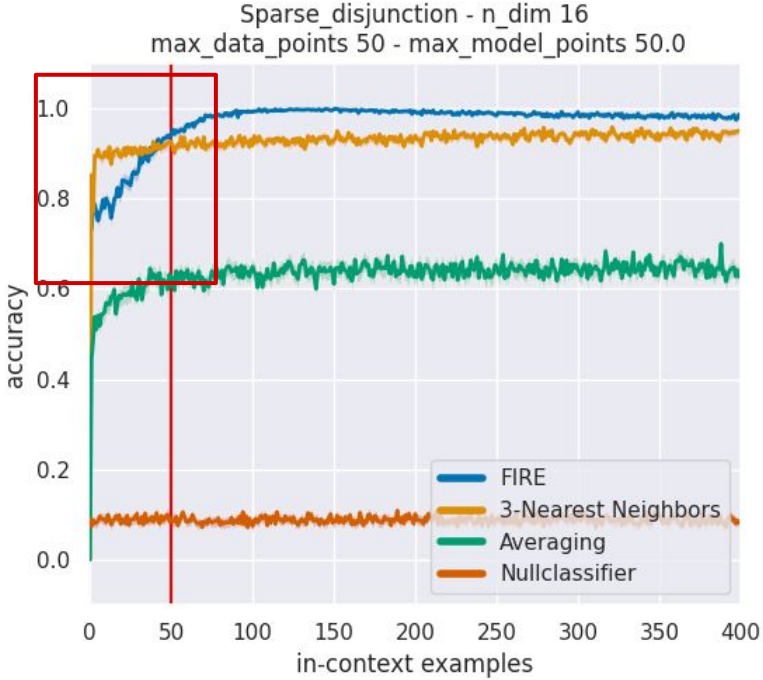


(d) FIRE

Easy to Memorize: Sparse Disjunction

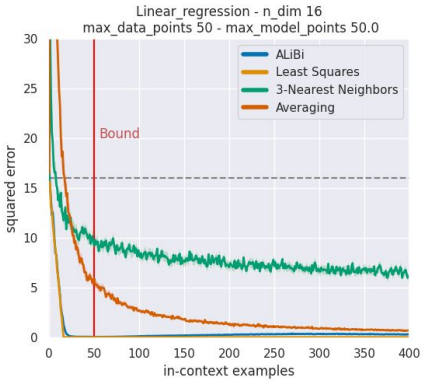


(c) ALiBi

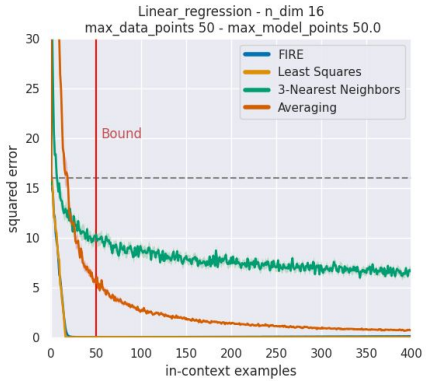


(d) FIRE

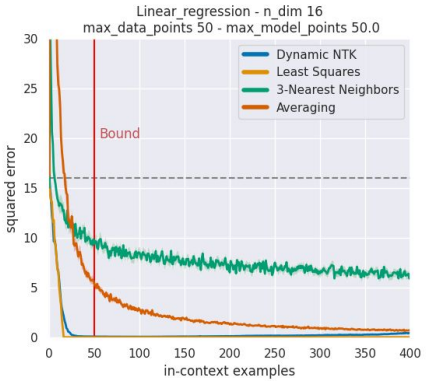
Demonstrate ICL Capability: Linear Regression



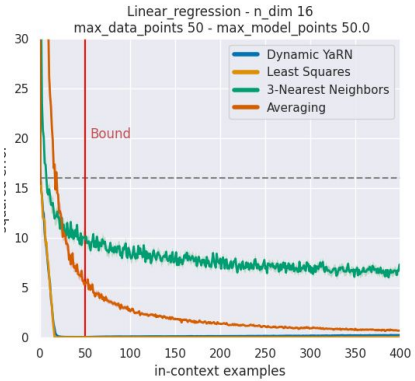
(c) ALiBi



(d) FIRE

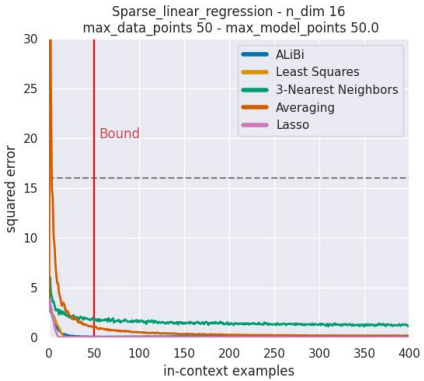


(f) Dynamic NTK

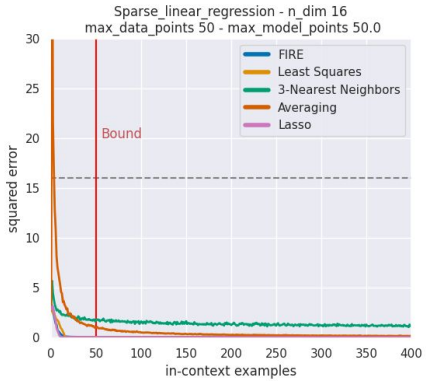


(h) Dynamic YaRN

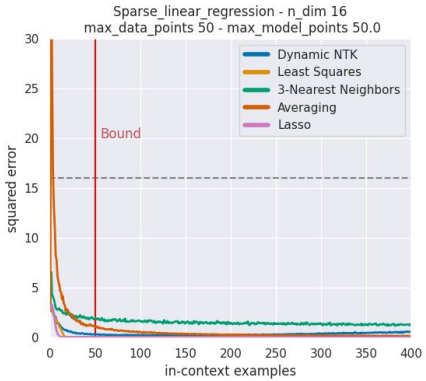
Demonstrate ICL Capability: Sparse Linear Regression



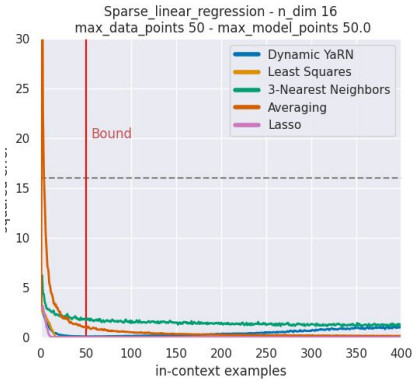
(c) ALiBi



(d) FIRE

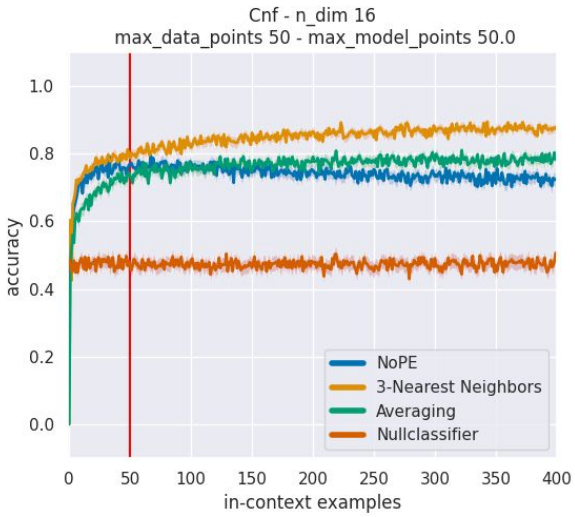


(f) Dynamic NTK

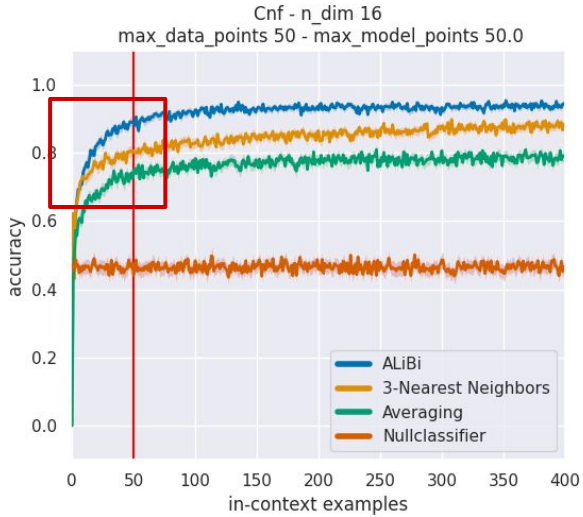


(h) Dynamic YaRN

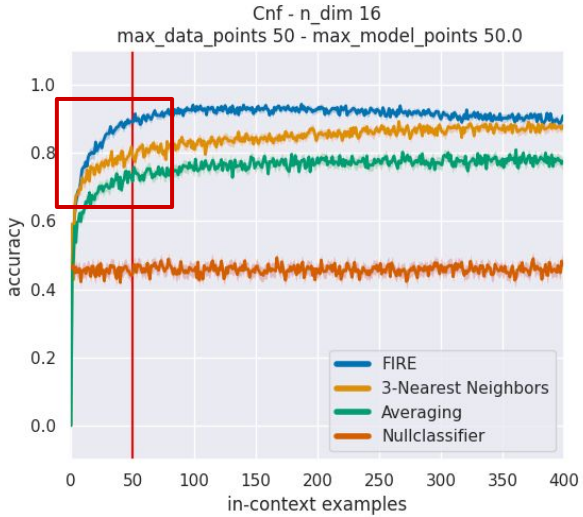
Demonstrate ICL Capability: CNFs



(a) NoPE

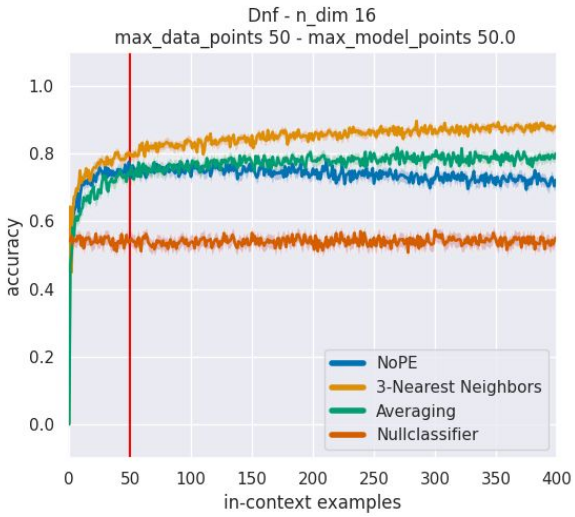


(c) ALiBi

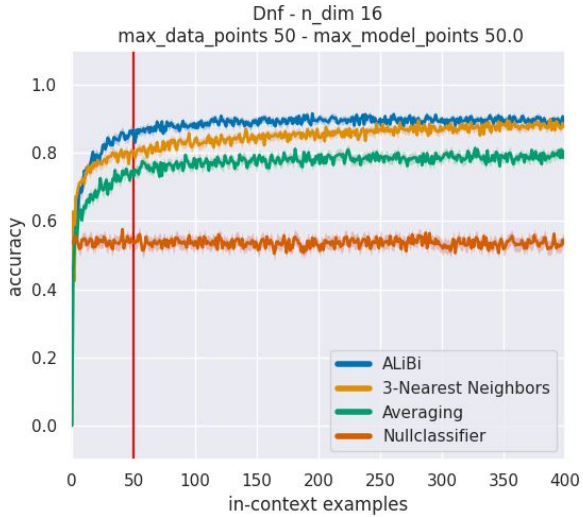


(d) FIRE

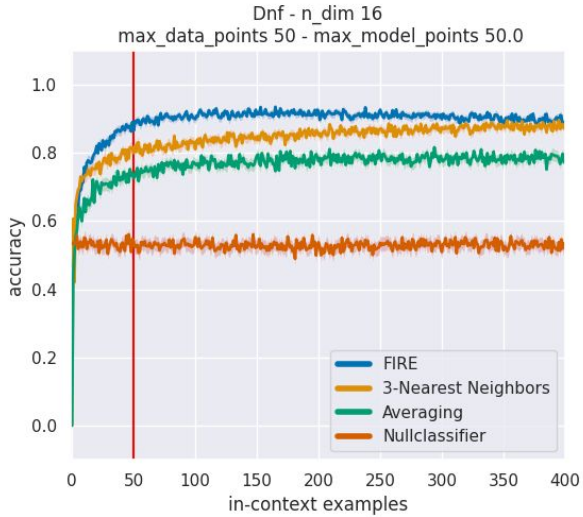
Demonstrate ICL Capability: DNFs



(a) NoPE

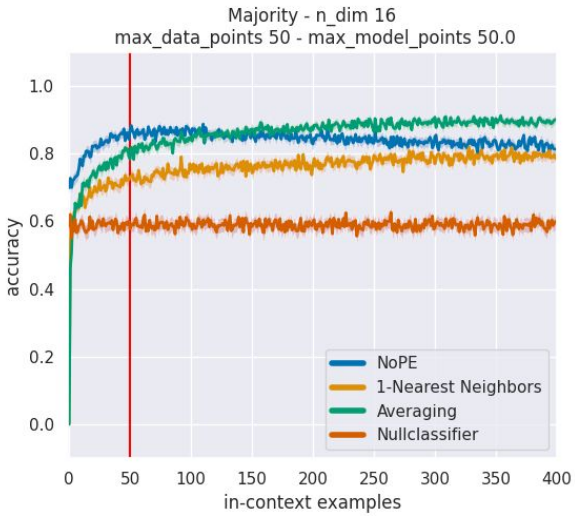


(c) ALiBi

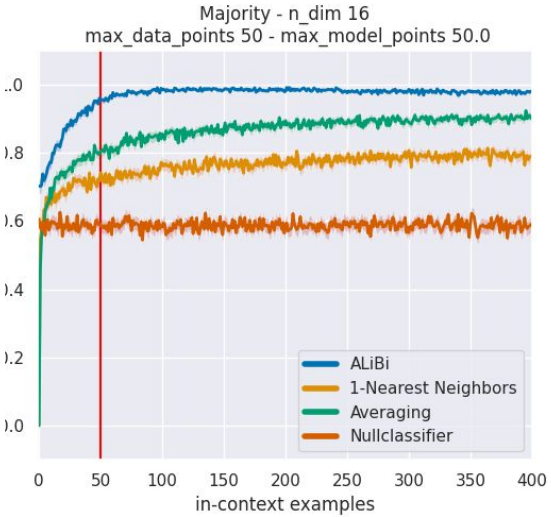


(d) FIRE

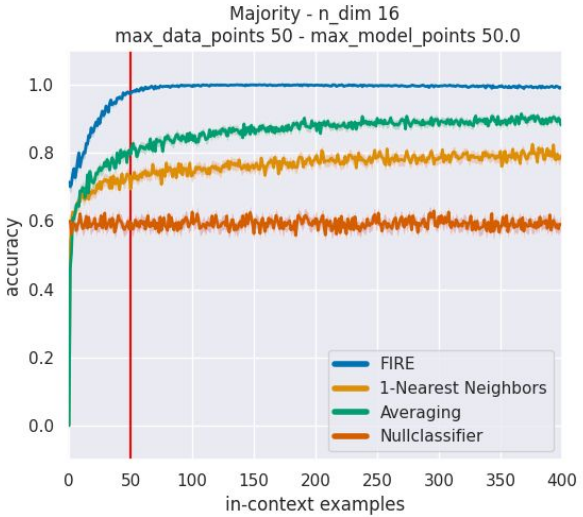
Demonstrate ICL Capability: Sparse Majority



