# Can Chain-of-Thought Improve Algorithmic Reasoning in Small Language Models?

*An Empirical Study of Integer Addition with nanoGPT*

Rohan Piya | Dickinson College | COMP 560 Independent Research

Spring 2026

## Abstract

This paper investigates whether Chain-of-Thought (CoT) prompting improves the ability of a small GPT-style language model to learn integer addition from scratch. Using a custom nanoGPT implementation with approximately 0.79 million parameters, we train separate models on 2-, 3-, and 4-digit addition tasks in two formats: a plain format (e.g., 47+26=73) and a CoT format that explicitly shows column-by-column arithmetic reasoning (e.g., 47+26=6+7=13;4+2+1=7;73). We evaluate each model in-distribution and across digit counts to probe generalization. CoT models consistently reach 99.66-99.90% in-distribution accuracy across all digit counts, while plain models range from 61.60% to 96.37%. The improvement is most pronounced at 3-digit addition, where CoT closes a 38-point carry accuracy gap entirely. However, neither model type generalizes beyond its training digit count, and analysis of the CoT reasoning steps reveals that models learn a fixed-length template rather than a generalizable algorithm. Training loss curves confirm that CoT makes the optimization landscape fundamentally easier, independent of digit count.

## 1. Introduction

A central question in modern language model research is whether these models can learn to perform algorithmic reasoning — tasks that require structured, multi-step computation — or whether they are fundamentally pattern-matching systems that memorize input-output mappings. Integer addition provides an ideal testbed: the task is well-defined, the ground truth is always verifiable, difficulty scales predictably with digit count, and the hardest subcase (carry propagation) requires precisely the kind of state tracking that tests a model's reasoning capacity.

Chain-of-Thought prompting demonstrated that large language models can perform better on multi-step reasoning tasks when prompted to show their work. A natural follow-up question is whether CoT provides the same benefit to small models trained from scratch on a synthetic task — and if so, why.

This project is part of a broader research agenda with three stages: (1) build a robust addition model and clearly characterize its failure modes; (2) train with Chain-of-Thought and determine whether it improves performance and what resources it requires; (3) give the model scratch space to *develop its own* reasoning chain rather than following a prescribed format. This paper covers stages 1 and 2.

We train 8 models in total — CoT and plain variants at 2-, 3-, and 4-digit addition — using identical architecture and hyperparameters, varying only the data format. We evaluate each model

exhaustively at its training digit count and across digit counts to characterize generalization. Beyond standard accuracy metrics, we introduce a step-level diagnostic measure for CoT models that evaluates the correctness of each intermediate reasoning step, providing a richer picture of what the model has and has not learned.

Our main findings are: (1) CoT consistently improves in-distribution accuracy, with the largest gains on carry-heavy problems; (2) plain models show an unexpected non-monotonic accuracy pattern across digit counts; (3) neither model type generalizes to unseen digit counts; and (4) CoT models learn a **rigid template** — they fix the number of reasoning steps to match their training distribution, and cannot adapt this count for inputs of different lengths.

# 2. Methodology

## 2.1 Model Architecture

All experiments use a GPT-style decoder-only transformer implemented in nanoGPT (Karpathy, 2022). The architecture is kept identical across all 8 experiments to ensure that any performance differences are attributable to training data format rather than model capacity.

| Hyperparameter | Value |
|---|---|
| Parameters | 0.79M |
| Layers (n_layer) | 4 |
| Attention heads (n_head) | 4 |
| Embedding dim (n_embd) | 128 |
| Block size (context length) | 128 tokens |
| Dropout | 0.0 |
| Batch size | 32 |
| Max iterations | 3,000 |
| Learning rate | 1e-3 (cosine decay to 1e-4) |
| Hardware | MacBook Pro M3 (CPU) |

*Table 1. Shared model architecture and training configuration for all experiments.*

## 2.2 Data Format

Each training example consists of a randomly generated addition problem in one of two formats:

**Plain format:** `47+26=73\n`

**CoT format:** `47+26=6+7=13;4+2+1=7;73\n`

The CoT format explicitly decomposes the addition into column-by-column steps, starting from the ones column and proceeding to higher-order columns. Each step shows the digits being added along with any incoming carry, followed by the column sum. The final answer follows the last semicolon.

