Position Coupling: Improving Length Generalization of Arithmetic Transformers Using Task Structure

NeurIPS 2024
The 38th Annual Conference on Neural Information Processing Systems

Hanseul Cho*, Jaeyoung Cha*, Pranjal Awasthi,
Srinadh Bhojanapalli, Anupam Gupta, Chulhee Yun

KAIST AI Kim Jaechul Graduate School
Google Research
NYU

Can Transformers Learn Algorithms from Data?

Can Transformers Learn Algorithms from Data?

- Yes(?), for trained sequence lengths.
  - ≈ In-Distribution Generalization

Can Transformers Learn Algorithms from Data?

- Yes(?), for trained sequence lengths.
  - ≈ In-Distribution Generalization
- **Length Generalization (LG)**
  - **"Train Short, Test Long."** ≈ Out-of-Distribution Generalization
  - Proxy to study LLM's algorithmic understanding capability

Can Transformers Learn Algorithms from Data?

- Yes(?), for trained sequence lengths.
  - ≈ In-Distribution Generalization
- **Length Generalization (LG)**
  - **“Train Short, Test Long.”** ≈ Out-of-Distribution Generalization
  - Proxy to study LLM’s algorithmic understanding capability
- In terms of LG, Yes and No.
  - **Can**: Sorting, Mode, Counting, Copy/Reverse w/o duplicates, …
  - **Can’t**: Addition, Multiplication, Copy/Reverse with duplicates, Parity, …

Can Transformers Learn Algorithms from Data?

- Yes(?), for trained sequence lengths.
  - ≈ In-Distribution Generalization
- **Length Generalization (LG)**
  - **“Train Short, Test Long.”** ≈ Out-of-Distribution Generalization
  - Proxy to study LLM’s algorithmic understanding capability
- In terms of LG, Yes and No.
  - **Can**: Sorting, Mode, Counting, Copy/Reverse w/o duplicates, …
  - **Can’t**: Addition, Multiplication, Copy/Reverse with duplicates, Parity, …

Can we inject the known structure of a task into a decoder-only Transformer so that it can automatically length-generalize?

Method: Position Coupling

- Main Contribution:
  - Trained Transformers on problem lengths 1-30 for several arithmetic & algorithmic tasks (**Addition**, Multiplication, Copy/Reverse with duplicates,...).
  - Achieved a robust and near-perfect generalization to problem length 200: $\approx 6.67\times$ **length extrapolation!**

Method: Position Coupling

- Main Contribution:
  - Trained Transformers on problem lengths 1-30 for several arithmetic & algorithmic tasks (**Addition**, Multiplication, Copy/Reverse with duplicates,…).
  - Achieved a robust and near-perfect generalization to problem length 200: $\approx 6.67\times$ **length extrapolation!**

- Established on top of **learned APE** (e.g., GPT3)
- … with a **task-specific position ID assignment rule**.

Method: Position Coupling

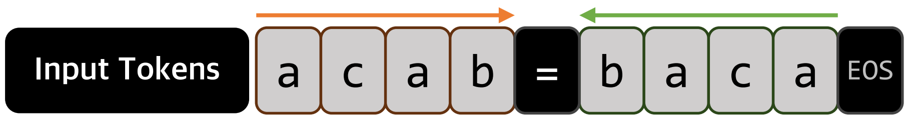
- Main Contribution:
  - Trained Transformers on problem lengths 1-30 for several arithmetic & algorithmic tasks (**Addition**, Multiplication, Copy/Reverse with duplicates,...).
  - Achieved a robust and near-perfect generalization to problem length 200: $\approx 6.67\times$ **length extrapolation!**

- Established on top of **learned APE** (e.g., GPT3)
- ... with a **task-specific position ID assignment rule**.

- Suppose we know/have:
  - A **task** we want a decoder-only Transformer to solve by NTP
  - **Structure between token positions** (regardless of sequence length)
  - A proper input formatting technique (e.g., reversing, zero-padding)

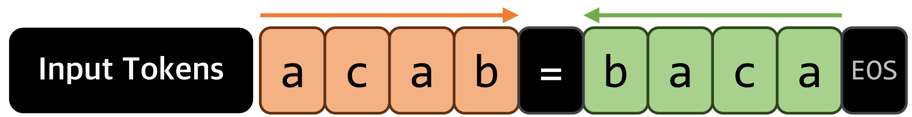
Method: Position Coupling (Reverse task)

- Position ID assignment rule for each input sequence:



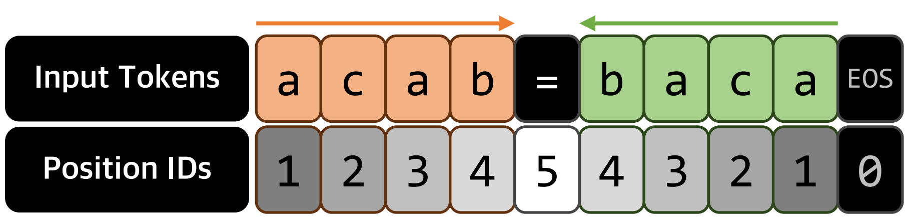
Method: Position Coupling (Reverse task)

- Position ID assignment rule for each input sequence:
  1) Group input tokens into “chunks” of (consecutive) tokens



Method: Position Coupling (Reverse task)

- Position ID assignment rule for each input sequence:
  1) Group input tokens into “chunks” of (consecutive) tokens
  2) Assign the same position ID to the tokens at the same “significance”
     - Every token in each chunk is of a unique significance
     - We assign consecutive position IDs for consecutive tokens in each chunk



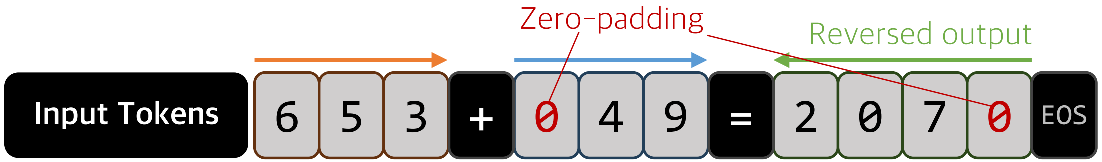
Method: Position Coupling (Reverse task)

- **Position ID assignment rule** for each input sequence:
  1) Group input tokens into "chunks" of (consecutive) tokens
  2) Assign the same position ID to the tokens at the same "significance"
     - Every token in each chunk is of a unique significance
     - We assign consecutive position IDs for consecutive tokens in each chunk
  3) At training time, randomly shift every position ID by a certain offset
     - Except for special tokens (BOS, EOS, PAD): fixed by '0'
     - Hyperparameter: Maximum possible position ID (`max_pos`)
  4) Apply Learned APE! 😉

Input Tokens	a	c	a	b	=	b	a	c	a	EOS
Position IDs	5	6	7	8	9	8	7	6	5	0

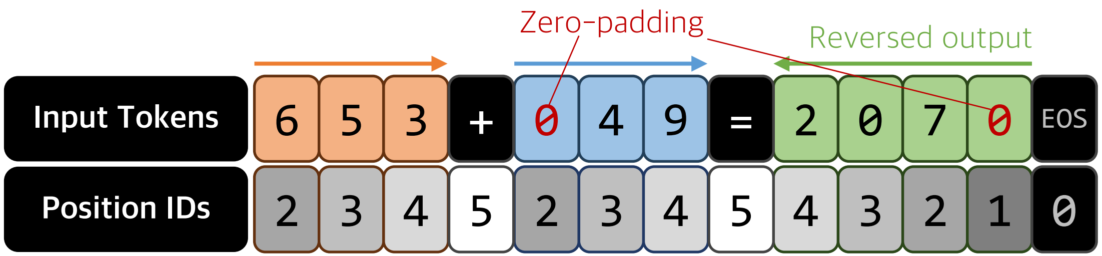
Method: Position Coupling (Addition task)

- **Position ID assignment rule** for each input sequence:
  1) Group input tokens into "chunks" of (consecutive) tokens
  2) Assign the same position ID to the tokens at the same "significance"
     - Every token in each chunk is of a unique significance
     - We assign consecutive position IDs for consecutive tokens in each chunk
  3) At training time, randomly shift every position ID by a certain offset
     - Except for special tokens (BOS, EOS, PAD): fixed by '0'
     - Hyperparameter: Maximum possible position ID (`max_pos`)
  4) Apply Learned APE! 😉



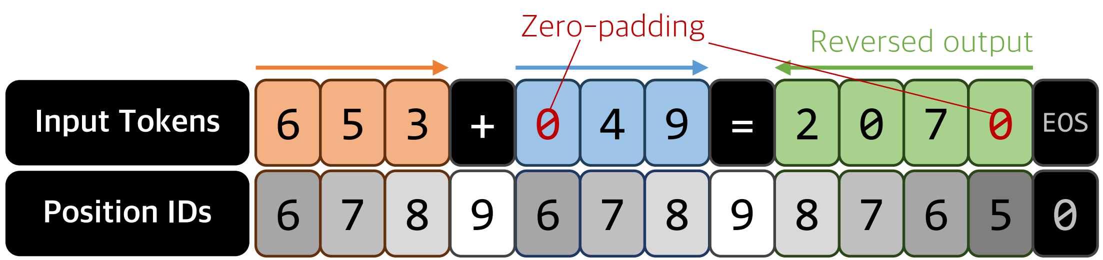
Method: Position Coupling (Addition task)