(a) NoPE

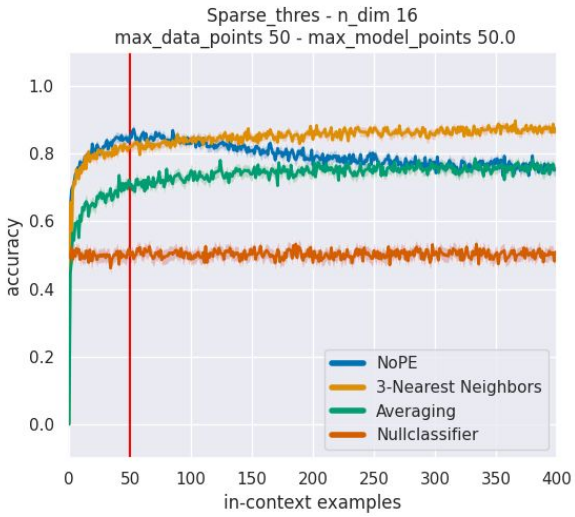


(c) ALiBi

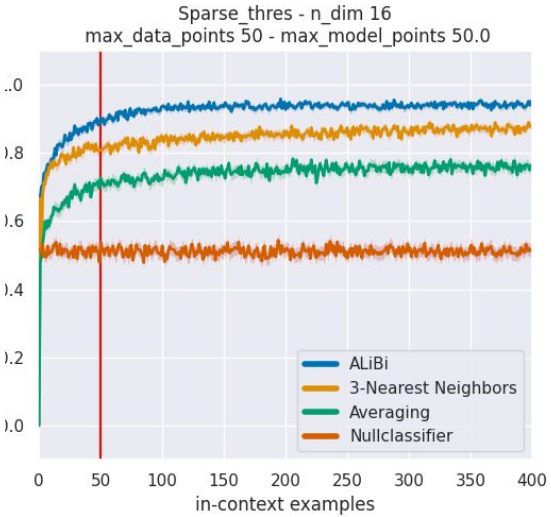


(d) FIRE

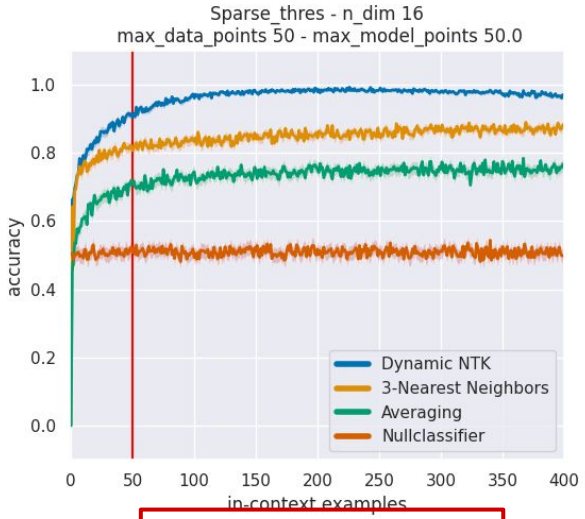
Demonstrate ICL Capability: 0-1 Threshold Functions



(a) NoPE

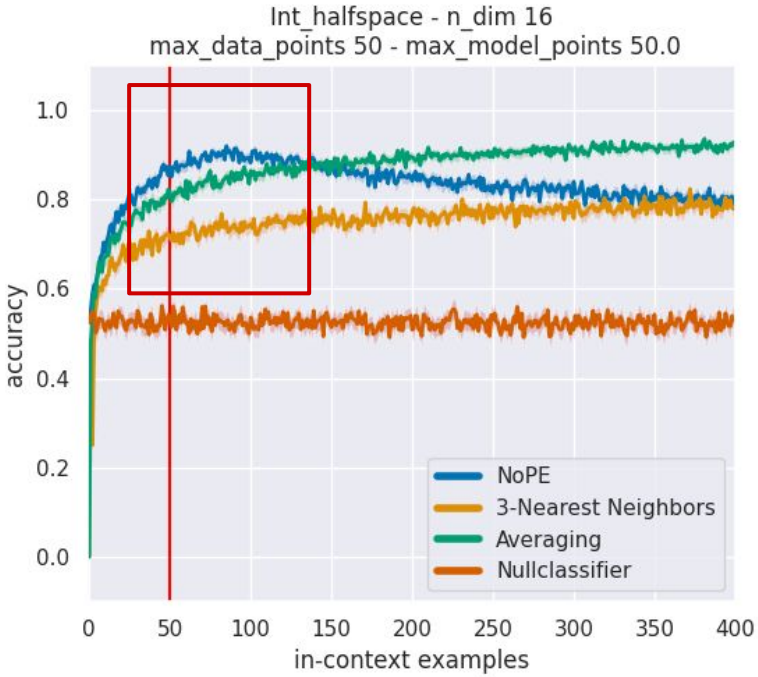


(c) ALiBi

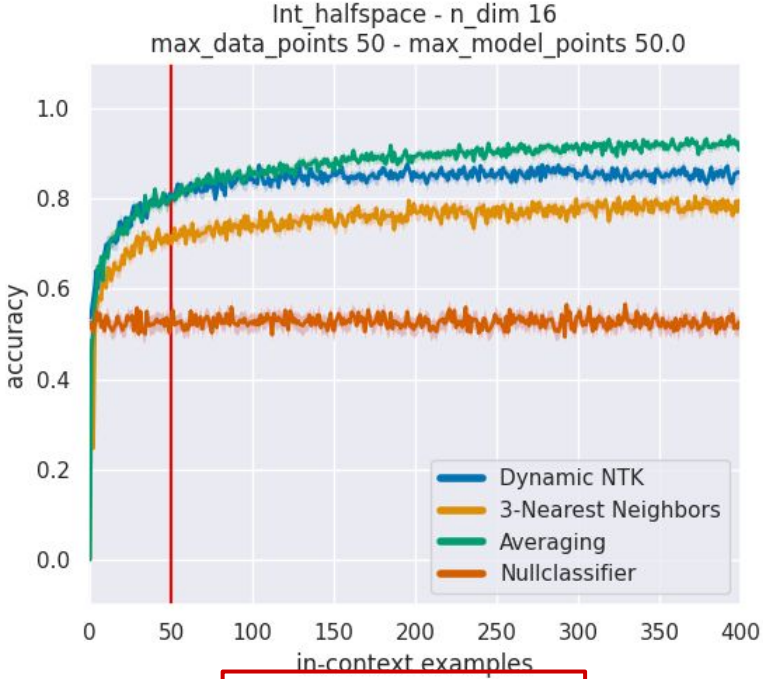


(f) Dynamic NTK

Demonstrate ICL Capability: Integer Halfspace

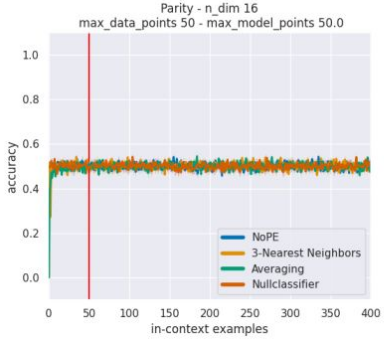


(a) NoPE

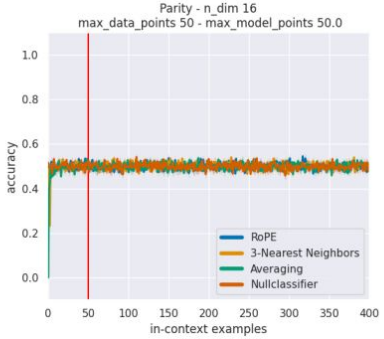


(f) Dynamic NTK

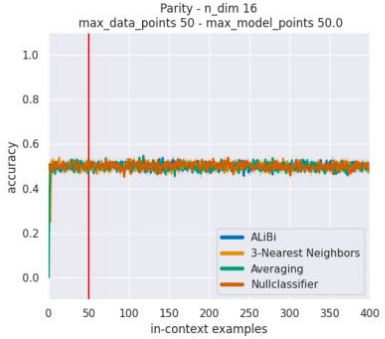
Hard to solve: Parity



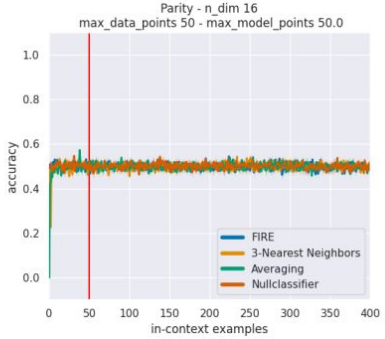
(a) NoPE



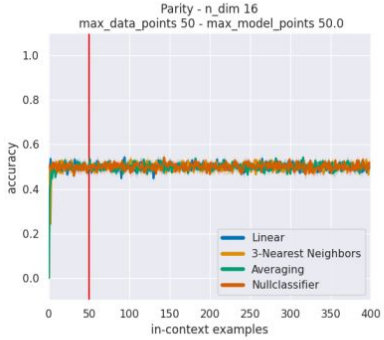
(b) RoPE



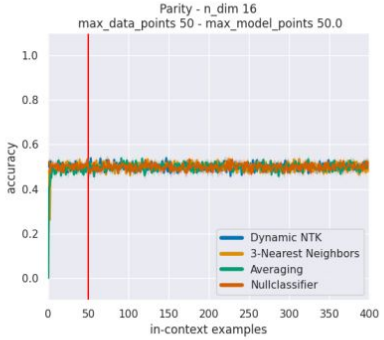
(c) ALiBi



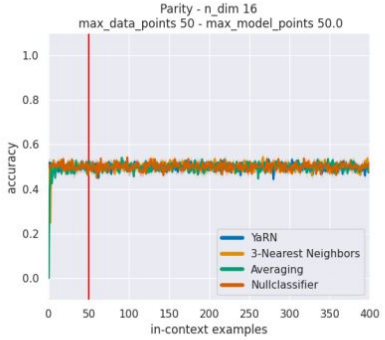
(d) FIRE



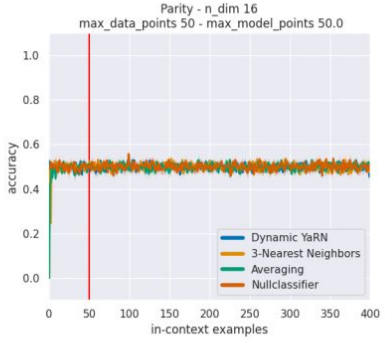
(e) PI



(f) Dynamic NTK

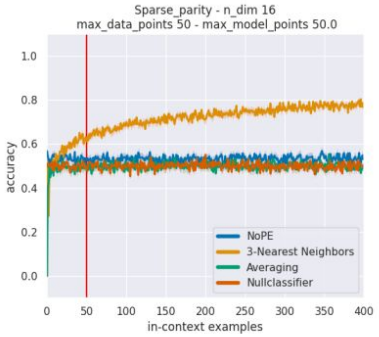


(g) YaRN

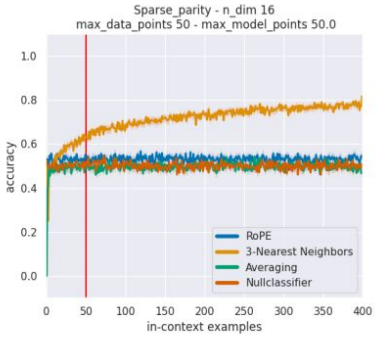


(h) Dynamic YaRN

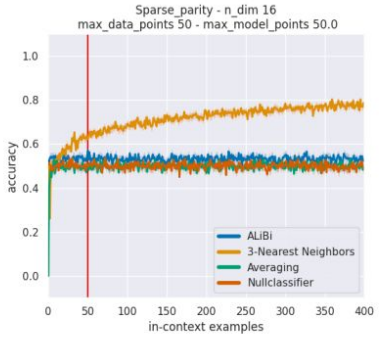
Hard to solve: Sparse Parity



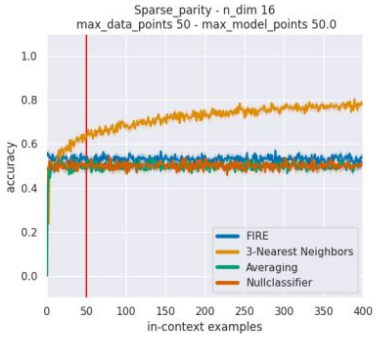
(a) NoPE



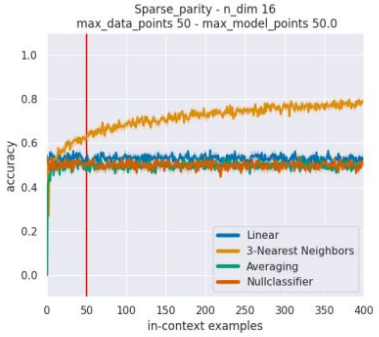
(b) RoPE



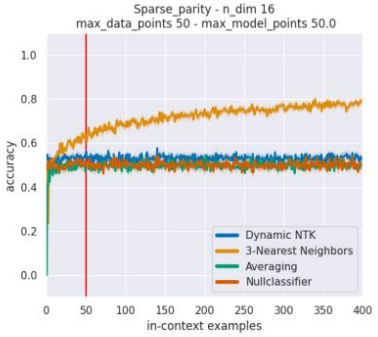
(c) ALiBi



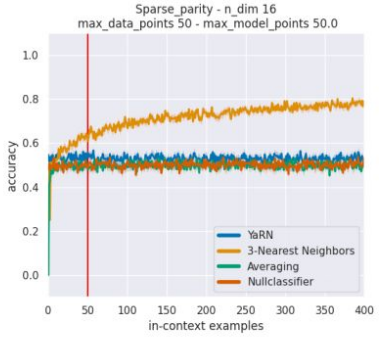
(d) FIRE



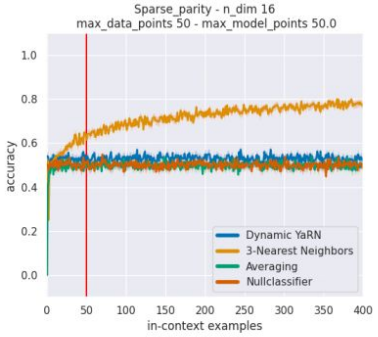
(e) PI



(f) Dynamic NTK



(g) YaRN



(h) Dynamic YaRN

Summary of Experiments

 ALiBi & FIRE seem good for Length Generalizatoin.