For 3-digit CoT:

```
473+261=3+1=4;7+6+0=13;4+2+1=7;734\n
```

For 4-digit CoT:

```
4731+2614=1+4=5;3+1+0=4;7+6+0=13;4+2+1=7;7345\n
```

## 2.3 Dataset Construction

For each digit count and format, we generate 100,000 training examples using a character-level tokenizer with a vocabulary of 13 tokens (digits 0-9, plus +, =, ;, and newline). Examples are split 90% train / 10% validation. Carry and non-carry examples are balanced at 50/50 to ensure the model is equally exposed to both cases. Earlier experiments with a 95/5 no-carry/carry split produced poor carry accuracy, motivating the balanced design.

## 2.4 Evaluation Protocol

For each trained model, we evaluate at multiple digit counts:

**Exhaustive evaluation:** For 1-digit (100 pairs) and 2-digit (8,100 pairs), all input combinations are tested.

**Sampled evaluation:** For 3-digit and higher, 10,000 random pairs are sampled.

Accuracy is reported overall and split by carry vs. no-carry status of the ground truth addition. The primary metric is strict exact-match accuracy on the final answer.

## 2.5 CoT Step Diagnostics

For CoT models, we introduce a secondary diagnostic metric that evaluates the correctness of individual reasoning steps. For each addition a+b, we compute the expected column sums (ones sum, tens sum, hundreds sum, etc.) and compare them to what the model produced in its intermediate steps. This yields three diagnostic measures: ones-column accuracy (did the model get the first step right?), average step accuracy (fraction of column steps that were correct), and all-steps accuracy (did every step match the ground truth?). These metrics are explicitly labeled as diagnostic and are not comparable to final answer accuracy.

# 3. Results

## 3.1 Training Losses

Table 2 shows final training and validation losses for all 8 models. Two clear patterns emerge.

| Model | Train Loss | Val Loss | Format |
|---|---|---|---|
| Plain 2-digit | 0.9752 | 0.9758 | Plain |
| CoT 2-digit | **0.4218** | **0.4228** | CoT |
| Plain 3-digit | 1.1229 | 1.1243 | Plain |
| CoT 3-digit | **0.4578** | **0.4578** | CoT |
| Plain 4-digit | 1.2204 | 1.2193 | Plain |
| CoT 4-digit | **0.4890** | **0.4877** | CoT |

*Table 2. Final training and validation losses for all 6 trained models.*

Plain model loss rises steadily with digit count: 0.98, 1.12, 1.22. Each additional digit makes the task meaningfully harder. CoT model loss stays nearly flat: 0.42, 0.46, 0.49. CoT essentially decouples task difficulty from optimization difficulty — the model always has an easy path to the answer through the step-by-step scaffold. The WandB training curves (Figure 1) show this visually: all three CoT models converge to a lower loss floor than any plain model, and they do so at a similar rate regardless of digit count.

