ICLR

Arithmetic Transformers Can Length-Generalize in Both Operand Length and Count

ICLR 2025

Hanseul Cho*[1], Jaeyoung Cha*[1], Srinadh Bhojanapalli[2], Chulhee Yun[1]

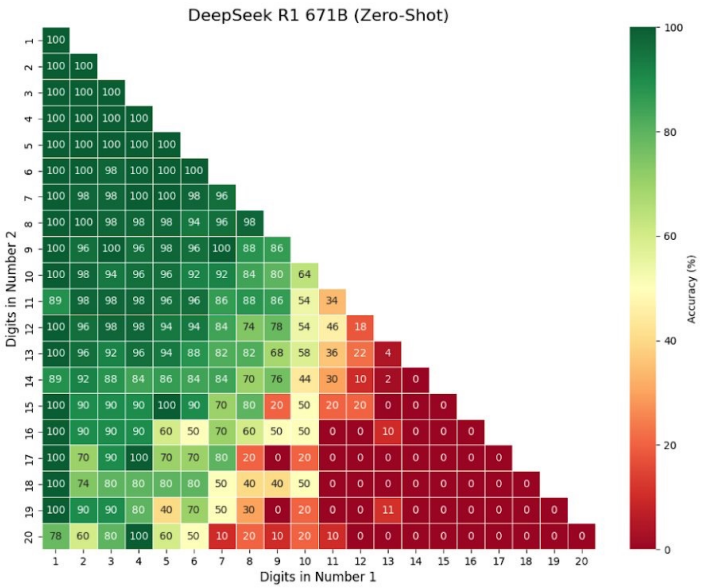
[1] Kim Jaechul Graduate School of AI [2] Google Research * Equal Contribution