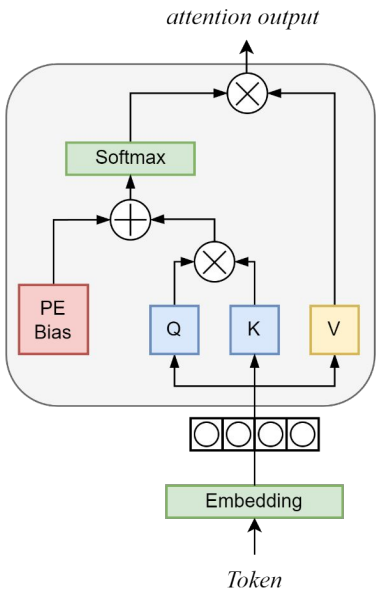
Dynamic NTK & Dynamic YaRN are also not bad.

Do not use NoPE. RoPE, PI, and YaRN need fine-tune.

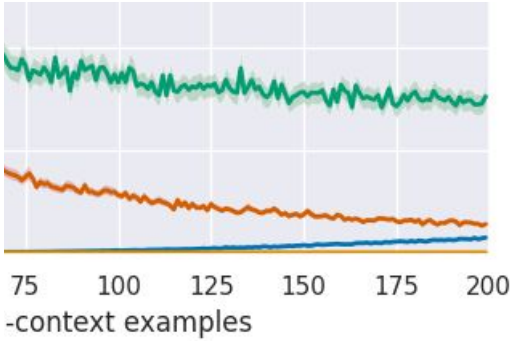
Analysis

- **Model Scale**
- **Recency Bias**
- **Inductive Bias**
- **Serial-Position Effect**

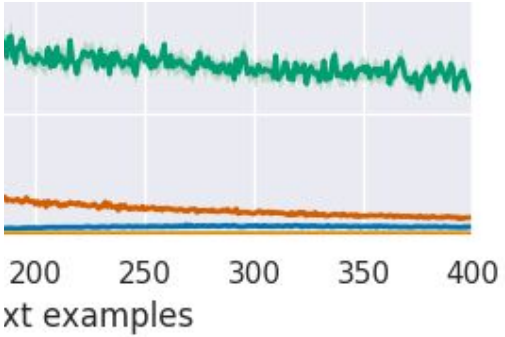
Is ARPE (ALiBi & FIRE) really that good?



(b) ARPE



(a) NoPE

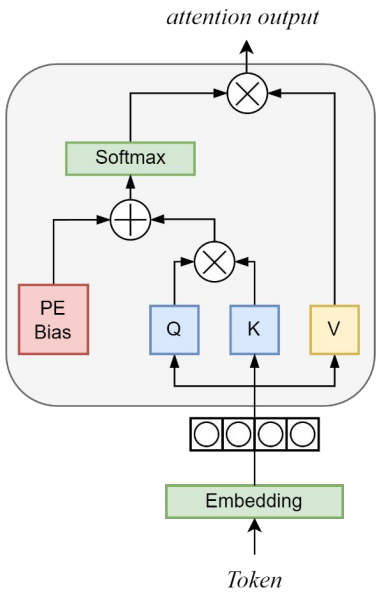


(c) ALiBi

Does NoPE have bias inside attention weight?

Or these results only happen on small model?

Is ARPE (ALiBi & FIRE) really that good?



(b) ARPE

Answer questions from a book

grass is green, sky is blue, … , sun is yellow

sun is yellow, grass is green, sky is blue, …

blue 34a7-02wz grass is green, sky is

grass is green, sky is blue, … , sun is yellow

Remember the uid with lots of noise

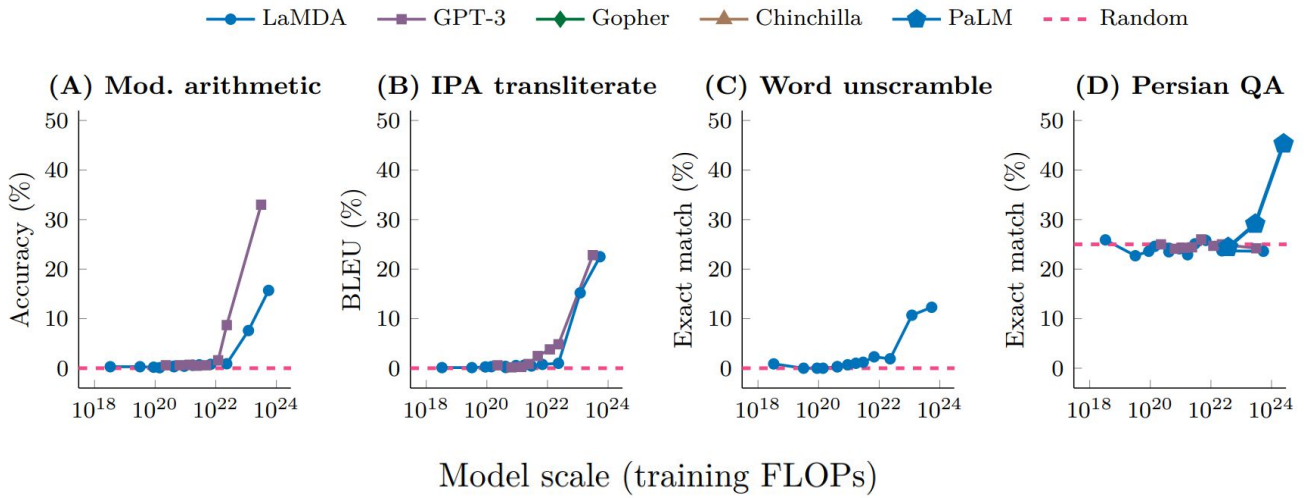
We want to data attack like these methods.

Does increasing model size enhance performance?

Background

Emergent Abilities: (Wei et al., 2022)

An ability is emergent if it is not present in smaller models but is present in larger models.

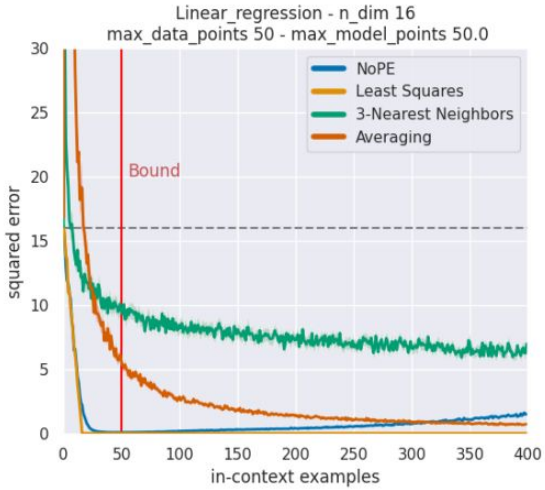


▲ Figure from Wei et al., 2022 : When model scaling up, it shows the ability to perform a task.

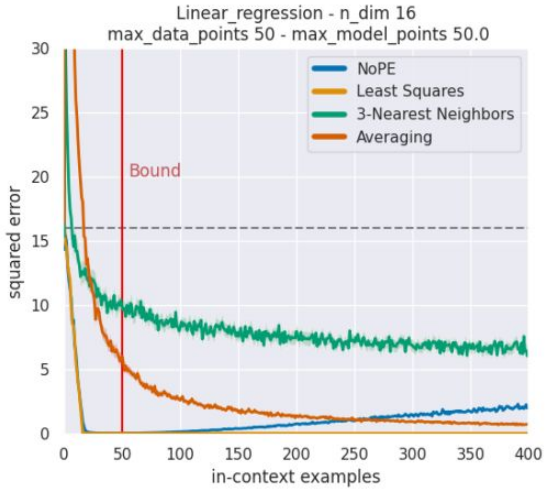
Does increasing model size enhance performance?

Results & Discussion

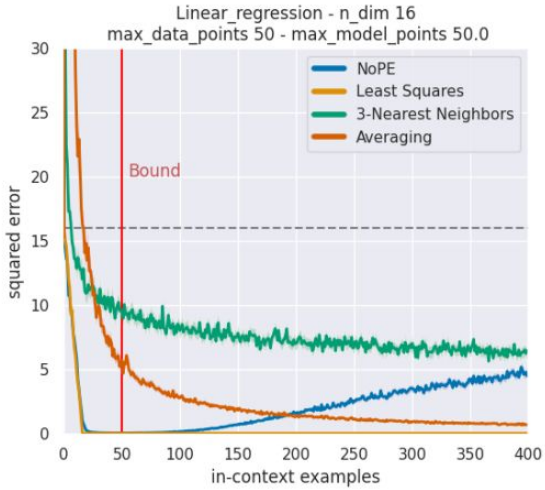
Linear Regression



(a) 12M



(b) 25M



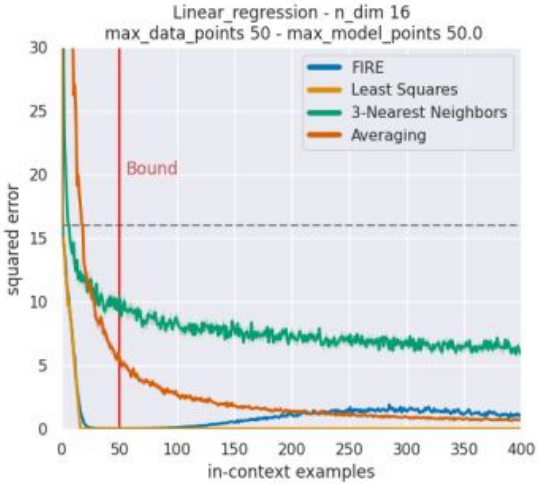
(c) 84M

▲ Figure: NoPE is geting worse when model size is larger.

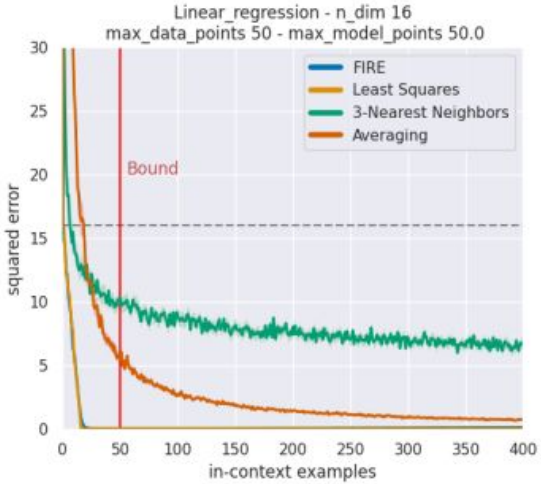
Does increasing model size enhance performance?

Results & Discussion

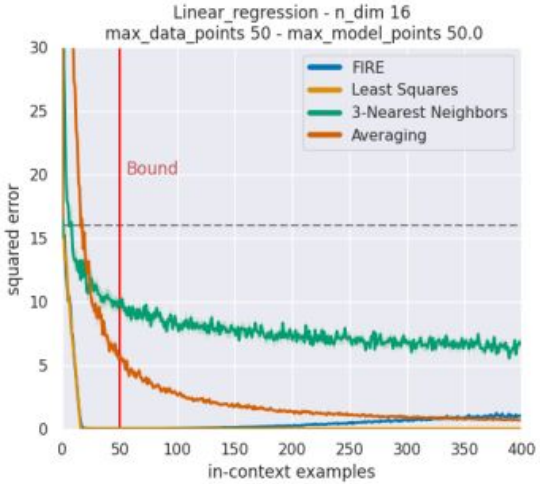
Linear Regression



(a) 12M



(b) 25M



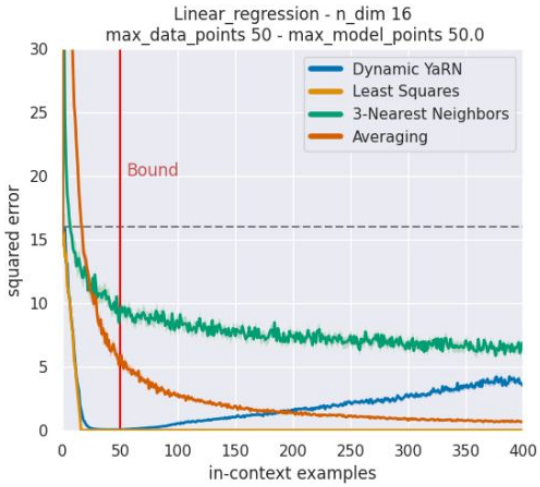
(c) 84M

▲ Figure: FIRE has weaker length extrapolation abilities at 84M model.

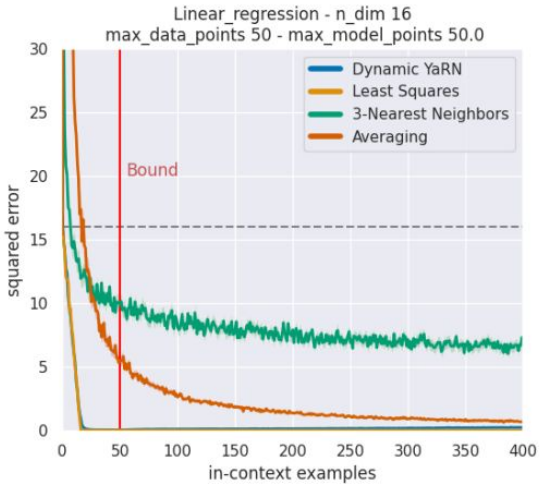
Does increasing model size enhance performance?

Results & Discussion

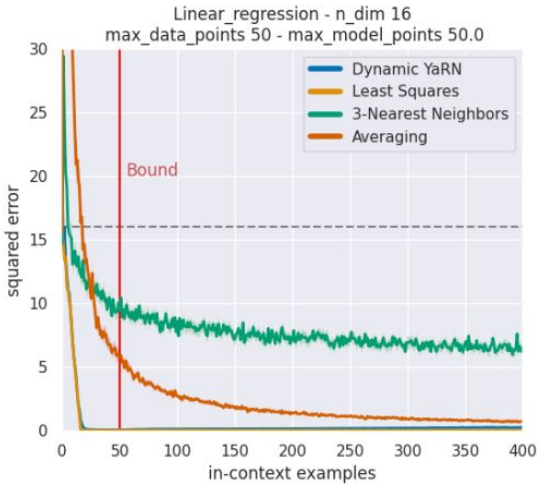
Linear Regression



(a) 12M



(b) 25M



(c) 84M

▲ Figure: Dynamic YaRN has poor performance at 12M model, but shows **consistency** at larger models.

Does increasing model size enhance performance?

Conclusion

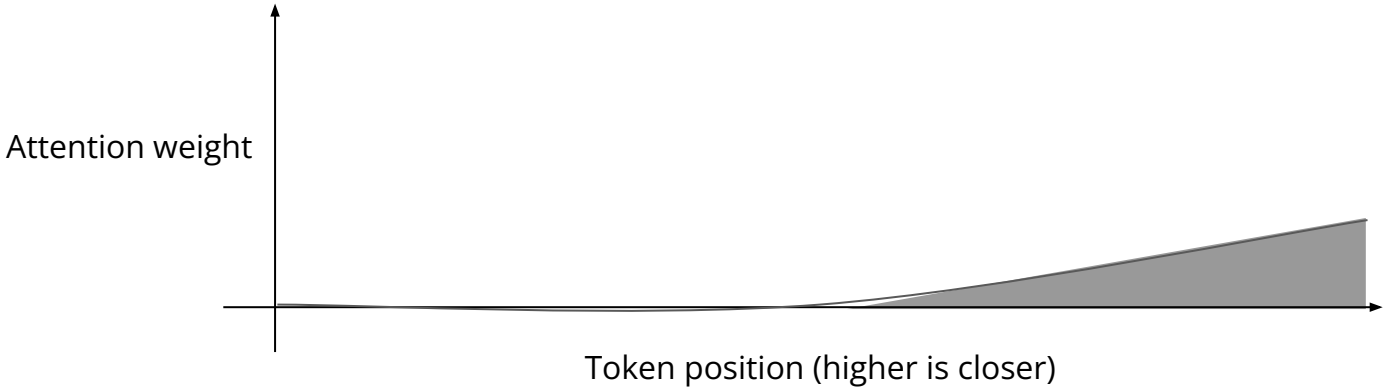
We can only say when the model is **sufficiently large**, its performance tends to stabilize.