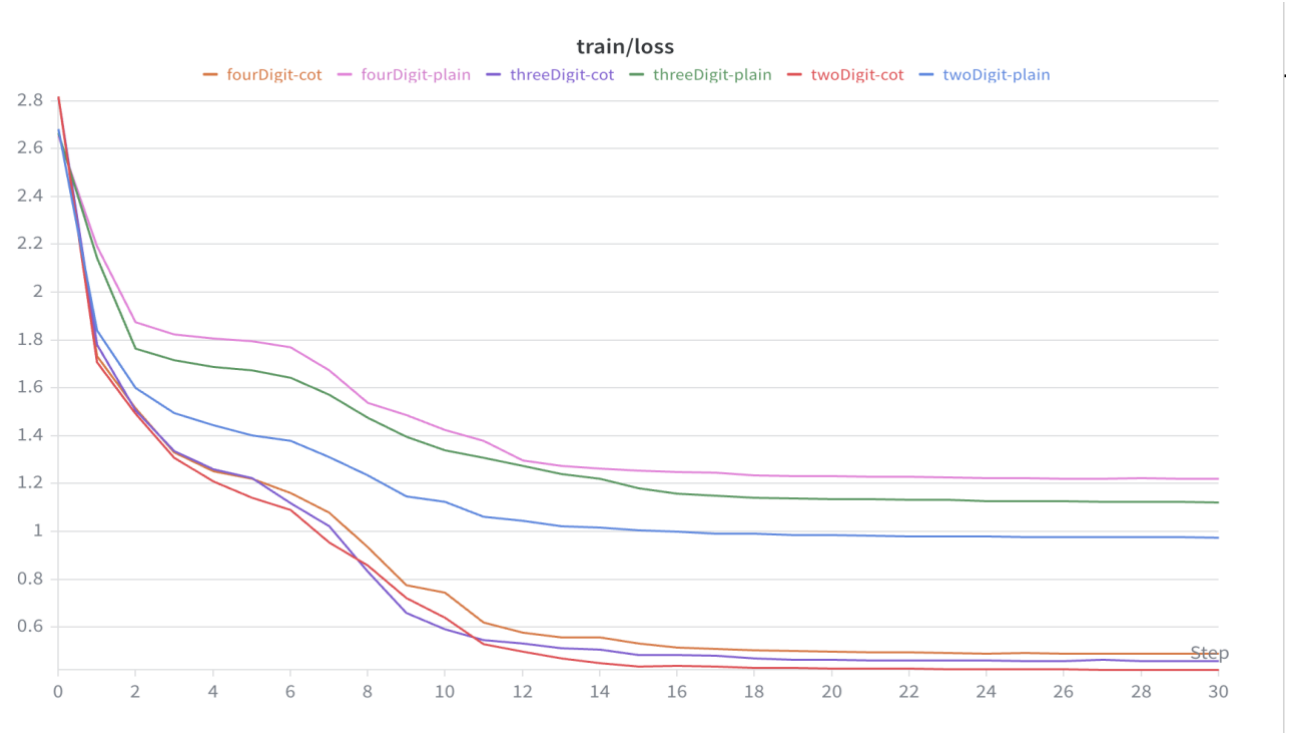


*Figure 1: WandB overlaid training loss curves for all 6 models. CoT models (lower cluster) converge to ~0.42-0.49. Plain models (upper cluster) converge to ~0.98-1.22, rising with digit count.*

## 3.2 In-Distribution Accuracy

Table 3 presents the full accuracy matrix across all models and evaluation digit counts.

| Model | Eval digits | Overall | No-carry | Carry |
|---|---|---|---|---|
| Plain 2-digit | 1 | 0.00% | 0.00% | 0.00% |
| Plain 2-digit | **2 ✓** | **96.37%** | 99.90% | 95.23% |
| Plain 2-digit | 3 | 0.00% | 0.00% | 0.00% |
| CoT 2-digit | 1 | 0.00% | 0.00% | 0.00% |
| CoT 2-digit | **2 ✓** | **99.90%** | 99.60% | **100.00%** |
| CoT 2-digit | 3 | 0.00% | 0.00% | 0.00% |
| Plain 3-digit | 1 | 0.00% | 0.00% | 0.00% |
| Plain 3-digit | 2 | 0.07% | 0.00% | 0.10% |
| Plain 3-digit | **3 ✓** | **61.60%** | 94.44% | 56.64% |
| CoT 3-digit | 1 | 4.00% | 0.00% | 8.89% |
| CoT 3-digit | 2 | 0.00% | 0.00% | 0.00% |
| CoT 3-digit | **3 ✓** | **99.71%** | 100.00% | **99.67%** |
| CoT 3-digit | 4 | 0.00% | 0.00% | 0.00% |
| Plain 4-digit | 1-3 | 0.00% | 0.00% | 0.00% |
| Plain 4-digit | **4 ✓** | **94.85%** | 94.30% | 94.90% |
| Plain 4-digit | 5 | 0.00% | 0.00% | 0.00% |
| CoT 4-digit | 1 | 24.00% | 0.00% | 53.33% |
| CoT 4-digit | 2 | 0.16% | 0.66% | 0.00% |
| CoT 4-digit | 3 | 0.02% | 0.00% | 0.02% |
| CoT 4-digit | **4 ✓** | **99.66%** | 100.00% | **99.63%** |
| CoT 4-digit | 5 | 0.00% | 0.00% | 0.00% |

*Table 3. Full accuracy matrix. ✓ marks in-distribution evaluations. Blue = plain models. Green = CoT models.*

