Learning Addition with a Small Transformer

Kaley Brauer
Center for Astrophysics
Harvard University
kaley_brauer@cfa.harvard.edu

https://github.com/kaleybrauer/transformer-addition

1 Task Setup

The goal of this project is to test whether a small causal Transformer can learn base-10 integer addition from examples, and to evaluate how (and whether) that behavior generalizes beyond the training distribution. I frame the task as causal next-token prediction: given a prompt encoding an addition problem, the model generates the sum.

Each input takes the form:

```
123 + 456 =
```

and the desired output is:

```
579
```

I focus on problems where both operands have exactly $k = 3$ digits ($a, b \in [100, 999]$). Note that the sum can be either 3 or 4 digits ($a + b \in [200, 1998]$), so the model must learn both the arithmetic and the correct output length.

Model. I train a GPT-style causal Transformer from scratch (randomly initialized weights), using a GPT-2-like configuration [3; 2] with:

- 4 Transformer layers
- hidden size 128
- 4 attention heads
- feedforward dimension 512

Tokenizer. I use the standard GPT-2 BPE tokenizer [2]. This avoids task-specific token engineering, but it also means digits may be represented by subword tokens rather than individual characters (e.g., some multi-digit strings may form single tokens). This choice may affect the degree to which solutions transfer across lengths.

Decoding and metric. I decode greedily and evaluate with exact-match accuracy after stripping leading/trailing whitespace. I cap generation at a small number of tokens to prevent the model from producing irrelevant extra text.

Codebase. The training scaffold (Hydra configuration, TRL SFT training loop, and checkpointing utilities) was adapted from the `trl_template` repository by Laura Ruis.[1]

[1] `https://github.com/LauraRuis/trl_template`

2 Data Design

2.1 Synthetic Data Generation

All data are generated synthetically. Each example samples two integers a and b independently and uniformly from $[10^{k-1}, 10^k - 1]$ and sets the target to the exact sum $a + b$.

In addition, I annotate each example with the number of digit-wise carry operations required to compute $a + b$. This enables diagnostic evaluation subsets targeting carry behavior.

2.2 Splits

I generate three datasets:

- Training set: 200,000 examples ($k = 3$)
- Validation set: 5,000 examples ($k = 3$)
- Test set: 5,000 examples ($k = 3$)

Each split is generated independently to avoid leakage.

2.3 Carry-Based Diagnostics

To probe what the model learns within the $k = 3$ regime, I construct additional test sets stratified by carry complexity:

- **No-carry**: problems requiring zero carry operations
- **Single-carry**: problems requiring one carry operation
- **Multi-carry**: problems requiring two or more carry operations

Each diagnostic set contains 2,000 examples and is used only for evaluation.

3 Training Details

I train the model from scratch using a standard causal language modeling loss (implemented with an SFT-style trainer wrapper). Hyperparameters are:

Parameter	Value
Optimizer	AdamW
Learning rate	3×10^{-4}
Batch size	64
Training epochs	10
Precision	FP16

Training was performed on a single NVIDIA T4 GPU in Google Colab and completed in about an hour in our main run (10 epochs).

4 Evaluation Suite

I evaluate with greedy decoding and exact-match accuracy. I report:

- **In-distribution (ID)**: $k = 3$ test set.
- **Carry-stratified tests**: no-carry, single-carry, and multi-carry subsets at $k = 3$.
- **Length generalization**: $k = 4$ addition (operands in $[1000, 9999]$) using the same prompt format, never seen during training.

The motivation is to separate improvements due to fitting the fixed-length training distribution from behavior that transfers to longer inputs.

5 Results

Figure 1 shows accuracy versus epoch for each evaluation slice.