Since we are using the model size 25M for our main experiment, we believe that the main experimental results remains valuable for reference.

Can recency bias explain why transformer fails?

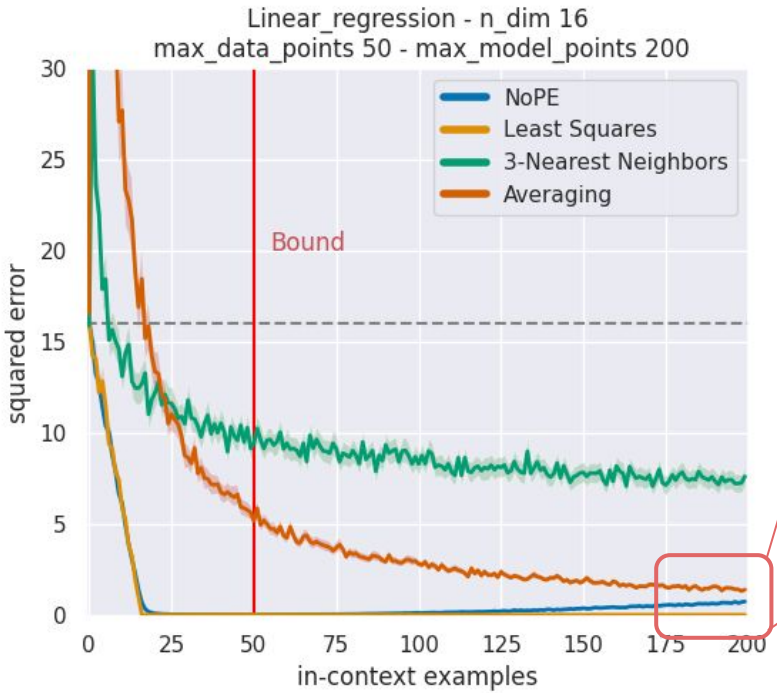
Background

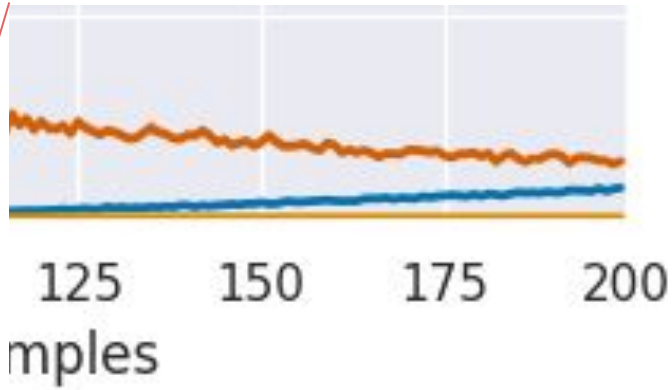
Recency bias is a well-known issue in sequence models, where the model tends to be **influenced more by recent states**.



▲ Figure: Decoder-only Transformer may also suffer from recency bias by attending to recent tokens.

Can recency bias explain why transformer fails?





Recap (p.30)

But our data is irrelevant to position.

▲ Figure: NoPE is not perfect. [Thesis p.14]

Can recency bias explain why transformer fails?

Results & Discussion

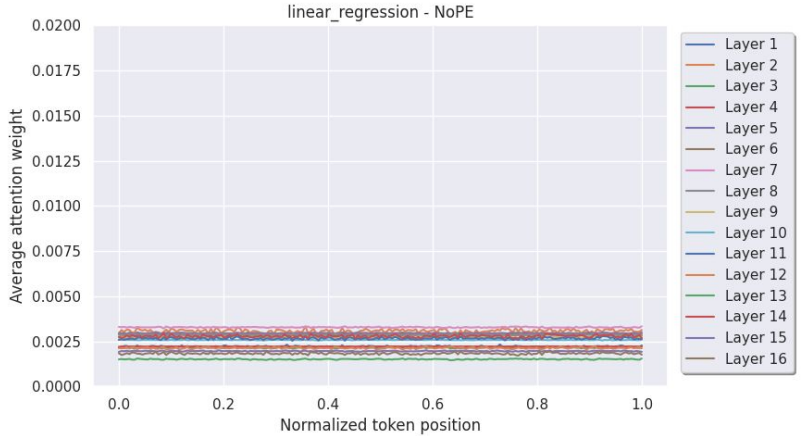
$$\alpha_{\text{RPE}} = \mathbf{q}_i \mathbf{k}_i^\top + b(i,j), \quad \text{ALiBi: } b(i,j) = m|j-i|,$$

Can recency bias explain why transformer fails?

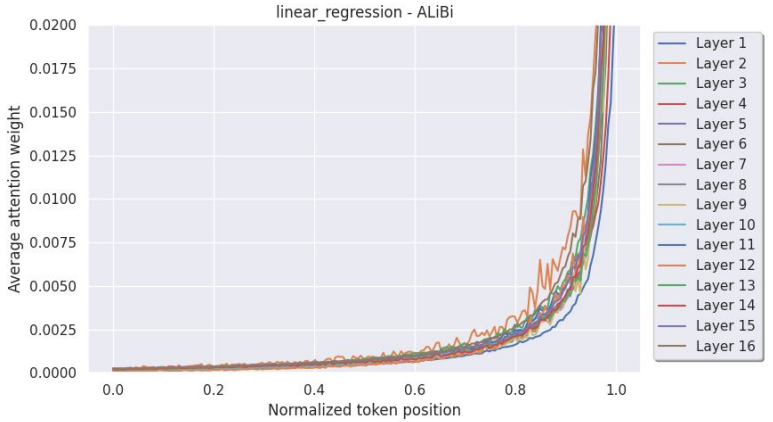
Results & Discussion

Linear Regression

Average attention weight for each layer:



(a) NoPE



(b) ALiBi

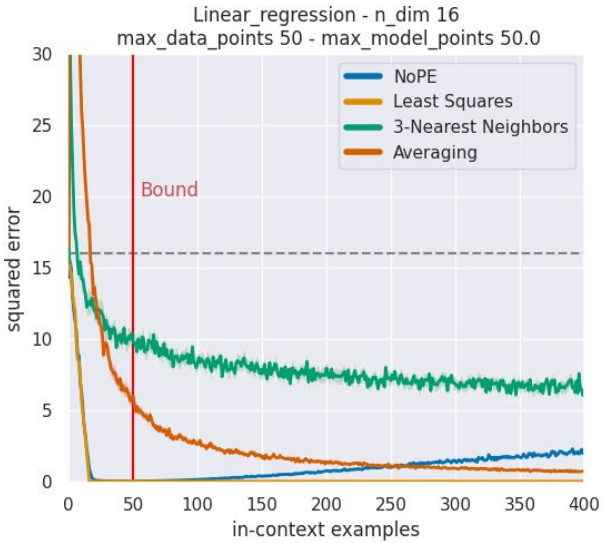
▲ Figure: Normalized token position means 0.0 is the farthest token and 1.0 is the most recent token.

Recap: Who wins?

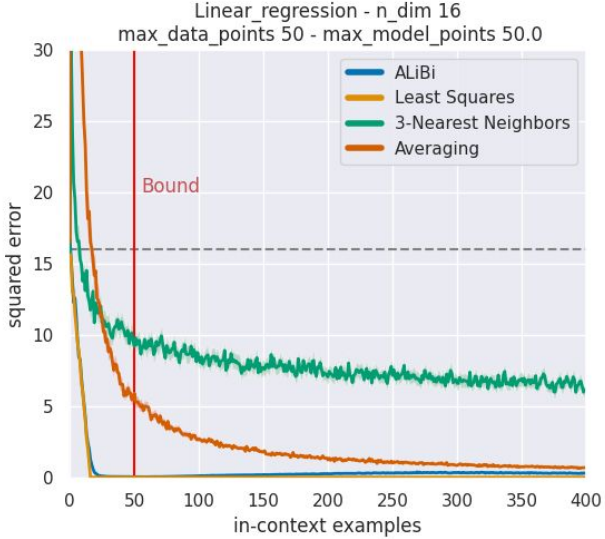
Linear Regression

Results & Discussion

WIN



(a) NoPE



(b) ALiBi

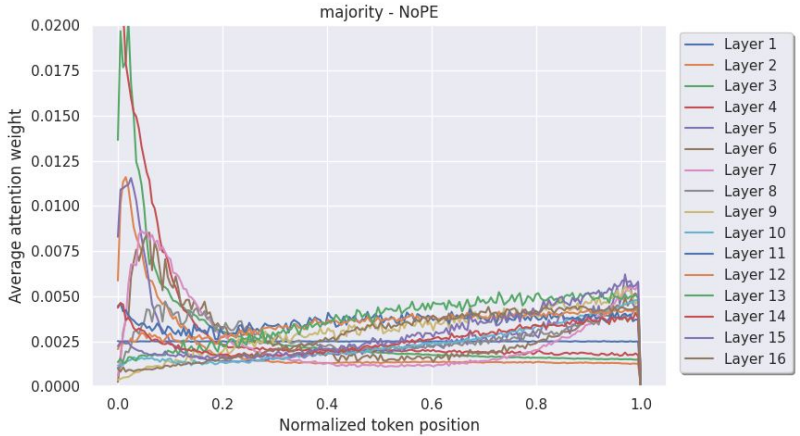
▲ Figure: Actually ALiBi has better performance on Linear Regression.

Can recency bias explain why transformer fails?

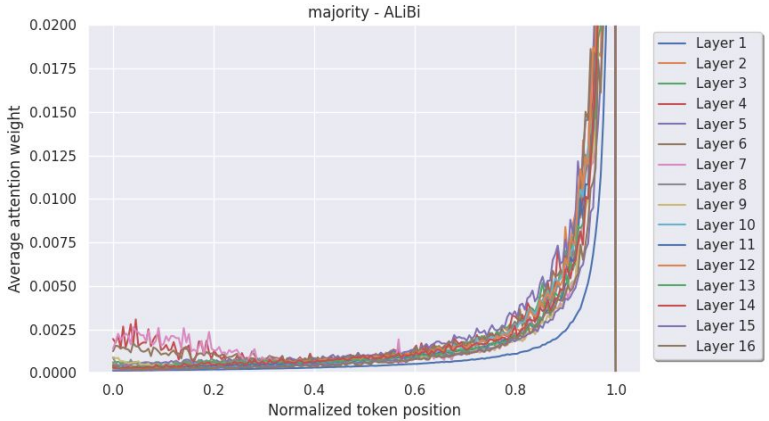
Results & Discussion

Majority

Average attention weight for each layer:



(a) NoPE



(b) ALiBi

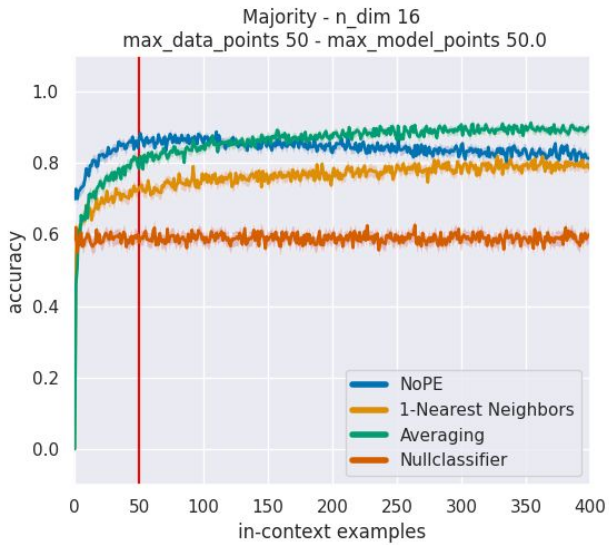
▲ Figure: Normalized token position means 0.0 is the farthest token and 1.0 is the most recent token.

Recap: Who wins?

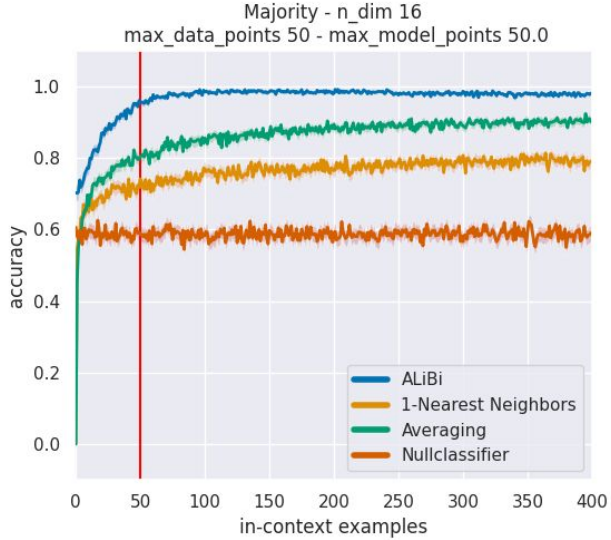
Majority

Results & Discussion

WIN



(a) NoPE



(b) ALiBi

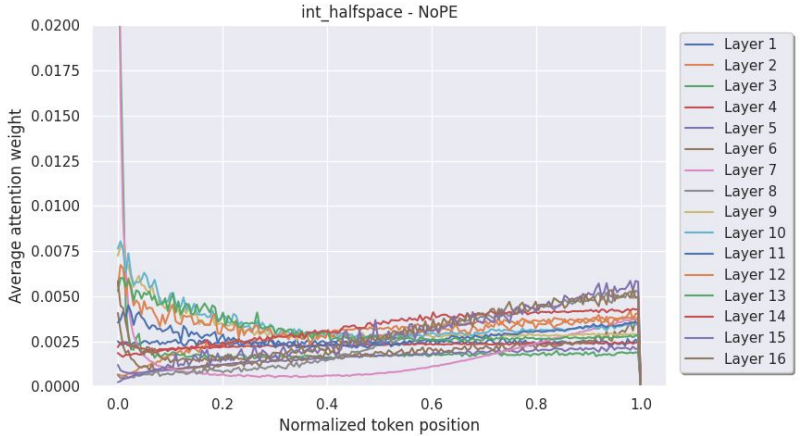
▲ Figure: Actually ALiBi has better performance on Majority.

Can recency bias explain why transformer fails?