KAIST AI
Kim Jaechul Graduate School

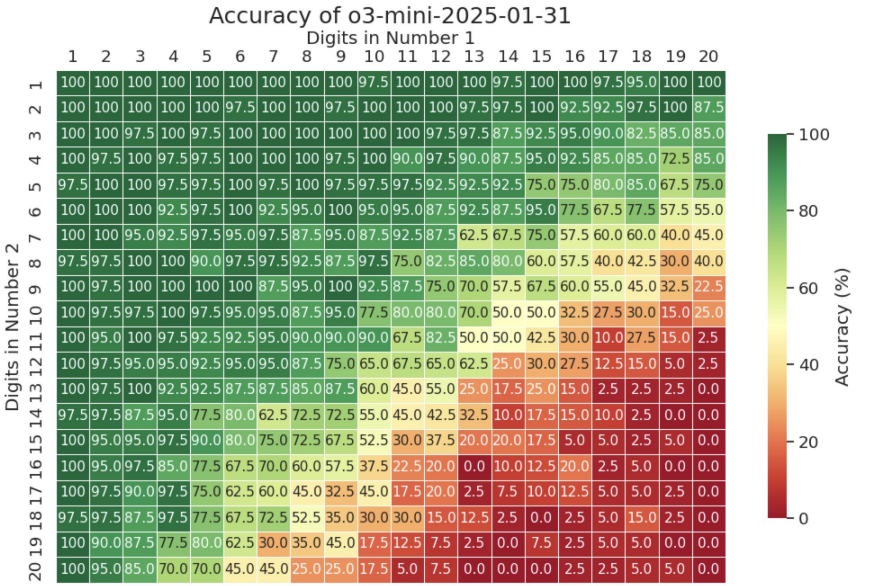
Google Research

LLMs v.s. Humans: Arithmetic Tasks

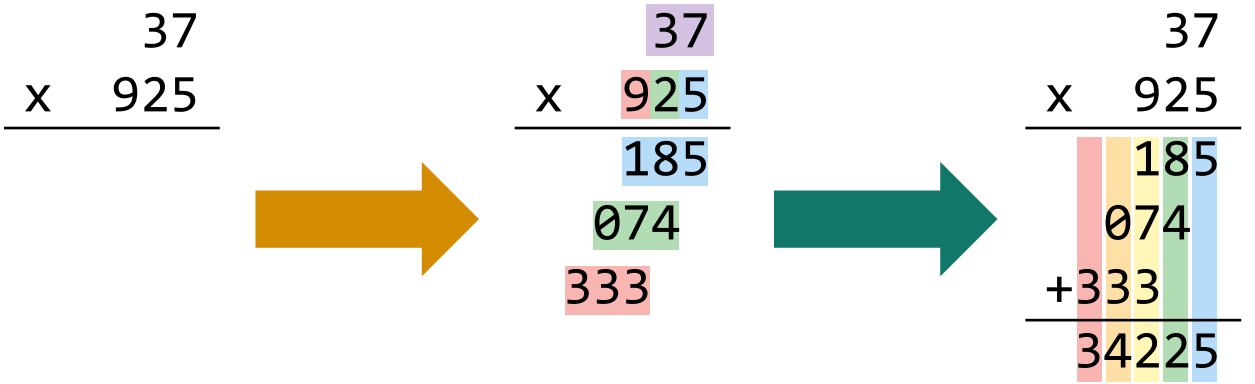
Integer Multiplication: DeepSeek-R1
(2025.02.03 on X @nouhadziri)



Integer Multiplication: OpenAI o3-mini
(2025.02.13 on X @yuntiandeng)

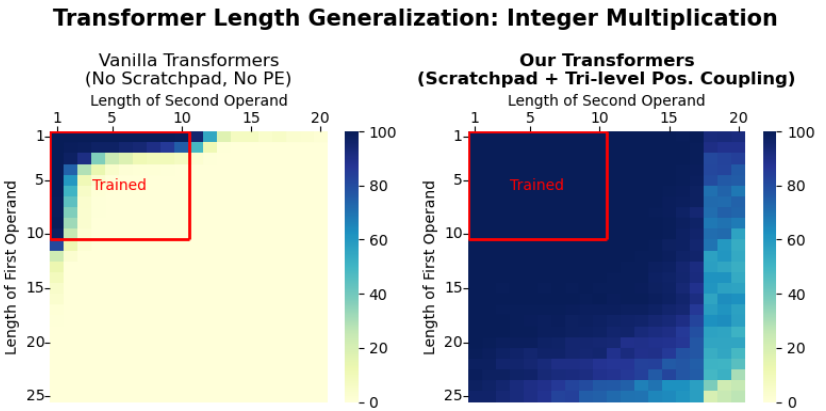


LLMs v.s. Humans: Arithmetic Tasks



Q) Can a Transformer🤖 do these👆 w/o training across the full range of context lengths?

🚨 Spoiler Alert:



Setup & Methods

- Arithmetic tasks:
  1. **Multi-operand addition** task: #(op) and len(op) can vary.

     AA…A + BB…B + … + ZZ…Z = ?
  2. **Integer multiplication** task: len(op1) and len(op2) can vary.

     AA…A x BB…B = ?
- Decoder-only Transformers, trained with next-token prediction (NTP).
  - Reasoning capability  as context length  (“Poor Length Generalization”)

Setup & Methods

- Arithmetic tasks:
  1. **Multi-operand addition** task: #(op) and len(op) can vary.

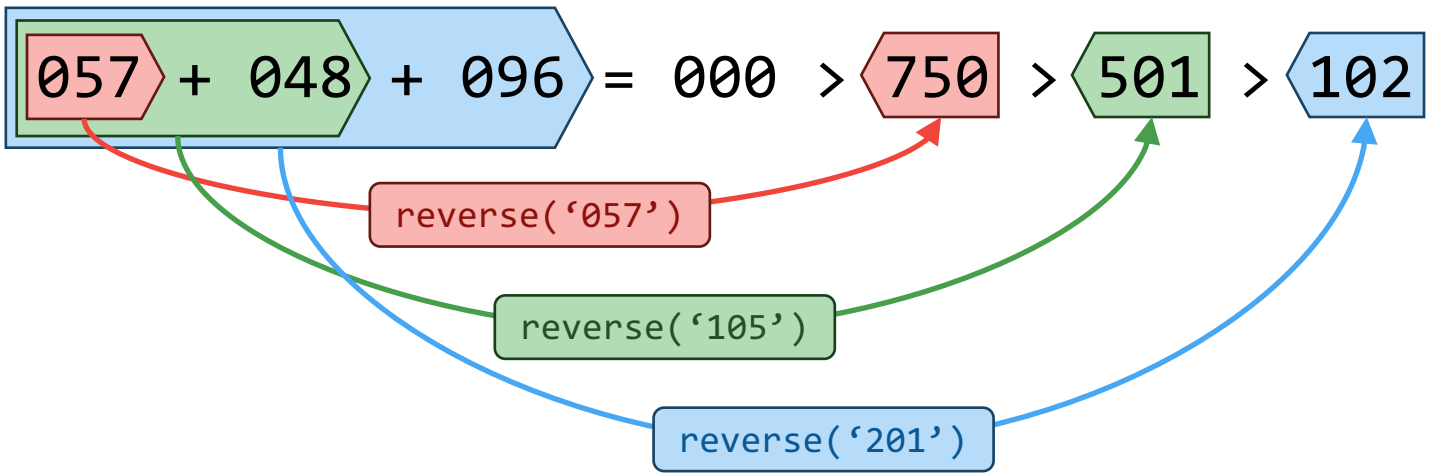
     AA…A + BB…B + … + ZZ…Z = ?
  2. **Integer multiplication** task: len(op1) and len(op2) can vary.