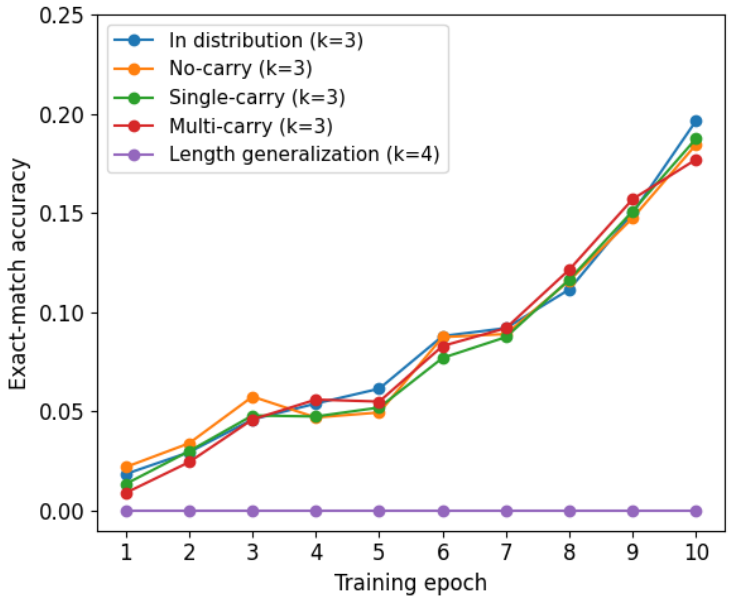

Figure 1: Exact-match accuracy versus training epoch. In-distribution and carry-stratified performance improve steadily with training, while length generalization ($k = 4$) remains at zero throughout.

At epoch 10, accuracies are:

Evaluation slice	Exact-match accuracy
In-distribution ($k = 3$)	0.1965
No-carry ($k = 3$)	0.1845
Single-carry ($k = 3$)	0.1875
Multi-carry ($k = 3$)	0.1770
Length generalization ($k = 4$)	0.0000

Notably, carry-stratified performance becomes similar by later epochs, indicating that within the fixed $k = 3$ regime, the number of carry operations is not strongly predictive of success once the model has trained longer. In contrast, performance on $k = 4$ remains at zero across all epochs.

6 Interpretation

For the first few epochs, multi-carry examples are modestly harder than no-carry examples, consistent with carry being an initial source of difficulty. With additional training, the gap between carry-stratified datasets largely disappears, suggesting the model learns to handle common carry patterns within the $k = 3$ distribution.

However, the complete lack of length generalization indicates that these improvements do not reflect a length-invariant addition procedure. Instead, the model appears to learn a length-specific approximation that improves on $k = 3$ without transferring to $k = 4$.

7 Discussion

I intentionally chose a simple prompt format and standard left-to-right generation. This is potentially misaligned with the classical right-to-left structure of long addition, where carry propagates from

least significant to most significant digits. Alternative formats (e.g., reversing digits) may better align generation order with carry propagation and may improve length generalization.

This split isolates a single generalization dimension: number length. The results show that strong improvements within $k = 3$ can coexist with zero transfer to $k = 4$, so in-distribution performance alone is insufficient evidence of algorithmic learning.

What remains uncertain is whether stronger inductive biases or curriculum learning [1] could induce length-invariant behavior in comparably small models. Future work would test (i) reversed-digit prompting, (ii) curriculum schedules that gradually increase k, and (iii) auxiliary supervision for carry states, and would evaluate whether any of these changes yield improved accuracy on $k = 4$.

Overall, these experiments illustrate that carry-stratified tests can diagnose in-distribution difficulty, while length generalization provides a stricter probe of whether the learned computation transfers beyond the training regime.

References

[1] Y. BENGIO, J. LOURADOUR, R. COLLOBERT, AND J. WESTON, *Curriculum learning*, in Proceedings of the 26th annual international conference on machine learning, 2009, pp. 41–48.

[2] A. RADFORD, J. WU, R. CHILD, D. LUAN, D. AMODEI, AND I. SUTSKEVER, *Language models are unsupervised multitask learners*, (2019).

[3] A. VASWANI, N. SHAZEER, N. PARMAR, J. USZKOREIT, L. JONES, A. N. GOMEZ, L. U. KAISER, AND I. POLOSUKHIN, *Attention is all you need*, in Advances in Neural Information Processing Systems, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, eds., vol. 30, Curran Associates, Inc., 2017.