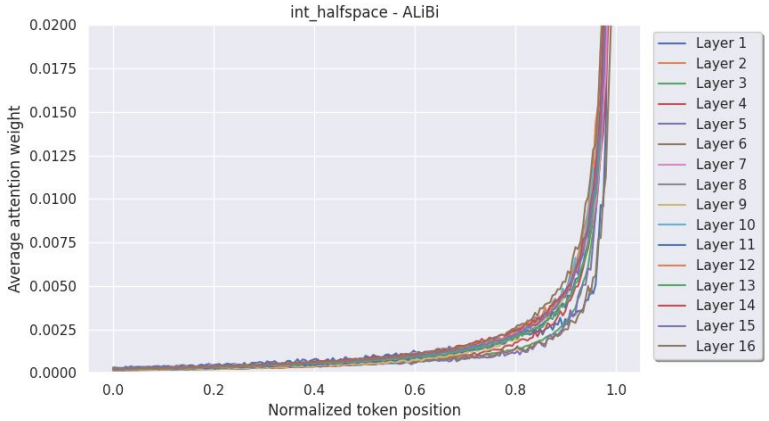
Results & Discussion

Integer Halfspace

Average attention weight for each layer:



(a) NoPE



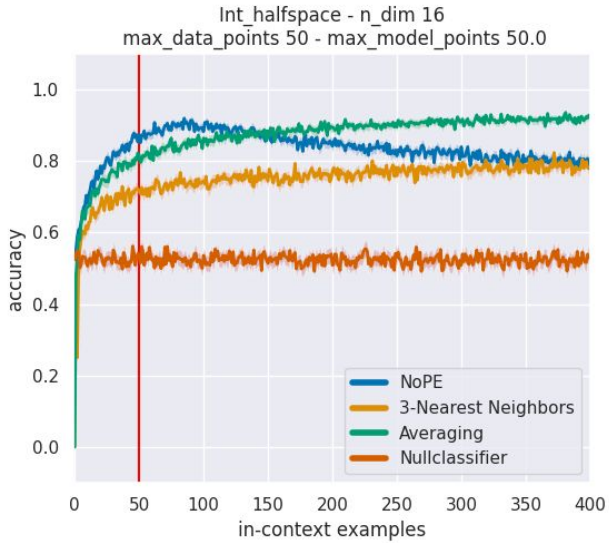
(b) ALiBi

▲ Figure: Normalized token position means 0.0 is the farthest token and 1.0 is the most recent token.

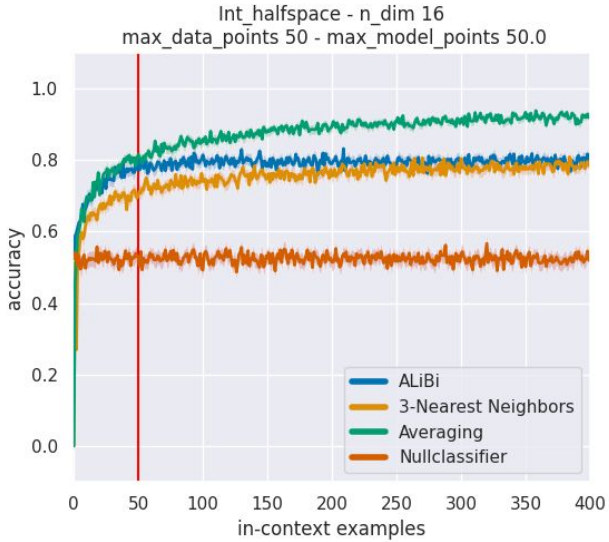
Recap: Who wins?

Integer Halfspace

Results & Discussion



(a) NoPE



(b) ALiBi

▲ Figure: Although NoPE has higher accuracy, its score is still decaying.

Can recency bias explain why transformer fails?

Conclusion

Recency bias does not explain why some PEs produce errors at OOD lengths.

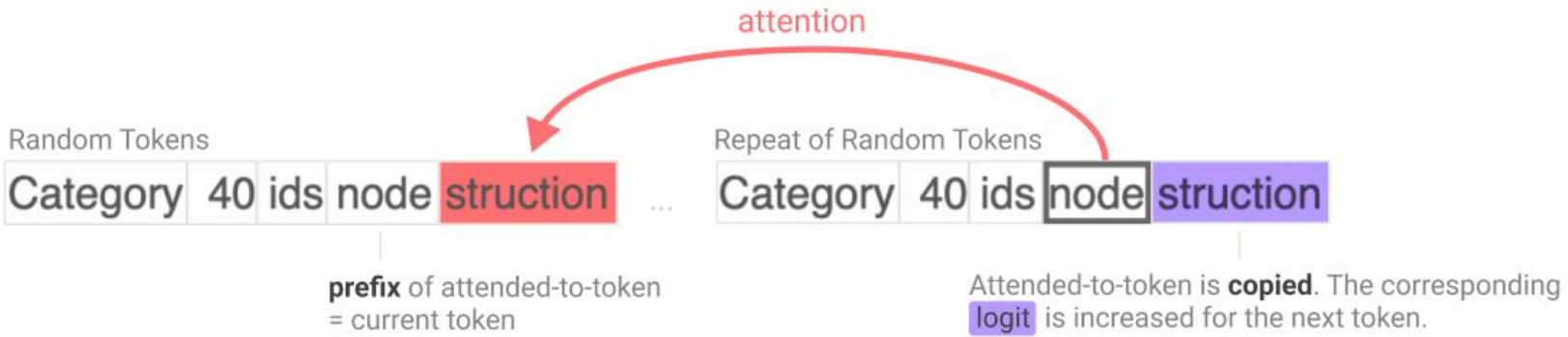
Instead, when the data is simple and each token's importance is uniform, recency bias appears to support length generalization.

Does inductive bias exist in In-Context Meta Learning?

Background

Induction Heads: (Olsson et al., 2022)

When LM can do ICL = LM has Induction Heads

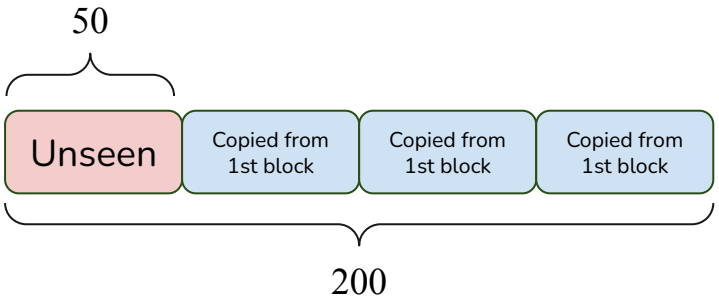


▲ Figure from Olsson et al., 2022 : Induction Heads is an important ability for ICL.

Does inductive bias exist in In-Context Meta Learning?

Results & Discussion

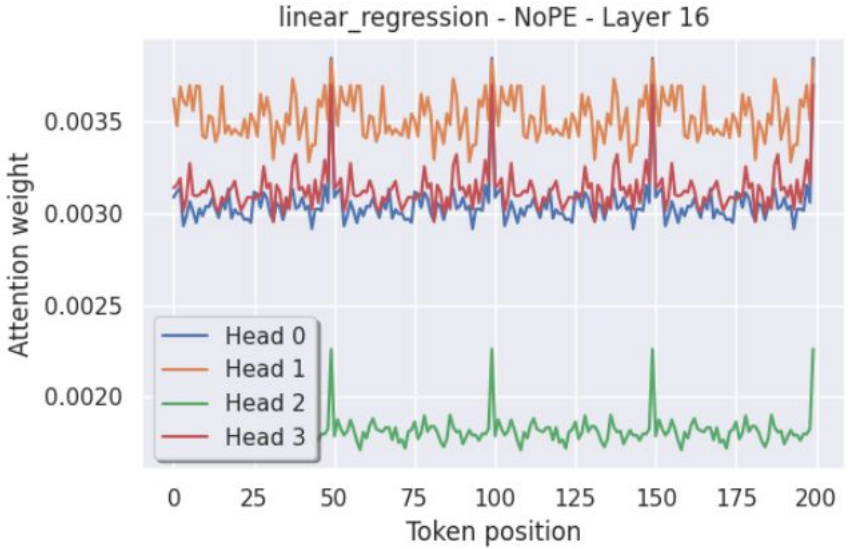
1 Repeated Examples:

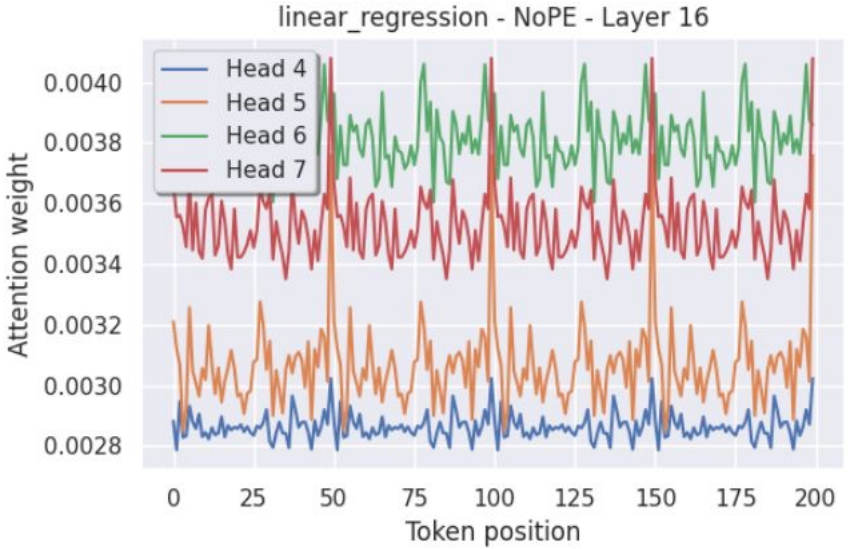


Does inductive bias exist in In-Context Meta Learning?

Results & Discussion

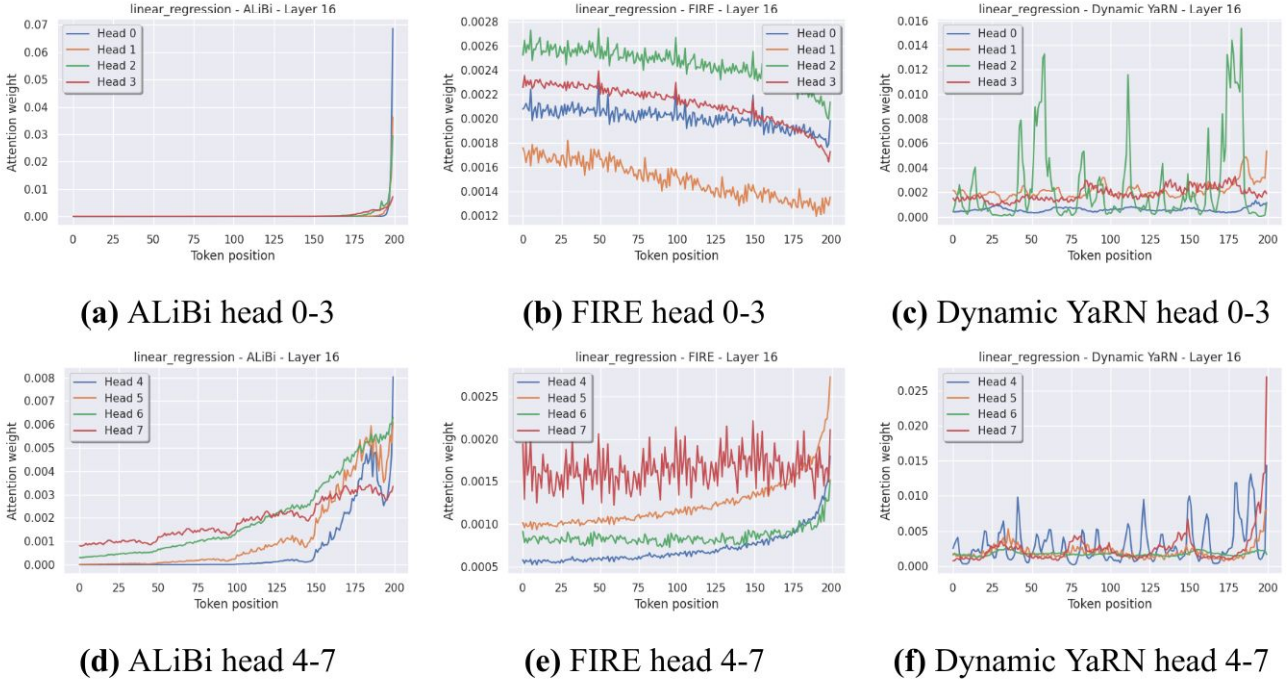
1





▲ Figure: Induction Heads exist in NoPE, it knows how to “cheat”.

Does inductive bias exist in In-Context Meta Learning?



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

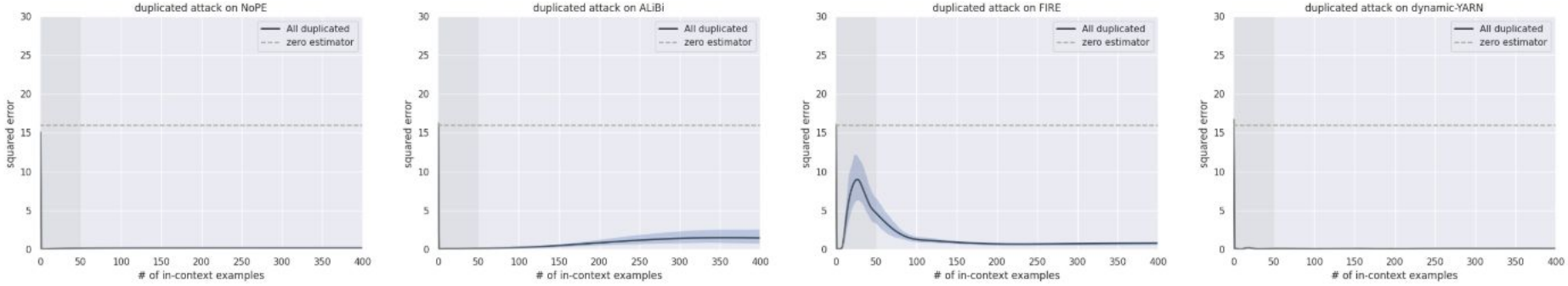
(d) ALiBi head 4-7 **(e)** FIRE head 4-7 **(f)** Dynamic YaRN head 4-7

1

▲ Figure: Induction Heads disappear when we change the PE.

Does inductive bias exist in In-Context Meta Learning?

2 **Repeated Examples**: All (x, y) is the same.



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

▲ Figure: Induction Heads disappear when we change the PE.

Does inductive bias exist in In-Context Meta Learning?

Conclusion

ICL ability of various PEs would disappear when encountering duplicated data.

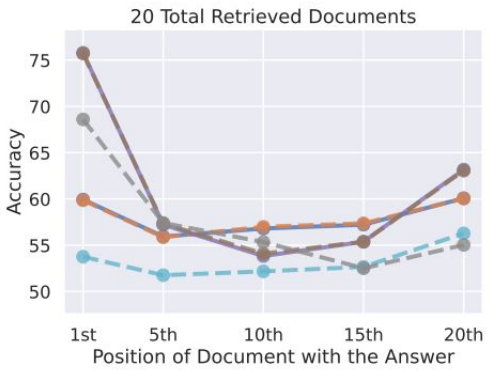
After incorporating PEs, although the model's length extrapolation is enhanced, its inductive ability is diminished. Therefore, there appears to be a trade-off between these two aspects.