     AA…A x BB…B = ?
- Decoder-only Transformers, trained with next-token prediction (NTP).
  - Reasoning capability  as context length  ("Poor Length Generalization")

- Method: **Scratchpad** [Nye et al., 2021] + **Position Coupling** [Cho et al., 2024]
  - We designed novel **task-specific scratchpad formats** recording intermediate calculations.
  - We introduced a **multi-level Position Coupling** technique on top of our scratchpad.

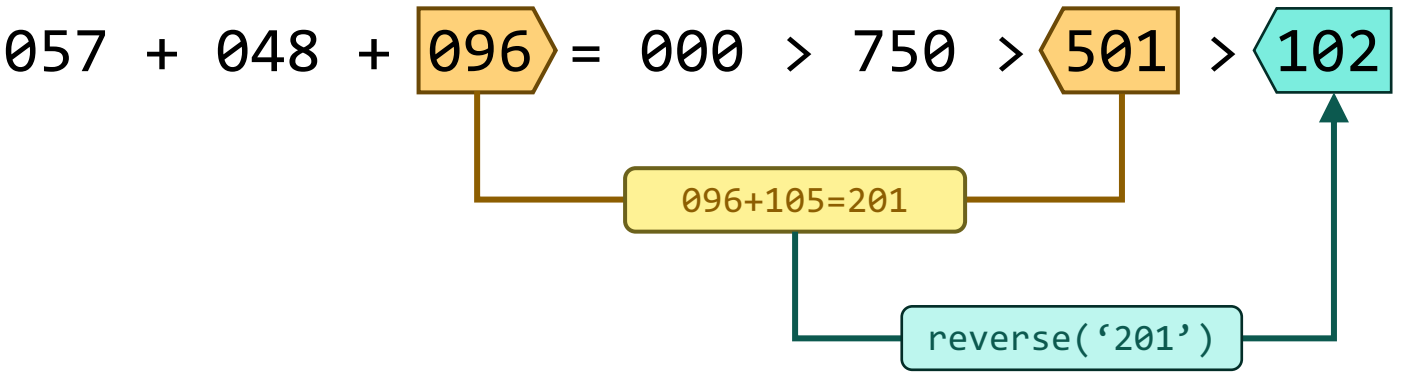
Multi-op Addition: Scratchpad

- Record Intermediate calculations.
- Zero-padding and Reversing the outputs can ease the decoder-only Transformer's calculation. [Lee et al., 2024]



