# Data Augmentations for Arithmetic Length Generalization in Transformers

**Lynnix Zou, Muhammad H. Ashiq, Grigorios Chrysos**
*University of Wisconsin-Madison*

## Introduction

Transformers achieve impressive performance on many tasks, yet they often fail to generalize to longer input lengths even for basic arithmetic such as multi-digit addition. Existing methods [1, 2] usually modify the architecture, for example by introducing new position markers to positional embeddings.

We ask whether such machinery is actually required once a standard Transformer already fits the training distribution. We study whether a plain decoder-only Transformer can achieve strong length generalization on arithmetic purely through a data format.

## Methods

A central difficulty in arithmetic length generalization is identifying which digits should be operated on when sequence length grows. Transformers tend to retrieve digits by absolute index, so a local update rule does not automatically extend to unseen positions. Prior work [3] shows that the model learns a per-digit rule of the form:

$$y_i = (a_i + b_i + c_{i-1}) \bmod 10, \quad c_i = \left\lfloor \frac{a_i + b_i + c_{i-1}}{10} \right\rfloor$$

Our idea is to remove this bottleneck by forcing corresponding digits in the operands and the sum to share the same relative positions inside a longer padded layout. The Transformer can then reuse the same per-digit computation across all positions that become visible through padding.

## Results

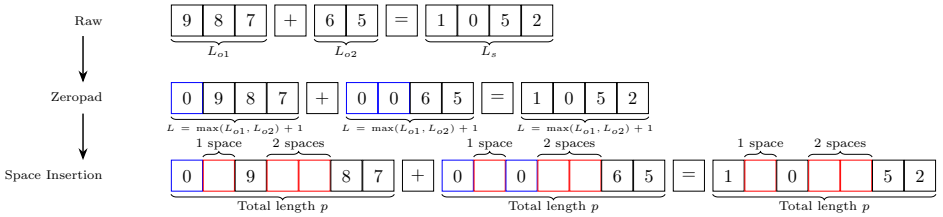


**Figure 1:** Illustration of the Aligned Blankspace Augmentation (ABA) method on addition

**ABA** is achieved by:

1. Zero-Padding: First, each numerical in the equations zero-padded to length L. This length L is determined to be sufficiently long to the expected maximum length of the sum.
2. Synchronized Space Insertion: An identical pattern of blank spaces is applied to expand the numbers to a target length, p. We first sample L unique positions from the p available slots uniformly at random without replacement, then insert The digits sequentially, with any unfilled positions becoming blank spaces.

## Results on addition

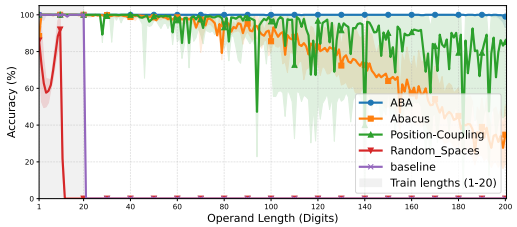


**Figure 2:** Methods for Length Generalization on Addition.

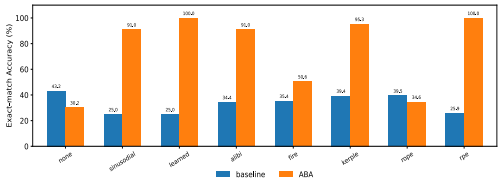


**Figure 3:** Different positional embeddings using ABA.

## Results

| Task | baseline | ABA | Abacus | PC |
|---|---|---|---|---|
| copy | 9.98 | 99.9 | 38.2 | 85.5 |
| reverse | 10.08 | 45.5 | 14.13 | 78.5 |
| multi_add | 4.84 | 69.23 | 6.35 | 41.5 |
| sort | 14.55 | 39.72 | 21.78 | 37.7 |
| N*2 multiply | 17.06 | 69.15 | 18.95 | 93.7 |

**Figure 3:** Methods for Length Generalization on multiple task.

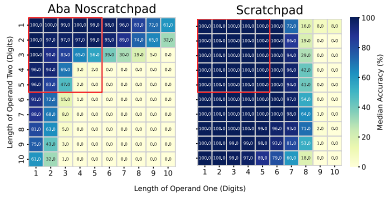


**Figure 4:** Performance comparison on multiplication.

## Conclusions

**Conclusion.** We show that a plain decoder-only Transformer can achieve strong length generalization on arithmetic tasks through a simple change in data format. Experiments on addition, copy, reverse, multi-operand addition, sorting, and N × 2 multiplication confirm that ABA consistently improves length generalization over standard formats. It also improves multiplication generalization with a scratchpad.

**Future work.** Future work includes applying ABA-style aligned formatting to broader classes of algorithmic and reasoning tasks where key tokens have a natural alignment structure. Another direction is to study how ABA interacts with larger pretrained language models by reformatting arithmetic or algorithmic prompts without retraining the backbone model.

## Reference

[1] Sean McLeish, Arpit Bansal et.al. Transformers can do arithmetic with the right embeddings, Nuerips 2024.

[2] Hanseul Cho, Jaeyoung Cha et.al. Position coupling: Improving length generalization of arithmetic transformers using task structure, Nuerips 2024.

[3] Nayoung Lee, Kartik Sreenivasan et.al. Teaching arithmetic to small transformers, ICLR 2024.