- Position ID assignment rule for each input sequence:
  1) Group input tokens into “chunks” of (consecutive) tokens
  2) Assign the same position ID to the tokens at the same “significance”
     - Every token in each chunk is of a unique significance
     - We assign consecutive position IDs for consecutive tokens in each chunk
  3) At training time, randomly shift every position ID by a certain offset
     - Except for special tokens (BOS, EOS, PAD): fixed by ‘0’
     - Hyperparameter: Maximum possible position ID (`max_pos`)
  4) Apply Learned APE! 😉



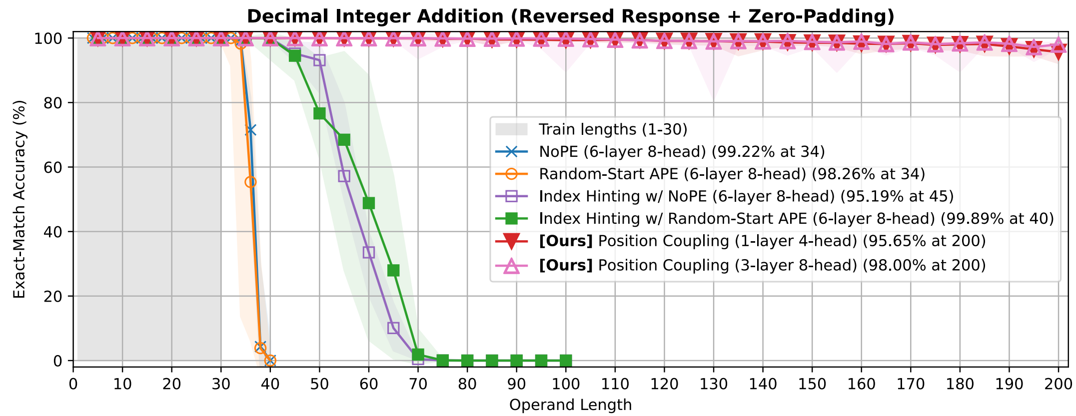
Method: Position Coupling (Addition task)

- Position ID assignment rule for each input sequence:
  1) Group input tokens into "chunks" of (consecutive) tokens
  2) Assign the same position ID to the tokens at the same "significance"
     - Every token in each chunk is of a unique significance
     - We assign consecutive position IDs for consecutive tokens in each chunk
  3) At training time, randomly shift every position ID by a certain offset
     - Except for special tokens (BOS, EOS, PAD): fixed by '0'
     - Hyperparameter: Maximum possible position ID (`max_pos`)
  4) Apply Learned APE! 😉