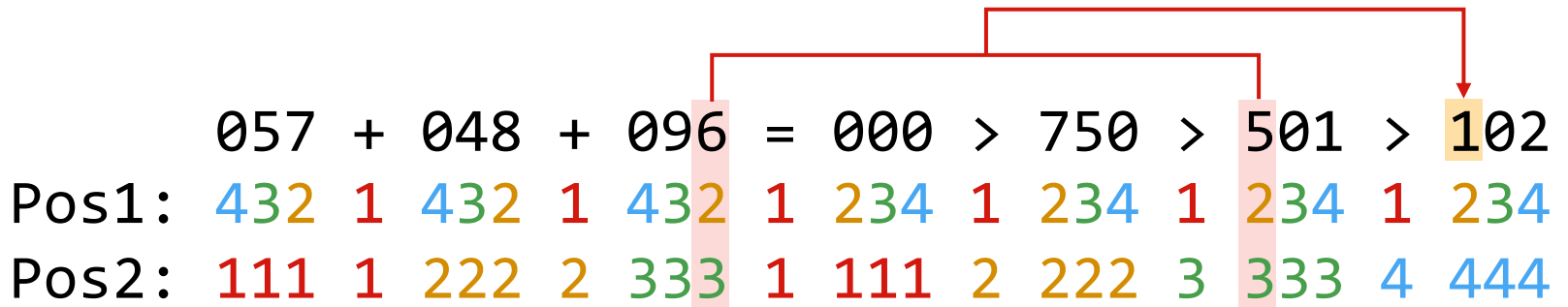
Multi-op Addition: Scratchpad

- Record Intermediate calculations.
- Every intermediate step is “possibly” easily proceeded by exploiting the previous calculations.



Multi-op Addition: Bi-level Position Coupling

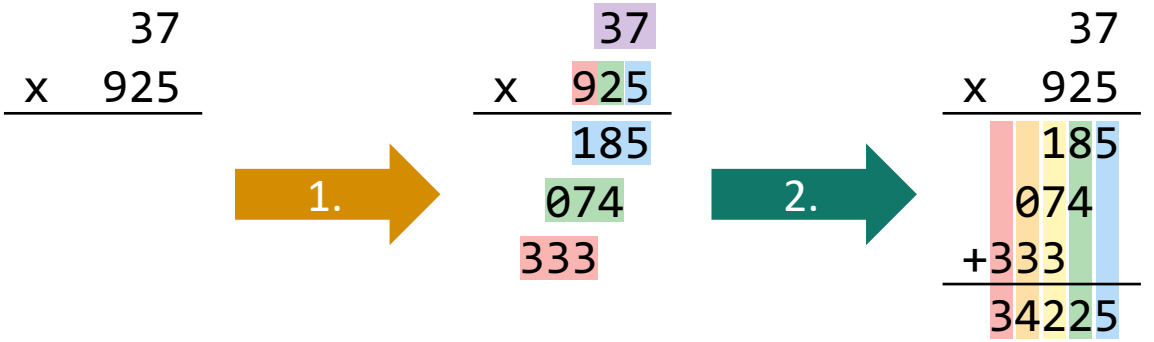
- Position Coupling [Cho et al., 2024]
  - Task-specific position ID assignment rule that **guides attention to the correct positions**, thereby improving length generalization.



- Pos. ID #1: same IDs to identically significant digits
- Pos. ID #2: distinguishing the numbers

(N-digit) x (M-digit) Multiplication task

- Two-stage Scratchpad:
  1. (N-digit) x (1-digit) multiplications M times
  2. M-operand addition