## 3.3 CoT Step Diagnostics

| Model | Eval | Ones-col correct | Step accuracy | All steps |
|---|---|---|---|---|
| CoT 2-digit | 1-digit | 1.00% | 1.01% | 2.00% |
| CoT 2-digit | **2-digit ✓** | **99.90%** | **99.94%** | **99.90%** |
| CoT 2-digit | 3-digit | 18.98% | 8.35% | 0.00% |
| CoT 3-digit | 1-digit | 0.00% | 0.00% | 1.00% |
| CoT 3-digit | 2-digit | **99.88%** | 54.42% | 8.94% |
| CoT 3-digit | **3-digit ✓** | **99.91%** | **99.87%** | **99.71%** |
| CoT 3-digit | 4-digit | 4.90% | 2.25% | 0.00% |
| CoT 4-digit | 1-digit | 14.00% | 14.14% | 15.00% |
| CoT 4-digit | 2-digit | 21.32% | 14.28% | 2.00% |
| CoT 4-digit | 3-digit | **97.36%** | 38.47% | 0.85% |
| CoT 4-digit | **4-digit ✓** | **99.76%** | **99.91%** | **99.70%** |
| CoT 4-digit | 5-digit | 46.62% | 13.78% | 0.00% |

*Table 4. CoT step diagnostics. ✓ marks in-distribution evaluations. Bold values indicate notably high accuracy relative to neighbors in the same column.*

# 4. Analysis

## 4.1 CoT Improves In-Distribution Accuracy — Most Importantly on Carry

Across all three digit counts, CoT models outperform their plain counterparts in-distribution. The improvement is most striking at 3 digits, where plain accuracy is 61.60% and CoT reaches 99.71% — a gap of over 38 percentage points. At 2 digits the gap is smaller (96.37% vs. 99.90%), and at 4 digits smaller still (94.85% vs. 99.66%).

The carry accuracy split tells the clearest story. For the plain 3-digit model, no-carry accuracy is 94.44% while carry accuracy is only 56.64% — a gap of nearly 38 points. For the CoT 3-digit model, both are above 99.6%. CoT's contribution is not to make easy problems easier; no-carry accuracy is already high for plain models. CoT specifically addresses carry, because it gives the model an explicit slot to record the carry digit at each column step rather than tracking it implicitly. The plain model must maintain carry state across the entire sequence with no structural support; the CoT model never has to.

At 2 digits, the CoT model achieved 100.00% carry accuracy — every single carry problem in the exhaustive 8,100-pair evaluation was solved correctly. This is the strongest version of the carry finding and worth emphasizing.

## 4.2 An Unexpected Pattern in Plain Model Accuracy

A notable and not immediately obvious result is that plain model accuracy does not decrease monotonically with digit count. The plain 3-digit model achieves only 61.60% in-distribution accuracy, while the plain 4-digit model reaches 94.85% — significantly *higher* than the 3-digit model, despite the task being harder. This was unexpected. This result is worth investigating further in future work, potentially by running experiments at 5 and 6 digits to see whether the accuracy trend continues non-monotonically or converges downward.

It is also worth noting that the no-carry/carry split reverses at 4 digits: no-carry accuracy (94.30%) is slightly lower than carry accuracy (94.90%). Given that only 754 of the 10,000 evaluated pairs were no-carry, this is likely statistical noise at that sample size rather than a meaningful finding.

## 4.3 Neither Model Type Generalizes Across Digit Counts

Every out-of-distribution evaluation — across all models, both directions — produces near-zero accuracy. This is true whether the model is evaluated on smaller or larger digit inputs than it was trained on. The generalization failure is hard and consistent.

Plain models fail out-of-distribution in progressively more interesting ways as digit count increases. The 2-digit plain model truncates — it outputs 2-3 digit numbers regardless of input size, as if it learned the rule 'answers have 2-3 characters'. The 3-digit model hallucinates — it outputs constant values like 949, 949, 949 for all 1-digit inputs, suggesting it has learned 'outputs are 3-digit numbers' as an unconditional prior. The 4-digit model partially deconstructs the input — outputs like 6+10 and 8+13 suggest the model has learned some internal arithmetic structure but cannot produce a valid answer in the right format.

These escalating failure modes suggest that as models learn more complex input-output patterns, they develop richer but increasingly brittle internal representations that break in more structured ways under distribution shift.

### 4.4 CoT Models Learn a Rigid Template, Not a Generalizable Algorithm

The step diagnostics reveal a precise and consistent pattern. When a CoT model is evaluated one digit level below its training distribution, **ones-column accuracy stays near-perfect** but the model adds a spurious extra column step that does not exist. The CoT 3-digit model on 2-digit inputs achieves 99.88% ones-column accuracy but only 8.94% all-steps accuracy — because it always generates 3 column steps, inventing a hundreds column even when none is needed. The CoT 4-digit model on 3-digit inputs similarly achieves 97.36% ones-column accuracy but only 0.85% all-steps accuracy.