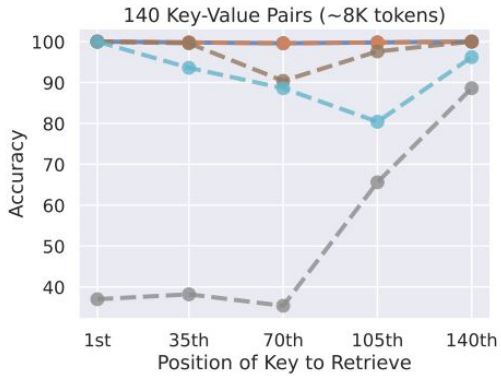
Does serial-position effect exist in IC Meta Learning?

Background

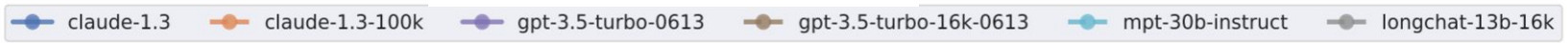
Lost in the Middle: (Hermann Ebbinghaus et al., 1913), (Liu et al., 2022)

Humans remember items at the beginning (primacy effect) or at the end (recency effect) of a list more easily





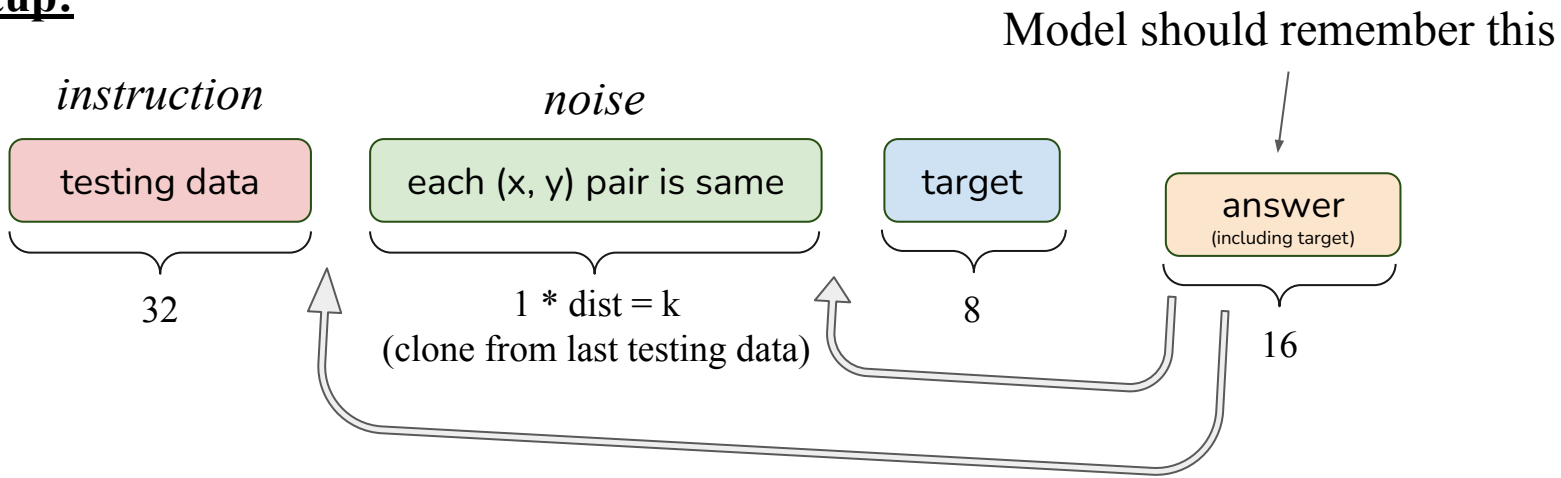
YaRN-7b (64K)	1st/ 1st	5th/ 35th	10th/ 70th	15th/ 105th	20th/ 140th
Doc	0.11	0.09	0.10	0.09	0.17
KV	0.74	0.23	0.50	0.12	0.90



Does serial-position effect exist in IC Meta Learning?

Results & Discussion

Setup:

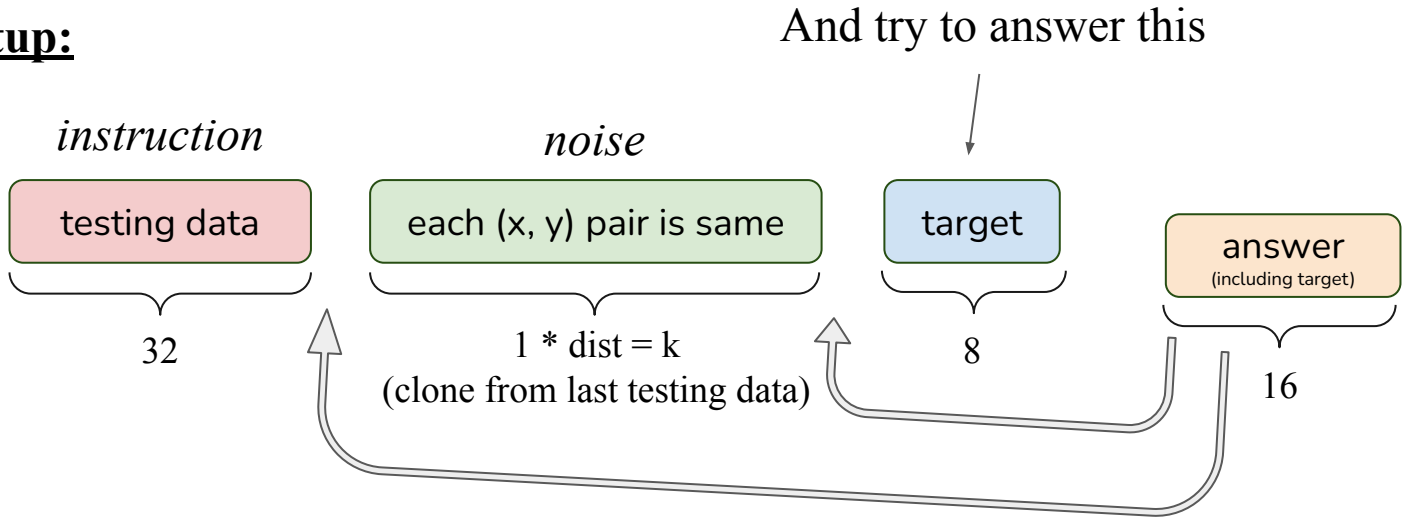


▲ Figure: Our experiment setting for Lost in the Middle.

Does serial-position effect exist in IC Meta Learning?

Results & Discussion

Setup:



▲ Figure: Our experiment setting for Lost in the Middle.

Does serial-position effect exist in IC Meta Learning?

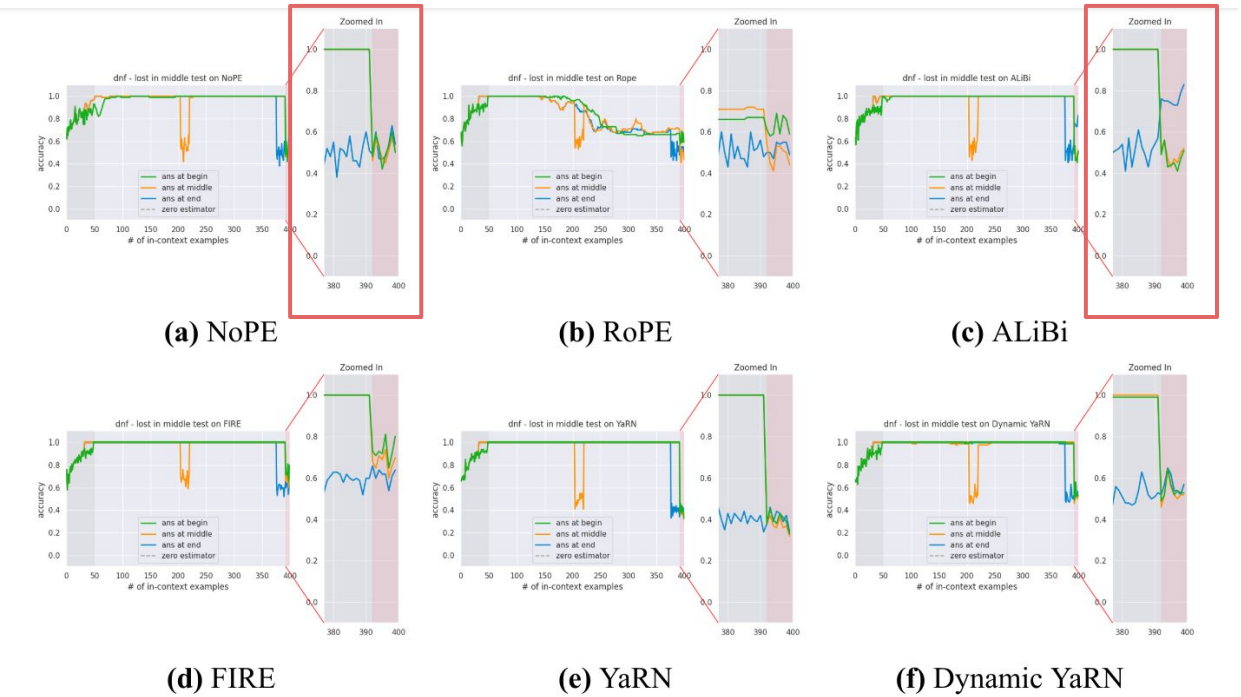
Results & Discussion

DNFs

Begin

Mid

End



(a) NoPE **(b)** RoPE **(c)** ALiBi

(d) FIRE **(e)** YaRN **(f)** Dynamic YaRN

Does serial-position effect exist in IC Meta Learning?

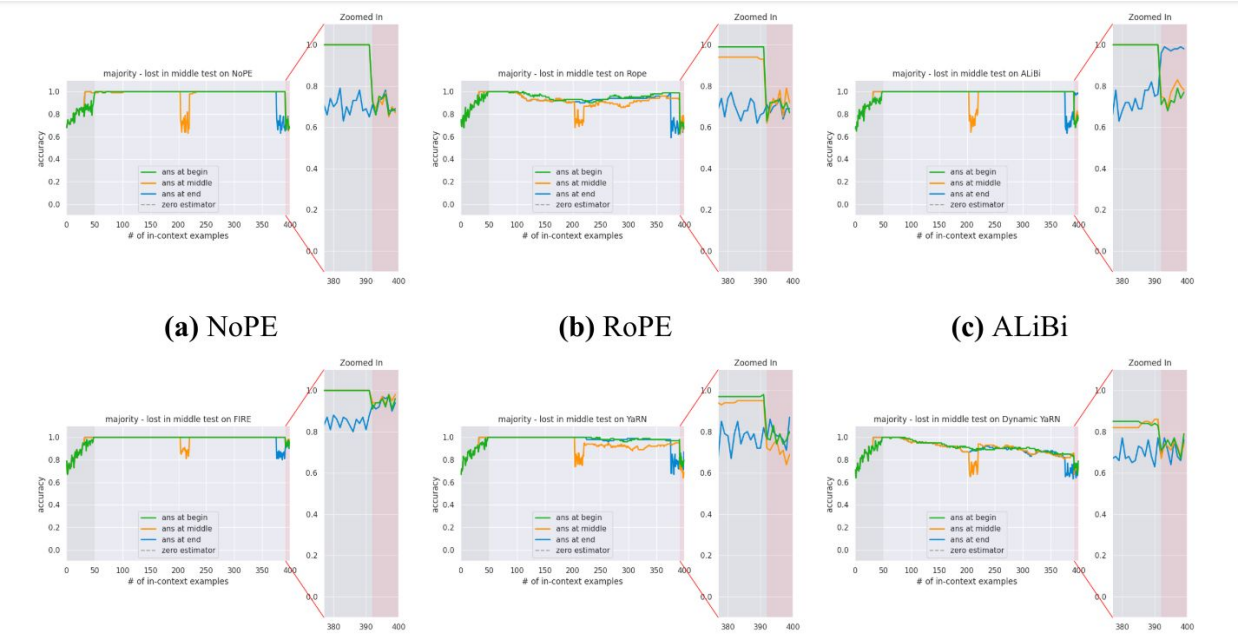
Results & Discussion

Majority

Begin

Mid

End



(a) NoPE **(b)** RoPE **(c)** ALiBi

(d) FIRE **(e)** YaRN **(f)** Dynamic YaRN

Does serial-position effect exist in IC Meta Learning?

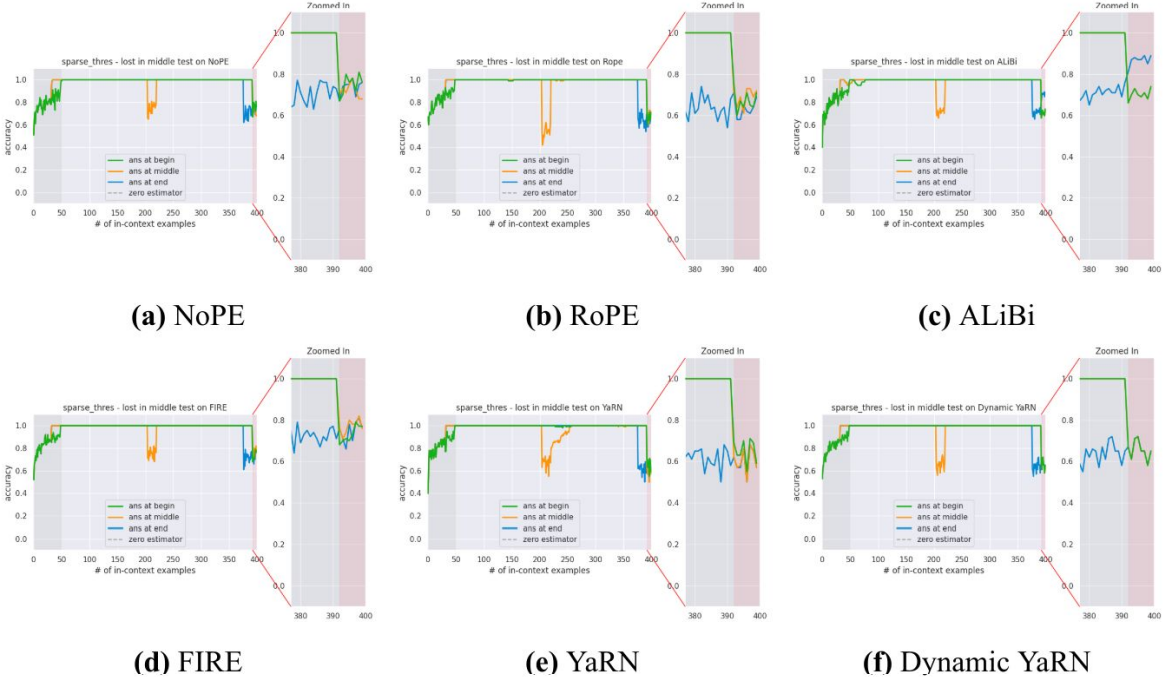
Results & Discussion

0-1 Threshold

Begin

Mid

End



(a) NoPE **(b)** RoPE **(c)** ALiBi

(d) FIRE **(e)** YaRN **(f)** Dynamic YaRN

Does serial-position effect exist in IC Meta Learning?