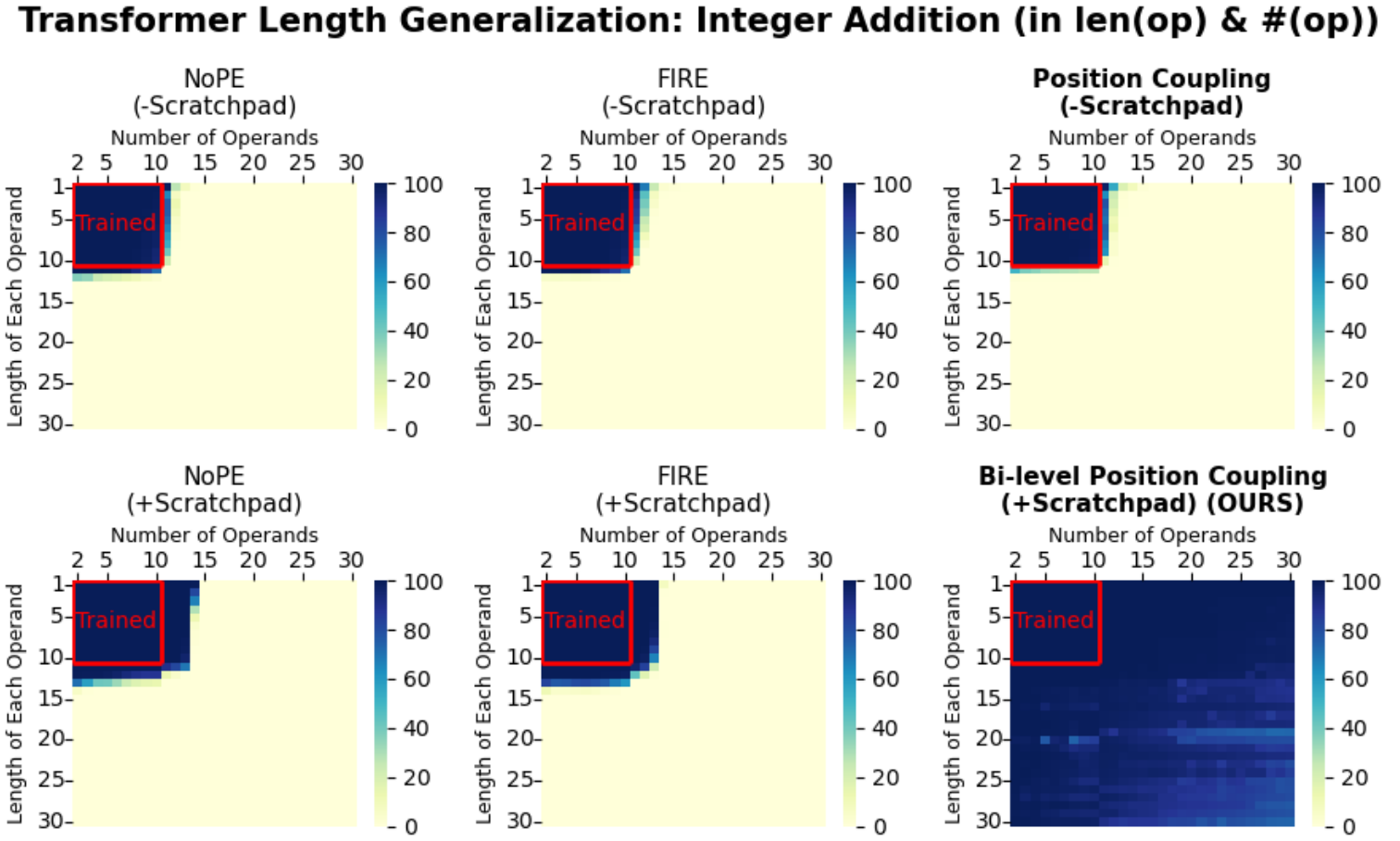
Two-stage Scratchpad + Tri-level Position Coupling

```
: 37 * 925 = 581 + 470 + 333 = 58100 > 52900 > 52243
Pos1: 32 0 000 1 234 1 234 1 234 0 00000 0 00000 0 00000
Pos2: 00 0 321 1 111 2 222 3 333 1 11111 2 22222 3 33333
Pos3: 00 0 000 1 234 2 345 3 456 1 23456 1 23456 1 23456
```