When a model is evaluated one digit level above its training distribution, the opposite occurs. The CoT 4-digit model on 5-digit inputs has only 46.62% ones-column accuracy and systematically drops the tens column, producing exactly 4 steps for a problem that requires 5. Across the 10,000-sample evaluation, the model consistently skips the tens column — not the highest-order column, not a random column, specifically the tens. This likely reflects positional biases in how the model reads digits from the input string.

The key insight is this: **the number of CoT steps is learned as a fixed property of the training format, not derived from the input.** The model learned 'this problem always has N steps' rather than 'count the columns and generate one step per column'. This is a fundamental limitation of the current training approach and directly motivates the third stage of this research project — giving the model scratch space to develop its own reasoning chain, potentially allowing it to determine the appropriate number of steps adaptively.

### 4.5 A Note on the CoT 4-Digit Model's 1-Digit Accuracy

The CoT 4-digit model achieves 24.00% accuracy on 1-digit inputs, which is the highest out-of-distribution accuracy of any model in this study. Closer inspection shows that all 24 correct answers come from carry cases (53.33% carry accuracy, 0% no-carry accuracy). This is not evidence of generalization. Single-digit carry sums (e.g., 7+8=15) produce answers in the range 10-18, which overlap with values the model has seen as intermediate column sums during training on 4-digit problems. The model is occasionally producing the right answer for the wrong reason — pattern-matching on a familiar intermediate value rather than computing the addition. This artifact disappears entirely for non-carry cases, confirming that it is not genuine generalization.

## 5. Conclusion

This study set out to characterize the failure modes of a small language model on integer addition and to determine whether Chain-of-Thought training can address those failures. The main conclusions are:

**1.** CoT consistently improves in-distribution accuracy at all digit counts tested. The improvement is most important on carry problems, where CoT provides a structural scaffold for tracking carry state that plain format lacks entirely.

**2.** Plain model accuracy is not monotonically decreasing with digit count — the 4-digit plain model outperforms the 3-digit plain model in-distribution. This unexpected result warrants further investigation.

**3.** Generalization across digit count boundaries fails completely for both model types, in both directions. These models learn distribution-specific mappings, not the addition algorithm.

**4.** CoT models learn the step count as a fixed template property. Step diagnostics show near-perfect ones-column accuracy even out-of-distribution, confirming that the digit-level arithmetic is learned — but the model cannot adapt the number of steps to match the actual input.

**5.** Training loss curves confirm that CoT makes optimization substantially easier and this advantage is consistent across digit counts, suggesting it is a property of the format rather than the specific task difficulty.

These results suggest that small models can learn arithmetic reasoning effectively within a CoT framework, but that the learned representation is brittle in a specific way: it encodes the reasoning structure as a fixed template rather than an adaptive procedure. Addressing this limitation is the central motivation for the next phase of this project.

## 6. Limitations

All experiments were run on a single hardware configuration (MacBook Pro M3, CPU only) with a fixed architecture of 0.79M parameters. It is possible that a larger model, more training iterations, or GPU acceleration would reveal different results. The 50/50 carry balance in training data was chosen to address a known weakness in prior results; the sensitivity of carry accuracy to this balance was not systematically studied. Evaluation at 3 digits and above uses random sampling (10,000 pairs) rather than exhaustive evaluation, introducing variance in the reported figures.

## 7. Future Work

The immediate next step is the third stage of the original research agenda: giving the model scratch space to develop its own chain of thought. Rather than prescribing the format of intermediate steps, this approach would train the model with access to an unstructured working memory, allowing it to develop its own internal reasoning structure. If successful, this might address the rigid template problem identified here — a model that generates its own reasoning format could in principle learn to produce the right number of steps adaptively.

Additional directions include: extending experiments to 5 and 6 digits to determine whether the non-monotonic plain accuracy trend continues; systematic study of how carry/no-carry data balance affects carry accuracy; analysis of whether the positional bias that causes the tens column to be dropped in out-of-distribution CoT can be corrected through data augmentation; and comparison of CoT with other structured prompting formats such as reversed-digit representations.