Conclusion

We conclude that this serial-position effect is independent of the PE.

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.

Conclusion

- Our findings indicate that in most tasks, NoPE is not the optimal choice. PEs can increase overall performance, including Length Generalization.
- There might be exist some trade-offs between Length Generalization and data generalization. We show that model will lose its induction heads for some PEs.
- We aim for these experimental designs and findings to provide future LLMs with more diverse evaluation criteria for selecting PEs, rather than relying methods such as perplexity.

Thank you for your listening!

Appendix

Why RoPE is relative?

Given q at position m and k at position n, we have:

$$
\begin{aligned}
\alpha_m &= \langle \text{RoPE}(q_m, m), \text{RoPE}(q_n, n) \rangle \\
&= \langle e^{im\theta_j} h, e^{im\theta_j} h \rangle \\
&= q_m k_n e^{im\theta_j} \overline{e^{in\theta_j}} \\
&= q_m k_n e^{i(m-n)\theta_j} \\
&= \text{RoPE}(q_m k_n, m - n)
\end{aligned}
$$

$$e^{ix} = \cos x + i \sin x$$

Why RoPE-base PEs can extrapolate?

$$\text{RoPE}(h, m) = e^{im\theta_j} h \quad (2.6)$$

Given a pre-trained context size L, our objective is to extend this size to L_o. We define a scaling factor $s = \frac{L}{L_o}$, and modify RoPE equation from 2.6:

$$\text{RoPE}'(h, m, \theta_j) = \text{RoPE}(h, g(m), l(\theta_j)). \quad (2.9)$$

- For PI: $g(m) = s \cdot m, \quad l(\theta_j) = \theta_j.$
- For NTK (ABF): $g(m) = m, \quad l(\theta_j) = (10000\lambda)^{\frac{-2j}{d}}, \quad where \quad \lambda = s^{\frac{d}{d-2}}.$

Why RoPE-base PEs can extrapolate?

- For YaRN, we first define a ramp function γ:

$$\gamma(d) = \begin{cases} 0, & \text{if } d < \alpha \\ 1, & \text{if } d > \beta \\ \frac{d-\alpha}{\beta-\alpha}, & \text{otherwise,} \end{cases}$$

we have:

$$g(m) = m,$$

$$h(\theta_d) = (1 - \gamma(r(d)))\,\frac{\theta_j}{s} + \gamma(r(d))\theta_j.$$

ALiBi

$$\text{softmax}(\mathbf{q}_i \mathbf{K}^\top + m \cdot [-(i-1), ..., -2, -1, 0]),$$

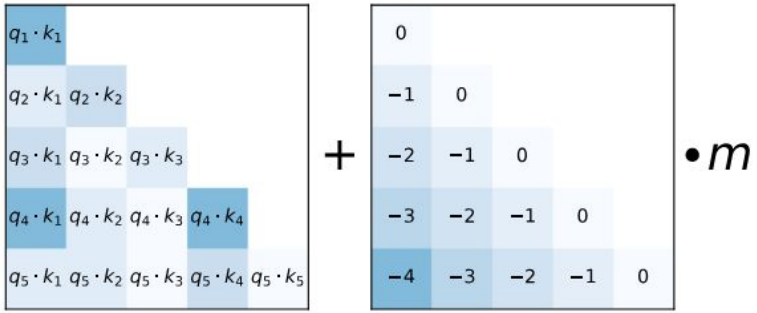


Figure 3: When computing attention scores for each head, our linearly biased attention method, ALiBi, adds a constant bias (right) to each attention score ($\mathbf{q}_i \cdot \mathbf{k}_j$, left). As in the unmodified attention sublayer, the softmax function is then applied to these scores, and the rest of the computation is unmodified. **m is a head-specific scalar** that is set and not learned throughout training. We show that our method for setting m values generalizes to multiple text domains, models and training compute budgets. When using ALiBi, we do *not* add positional embeddings at the bottom of the network.

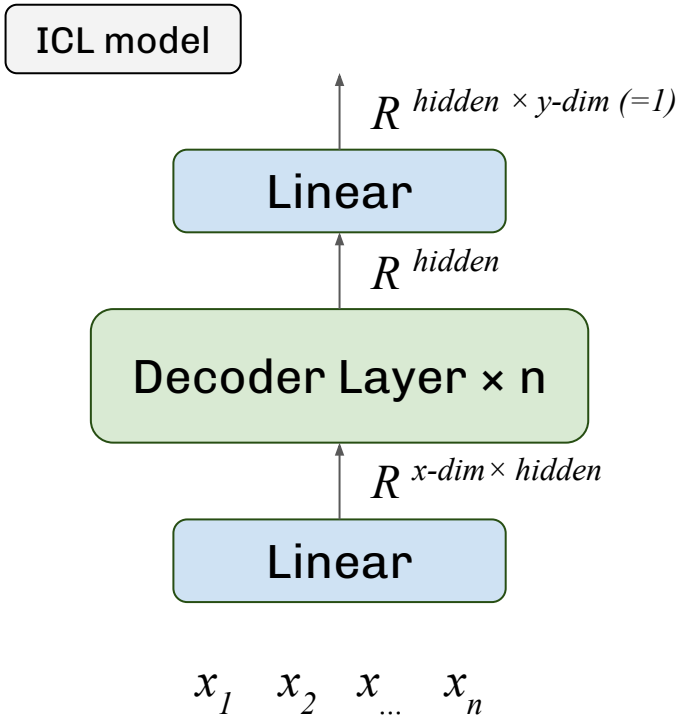
Preliminary: Set model max length = data max length

1. **Not practical**: During inference, the exact lengths of input and output sequences are unknown, and it is likely that these sequences will exceed the model's maximum length

2. **Not fail**: Certain PEs need training. Setting a larger model length without providing appropriately long training data would be unfair to these PEs.

3. We are training the model from scratch rather than using a pre-trained LLM for inference, we are not constrained by the inherent limitations of the model.

Model Architecture



▼ Table: Hyperparameters summary [Thesis p.19]

Hyperparameters	Value
Transformer Model	
Model Type	Causal (decoder-only)
Max Sequence Length	100
Hidden Size	256
Feed Forward Layer Hidden Size	1024
Decoder Layers	16
Attention Head	8
Normalization Layer	RMSNorm
Activation Function	SiLU

Task Description: Regression functions (Garg et al., NIPS 2022)

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 $\mathcal{N}(0, I_d)$.

Example

Given $d = 16$, $x = [0.1, 0.2, \ldots, 0.16]_{d=16}$, $w = [1, -2, \ldots, 16]_{d=16}$

Sparse Linear Regression:

Same definition as Linear Regression, but w has exactly s non-zero dims.

Task Description: Regression functions (Garg et al., NIPS 2022)

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 $\mathcal{N}(0, I_d)$.

Example

Given $d = 16$, $x = [0.1, 0.2, \ldots, 0.16]_{d=16}$, $w = [0, -2, \ldots, 0]_{d=16}$

s non-zero dims

Sparse Linear Regression:

Same definition as Linear Regression, but w has exactly s non-zero dims.

Task Description: Boolean functions (Bhattamishra et al., ICLR 2024)

Note that the sampler of Boolean functions is {0, 1}.

Conjunction:

The task is simply an $and(\wedge)$ operation of some subsets. The output = {0, 1}.

Example

Given $d = 16$, $X = [\, x_1, x_2, \ldots, x_{d-1}, x_d \,]_{\mathrm{d=16}}$, *output* = {0, 1}.

$w = x_2 \wedge \neg x_4 \wedge x_8$

Disjunction:

Similar to Conjunction, Disjunction change the operation to $or(\vee)$.

Task Description: Boolean functions (Bhattamishra et al., ICLR 2024)

Sparse Disjunction:

Same definition as Disjunction, but w has exactly s to choose. (we set $s = 3$)

CNFs & DNFs:

CNFs is a conjunction of one or more disjunctions. In contrast, DNFs is a disjunction of one or more conjunctions.

Example

$$w_{CNFs} = (x_2 \wedge \neg x_4 \wedge x_8) \vee \ldots \vee (\neg x_{11} \wedge x_{15} \wedge \neg x_{16})$$

Task Description: Boolean functions (Bhattamishra et al., ICLR 2024)

Sparse Majority:

The task outputs the majority number by selecting a subset of dimensions.

0-1 Threshold Functions:

The input space is changed to {-1, 1}, and can be define as $f : \text{sign}\left(\sum_{i=1}^{d} w_i x_i - b\right)$

, where $\text{sign} \in \{-, +\}, w_i \in \{-1, 0, 1\}, b \in \{-k, \cdots, k-1, k\}$.

Example

Given $d = 2$, $k = 3$, $X = [\, x_1 , x_2] = [\, -1, 1 \,]$, *output* can be:

output = (+ (1 * x_1 - 3)) + (- (0 * x_2 + 1)) = -5.

Task Description: Boolean functions (Bhattamishra et al., ICLR 2024)

Integer Halfspace:

Similar to 0-1 Threshold Functions, the only change is $b = 0.5$.

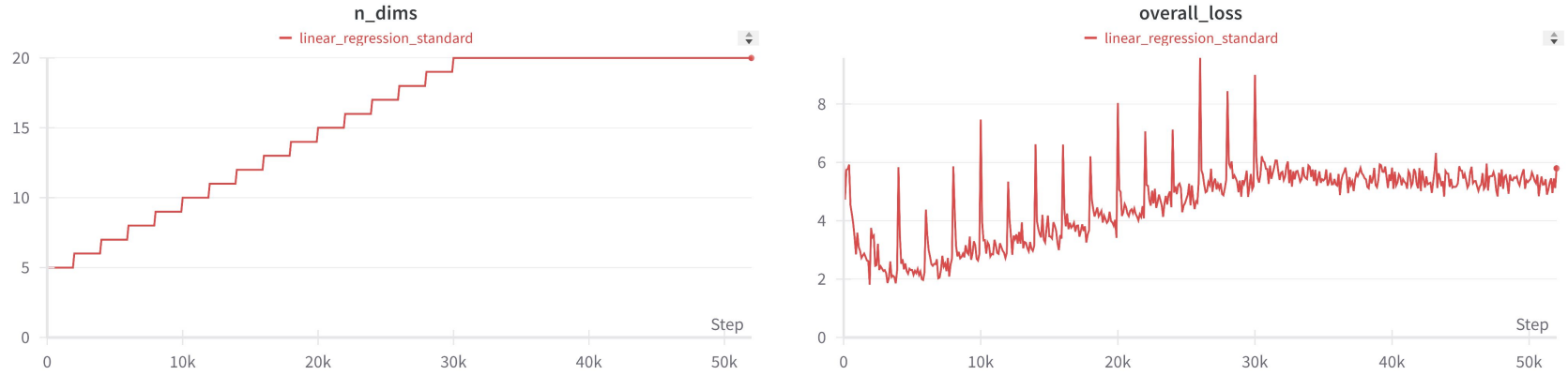
Parity:

The task is a XOR($\oplus$) of some subsets. The output will be 1 if input have odd number of 1s.

Sparse Parity:

Similar to other sparse tasks, we set $s = 2$ for this task.

Curriculum Learning



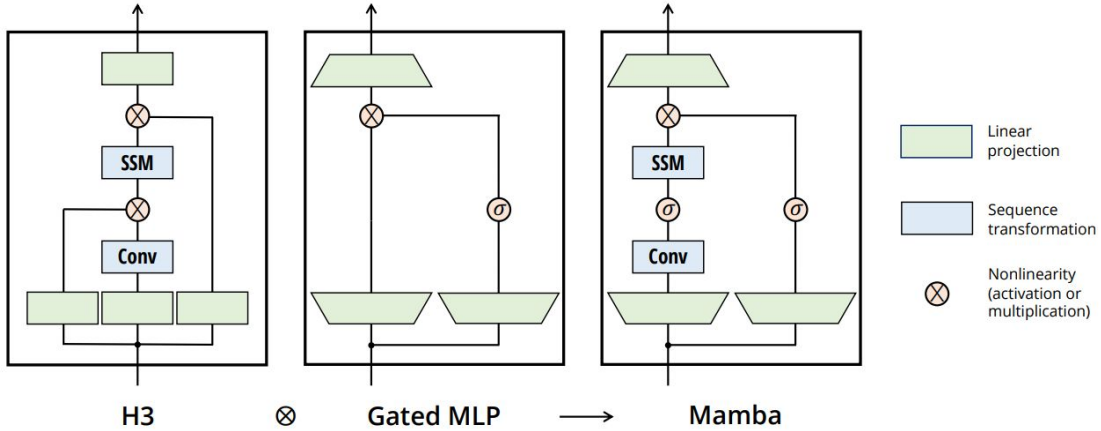
▲ Figure:Training log after applying curriculum learning. N_dims means the input x's vector dimensions.

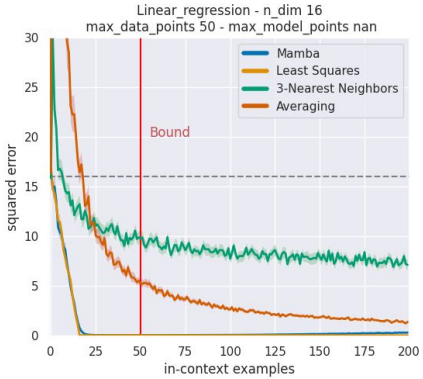
Baseline for Linear Regression

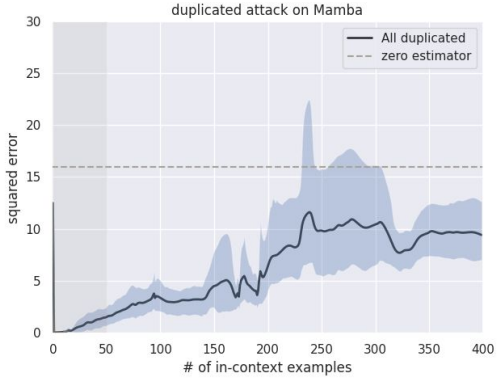
Given a set of data points, or prompt, $P = (x_1, y_1, \ldots, x_n, y_n, x_{\text{query}})$, we define that each baseline model M should solve for $\hat{w}$ and predict the output $\hat{y}$ with the target input x_{query}, i.e., $M(P) = \hat{w}x_{\text{query}}$. Note that $x \in \mathbb{R}^d$ and $y \in \mathbb{R}$.