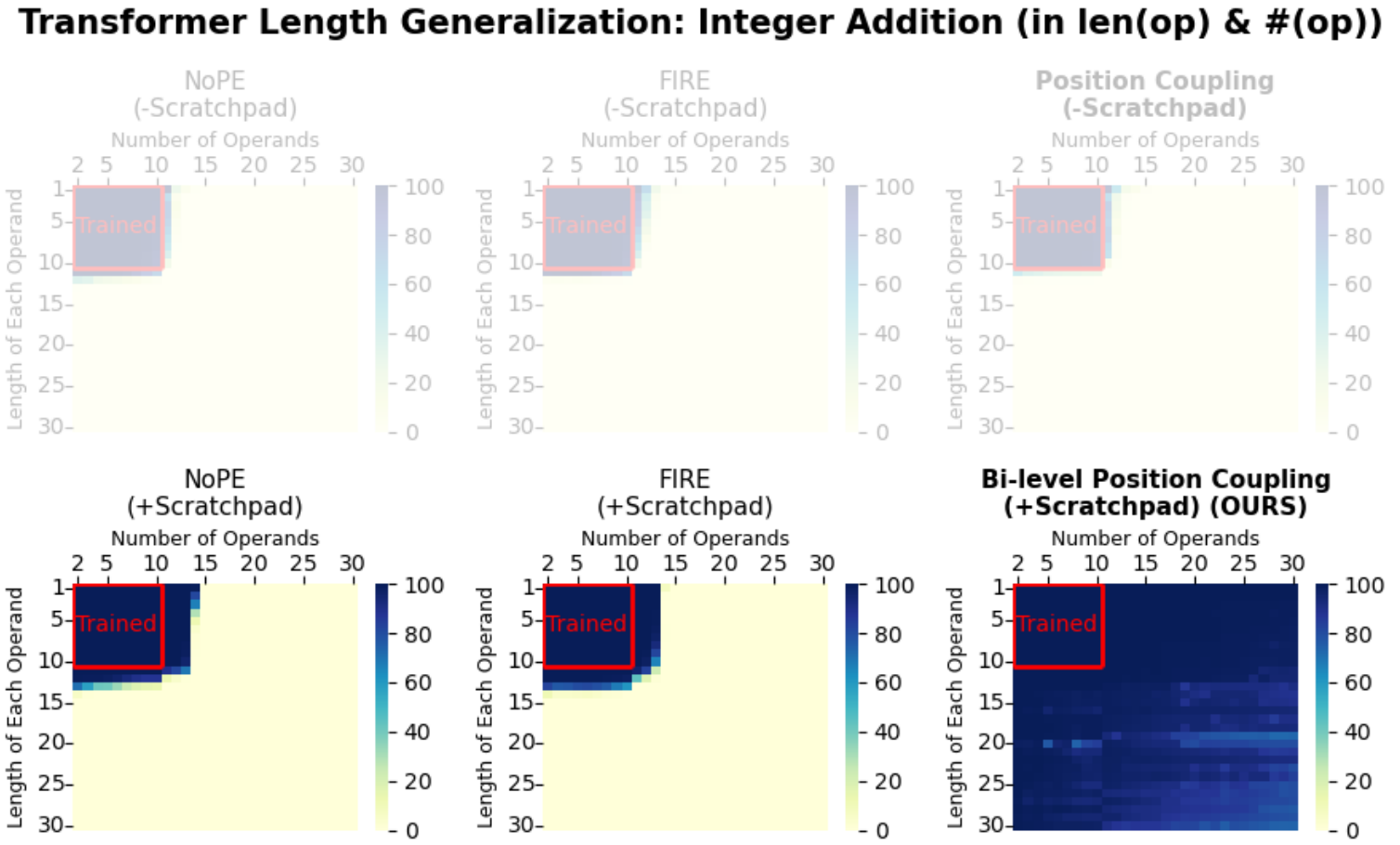
st.1.

st.2.