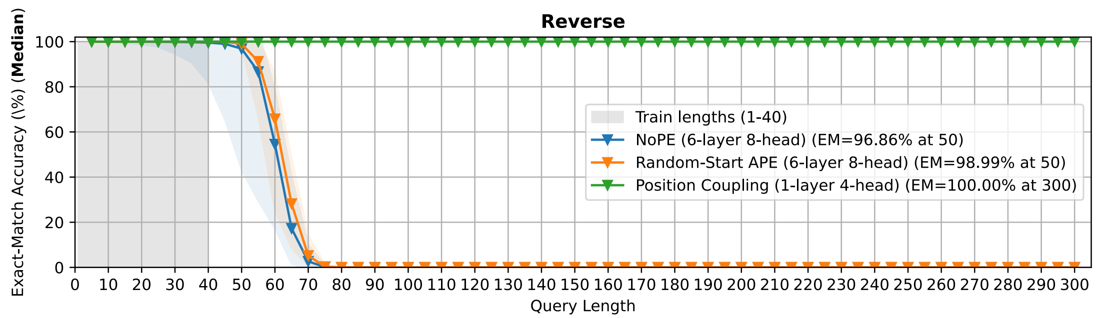


Experiments

- Addition Task

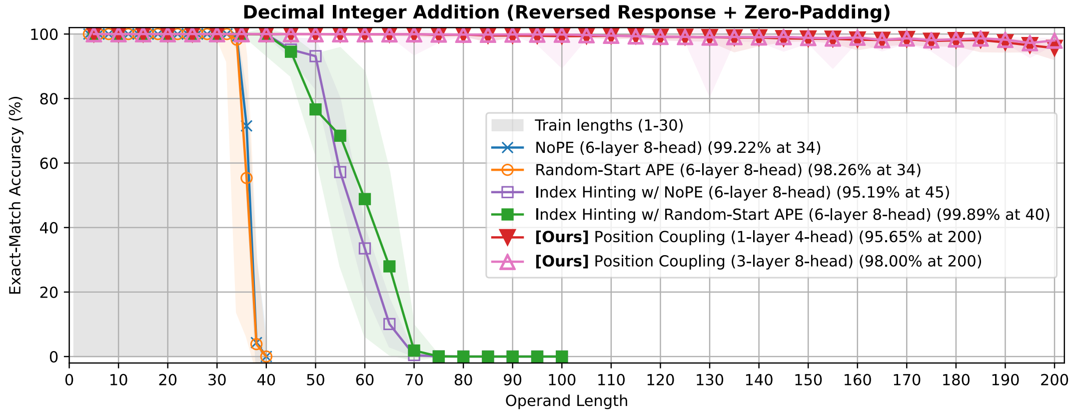


- Reverse Task (allowing duplicates)

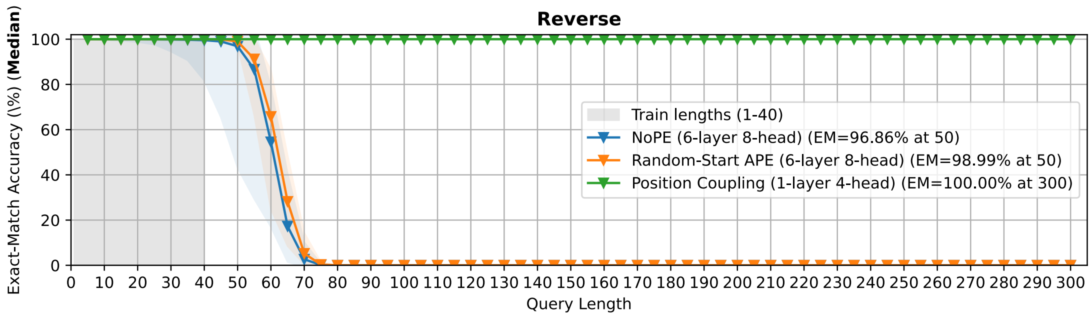


Experiments

- Addition Task



- Reverse Task (allowing duplicates)



- *Takeaway:*
  - **If you have any information about the task structure, use it!**
  - **It will lead a model to have a better inductive bias.**

Theoretical Analyses

- **Depth-1 Transformer + Position Coupling is sufficient to solve <u>exponentially</u> long additions entirely:**

> **Theorem 5.1.** There exists a 1-layer 2-head decoder-only Transformer with Position Coupling that solves the addition task. Here, the operand length is at most $2^{\mathcal{O}(d)}$, where d is the embedding dimension.

  - The proof is constructive.
  - In our construction, if $d = 512$, the maximum solvable length is $\approx 2.26\times10^{74}$.
  - Obviously extends to larger architectures with more layers & attention heads.

Theoretical Analyses

- **Depth-1 Transformer + Position Coupling is sufficient to solve exponentially long additions entirely:**

> **Theorem 5.1.** There exists a 1-layer 2-head decoder-only Transformer with Position Coupling that solves the addition task. Here, the operand length is at most $2^{\mathcal{O}(d)}$, where d is the embedding dimension.

  - The proof is constructive.
  - In our construction, if $d = 512$, the maximum solvable length is $\approx 2.26 \times 10^{74}$.
  - Obviously extends to larger architectures with more layers & attention heads.
  - In contrast, we prove that any depth-1 decoder-only Transformer without positional information (i.e., NoPE) cannot solve permutation-sensitive tasks (e.g., addition, multiplication, copy…) **(Proposition 5.2.)**

NeurIPS 2024

The 38th Annual Conference on Neural Information Processing Systems

Poster Session #6

Fri 13 Dec 4:30 p.m. PST — 7:30 p.m. PST

Check out our camera-ready version including:

- A striking similarity between our theoretical construction and actual trained Transformers
- **Ablations** on trained lengths, architectures, input formats, and more
- Results on **more tasks**, e.g., “Nx2” Multiplication, two-dimensional task (“minesweeper generator”)
- Comparison & Combination with **Rotary PE**

arXiv

arxiv.org/abs/2405.20671

GitHub

github.com/HanseulJo/position-coupling