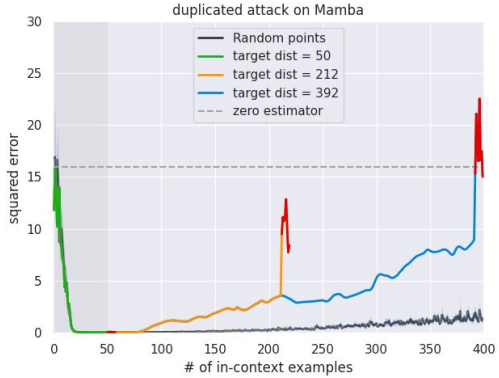
Lasso: The objective of Lasso is minimize $\frac{1}{2 \cdot n_{\text{samples}}} \|y - \hat{w}x\|_2^2 + \alpha \|\hat{w}\|_1$.

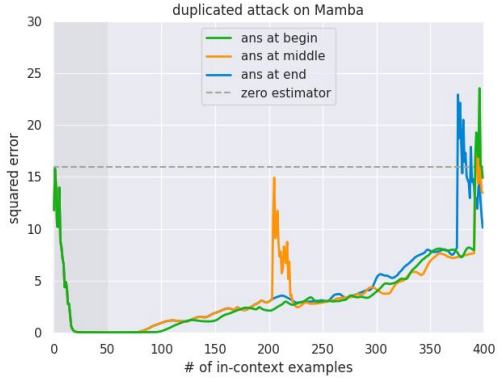
Mamba (Gu and Dao, 2023)





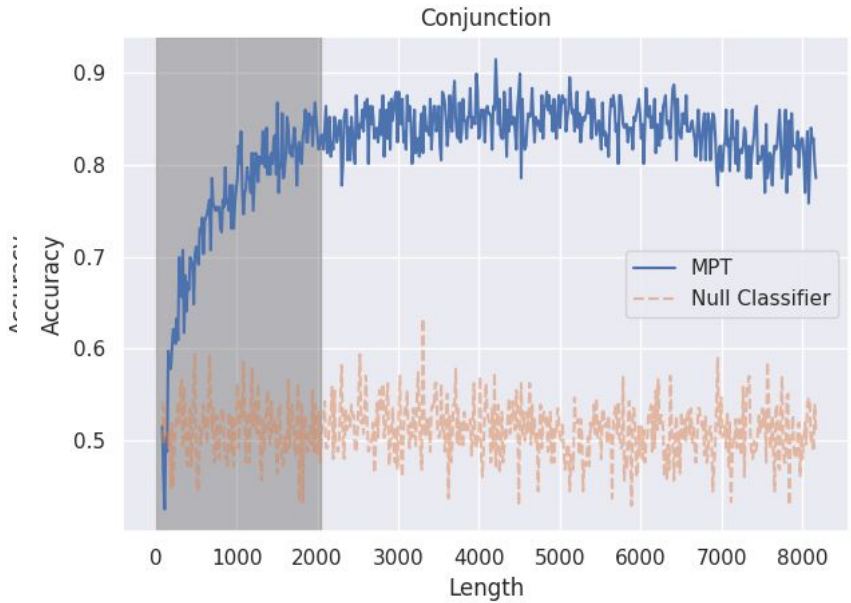


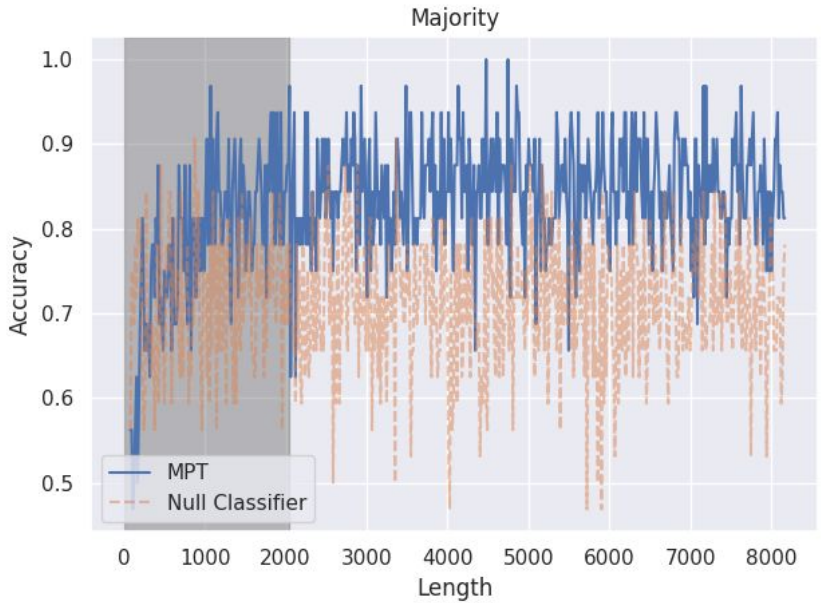




Inference on LLM: MPT-7B (ALiBi) and Llama2-7B (NTK)

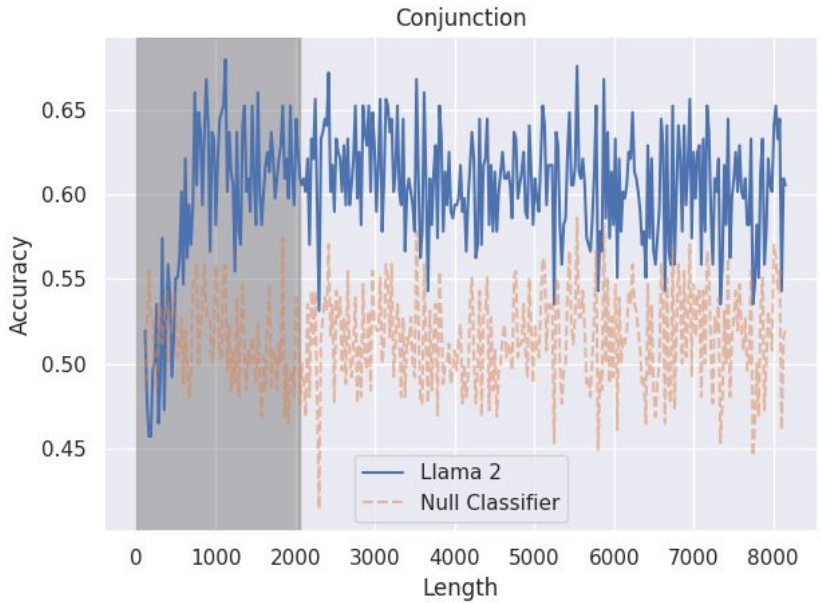
Standard settings (MPT)

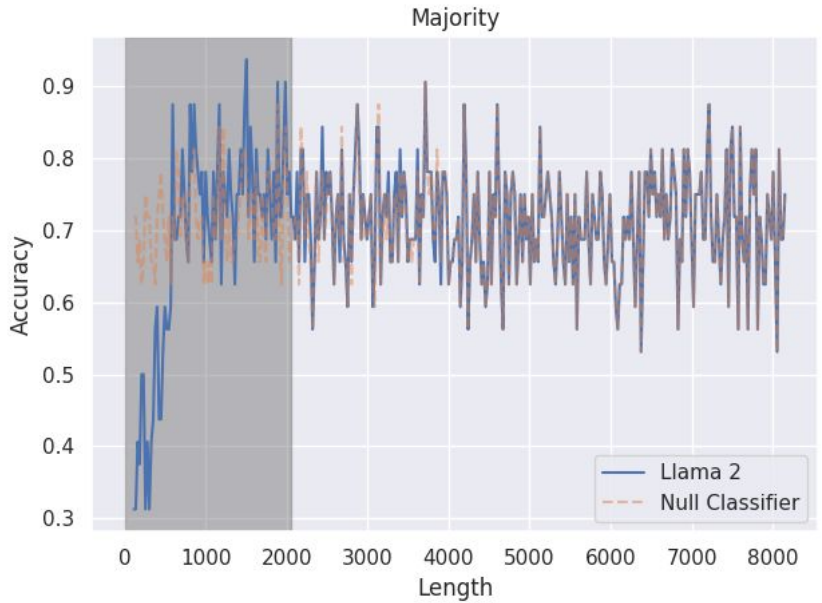




Inference on LLM: MPT-7B (ALiBi) and Llama2-7B (NTK)

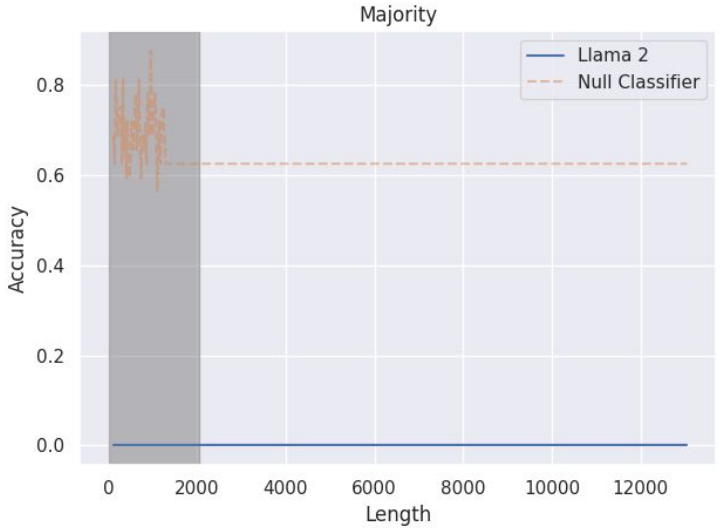
Standard settings (Llama)

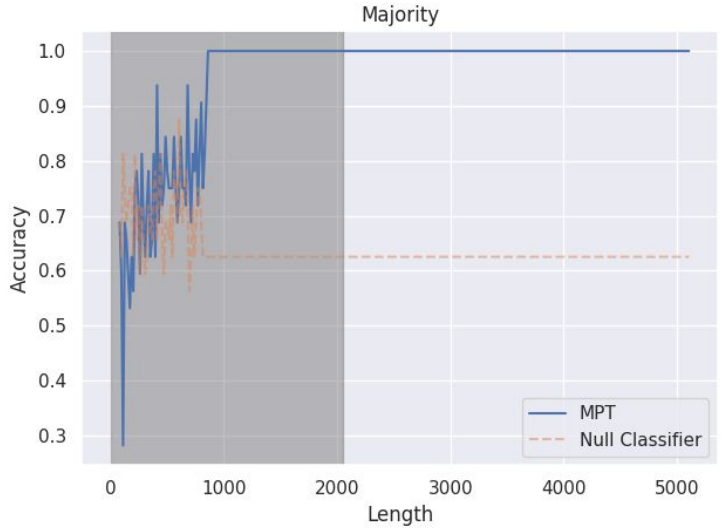




Inference on LLM: MPT-7B (ALiBi) and Llama2-7B (NTK)

Duplicated settings





Prompt for MPT

Below is an instruction that describes a task. Write a response that appropriately completes the request.
Instruction:
You are given some examples of inputs and their corresponding labels. You need to learn the underlying boolean function represented by these input-label examples. Predict the 'Label'(either 0 or 1) for the final input.

Input: 0 1 1 0 1 1 1 1
Label: 1
Input: 1 1 1 0 0 1 0 0
Label:

Prompt for Llama 2

<s>[INST] <<SYS>>
Below is an instruction that describes a task. Write a response that appropriately completes the request.
<</SYS>>
You are given some examples of inputs and their corresponding labels. You need to learn the underlying boolean function represented by these input-label examples. Predict the 'Label'(either 0 or 1) for the final input.

Input: 0 1 1 0 1 1 1 1
Label: 1
Input: 1 1 1 0 0 1 0 0[/INST]
Label:

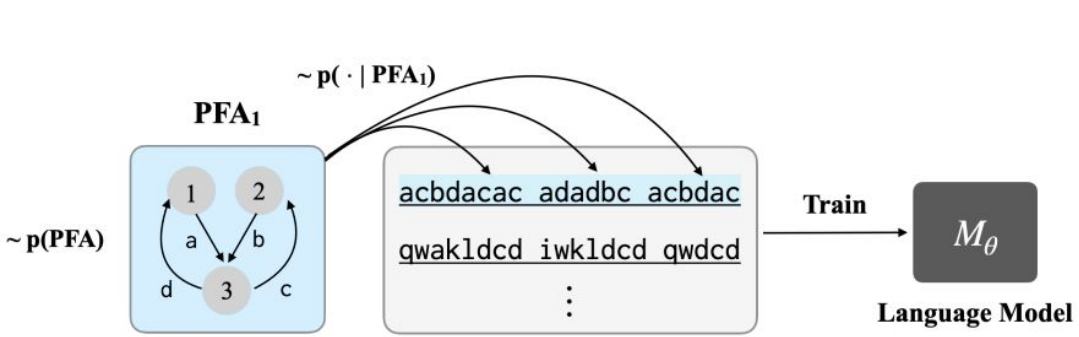
Action Items

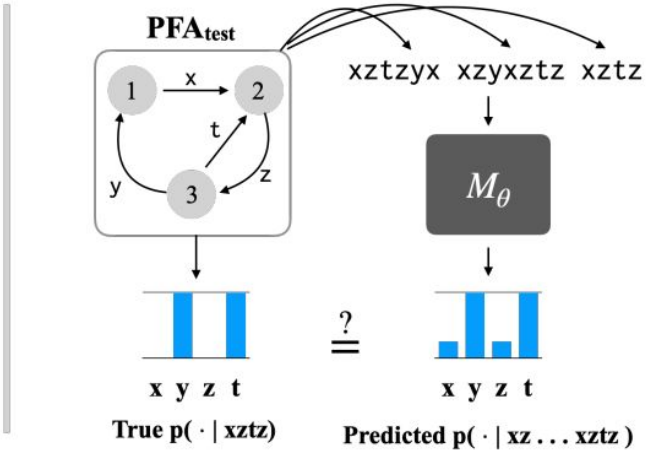
目標 1. 使用其他系列的架構進行實驗, 包含 : RNN、SSM、Linear time Transformer 等

- Attention: Transformer
- Linear Attention: Linear Transformer, (RetNet)
- Time Variant: LSTM, Mamba
- Time Invariant: RWKV, Hyena

Action Items

目標 2. 使用 probabilistic finite automata (PFA) 生成的語言來測試 Length Generalization 和 Position Encodings





AAAI paper 會強調的點

- NoPE 不是最佳解(過去的 papers 只測試在 NoPE 上), 加入 PE 可以改善準確度 & length generalization 。
- 現有的 PE 也存在著缺點, length generalization 能力好不代表 data generalization 也很好。

- 加入 limitation and discussion 的章節

修正內容總結

主要新增或修改的內容:

1. 增加 ch5 說明後續分析的目的以及必要性
   - NoPE 是否有其他優點，ALiBi 和 FIRE 是否有其他看不到的問題
   - ICL 有什麼已知的問題
2. 增加 ch5.5 (p.44~45) Will State Space Model Generalize on These Noise?
   - 不同於 transformer 架構的實驗 (Mamba)
3. 增加 ch5.6 (p.45~49) Further Experiments with Pretrained Large Language Models
   - Llama 2 和 MPT 的實驗結果

修正內容總結

其他細項調整:

1. 所有的 in-context meta learning -> in-context function learning
2. ch.3.2 改寫了 Problem Description 的一些內容，讓問題較為明確。
3. ch.3.3 增加任務實際數值範例的表格，讓輸入輸出更好理解
4. ch.6 增加一些 limitation 的敘述，說明實驗的 scope 是在 transformer 和 function learning 底下