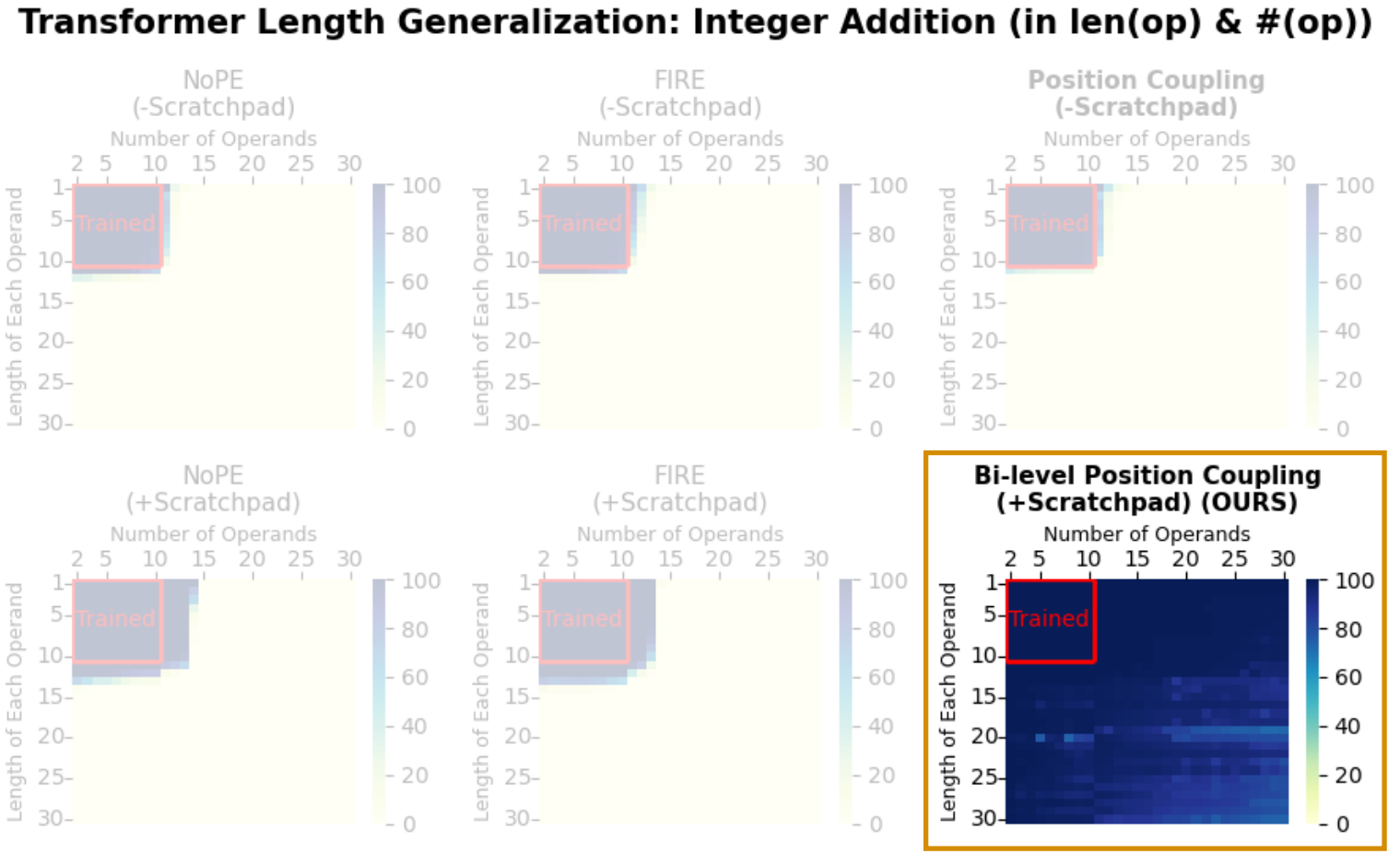
Experimental Results: Multi-op Addition



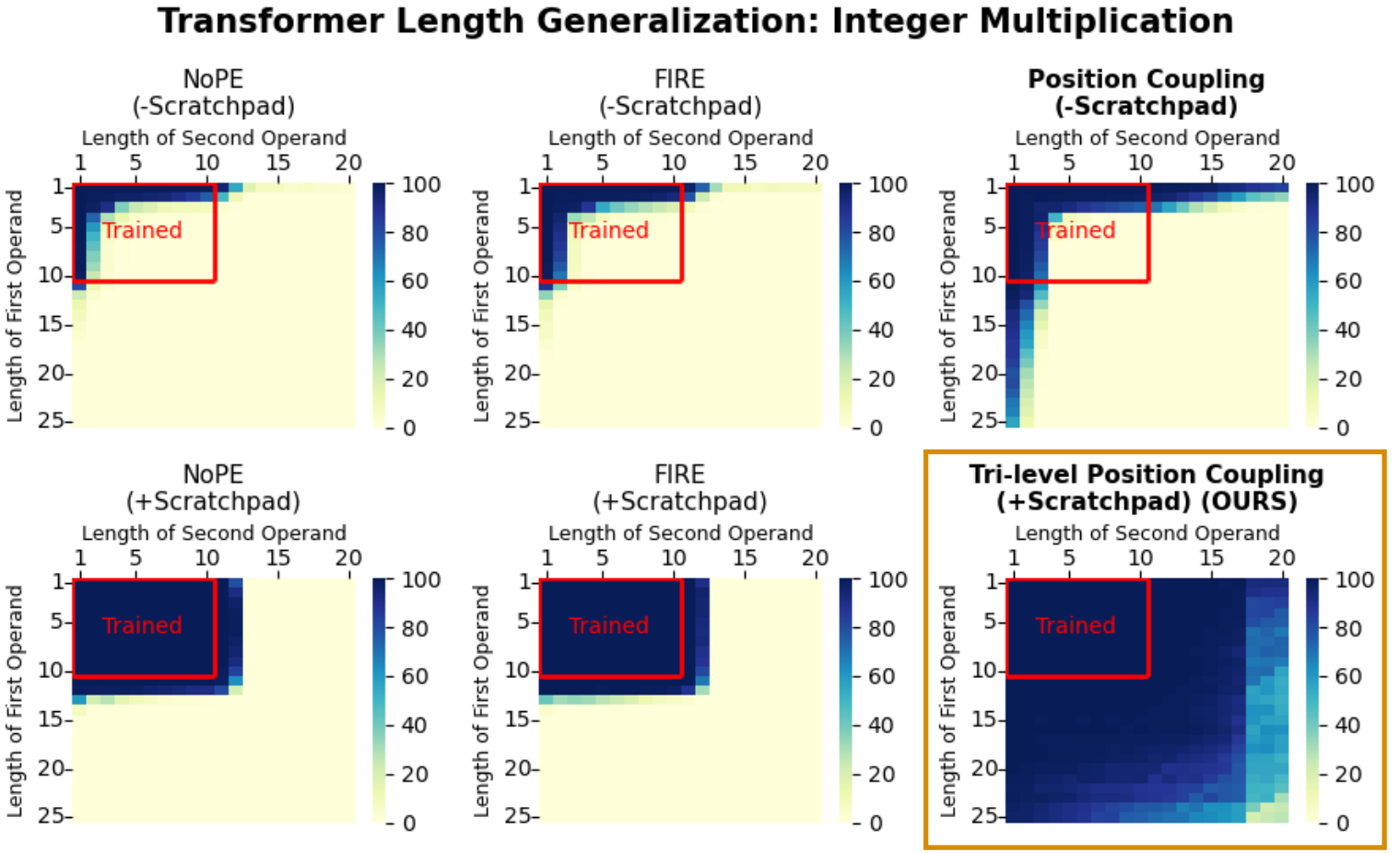
Experimental Results: Multi-op Addition



Experimental Results: Multi-op Addition



Experimental Results: Multiplication



Theoretical Result

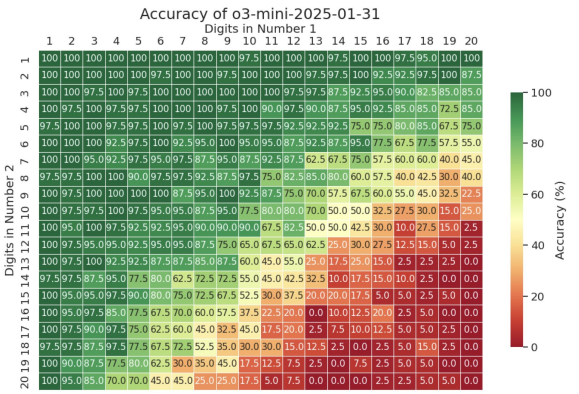
Theorem 4.1. There exists a parameter configuration of a **1-layer 4-head** decoder-only Transformer that entirely solves the addition task, with up to m summands each having n digits, by using our scratchpad and bi-level Position Coupling. Here, a sufficient choice of the embedding dimension is

$$d = \mathcal{O}(\log m + \log n).$$

- Applying our method, a shallow Transformer can solve additions with **exponentially many & exponentially long summands** in embed. dim.
  - The proof is constructive.
  - It is an extension of Theorem 5.1 proved by Cho et al. (2024).
  - **Trivially extends to larger Transformers with more layers & attention heads.**

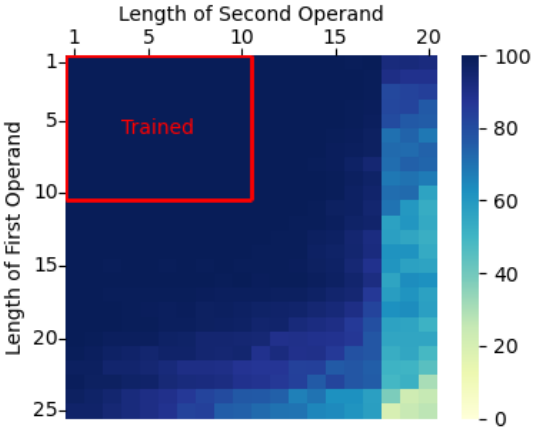
Recap & Takeaway

Integer Multiplication: OpenAI o3-mini
(2025.02.13 on X @yuntiandeng)



v.s.

Our Transformers
(Scratchpad + Tri-level Pos. Coupling)



Explicitly structuring intermediate calculation

➕

Rewiring positional relationships with position IDs

⏸

Significant Length Generalization in Arithmetics!

Poster Session #4

Fri 25 Apr 3 p.m. — 5:30 p.m. (GMT+08)

arxiv.org/abs/2410.15787

github.com/HanseulJo/position-coupling