Part 4: Analysis and Report

This section details the analysis of the trained Transformer model for arithmetic tasks.

1. Quantitative Performance (Baseline Model)

The baseline model was configured with the following key hyperparameters: `N_LAYERS=3`, `D_MODEL=128`, `N_HEADS=8`, `D_FF=512`, `DROPOUT=0.1`. The performance was evaluated on the standard test set (`MAX_DIGITS_TEST = 4`).

Baseline Model Performance (N_LAYERS=3) on Standard Test Set:

- **Loss:** 0.032
- **Perplexity:** 1.032
- **Exact Match Accuracy:** 96.16%
- **Character-Level Accuracy:** 98.44%

These results indicate that the baseline Transformer model achieves high accuracy on arithmetic problems similar in length and complexity to those seen during training. A perplexity close to 1 suggests high confidence in the predicted sequences. The high Exact Match accuracy demonstrates the model's ability to generate the completely correct numerical answer string for the vast majority of test cases.

2. Generalization

To assess the model's ability to generalize to unseen problem structures, specifically longer number sequences, it was evaluated on a generalization test set (`MAX_DIGITS_GENERALIZATION = 6`).

Baseline Model Performance (N_LAYERS=3) on Generalization Test Set:

- **Loss:** 4.368
- **Perplexity:** 78.873
- **Exact Match Accuracy:** 45.56%
- **Character-Level Accuracy:** 48.04%

PROF

Discussion:
There is a **significant drop** in performance when the model encounters inputs longer than those seen during training.

- Exact Match Accuracy plummets from 96.16% to 45.56%.
- Character-Level Accuracy drops dramatically from 98.44% to 48.04%.
- Perplexity increases substantially from 1.03 to 78.87, indicating much higher uncertainty and poorer predictions.

This demonstrates that while the model performs well on in-distribution data, it struggles to generalize its learned arithmetic procedures to sequences longer than `MAX_DIGITS_TRAIN * 2 + 1`. The attention mechanism, while powerful, may face challenges in maintaining precise positional dependencies and numerical relationships over these longer, unseen sequences without specific

training examples or architectural enhancements designed for length generalization. The model appears to have learned patterns tied to the training data length rather than a truly abstract, length-invariant arithmetic algorithm.

3. Error Analysis (Baseline Model)

Examining incorrect predictions from the baseline model (N_LAYERS=3) on the standard test set provides insights into its weaknesses.

(a) Common Error Types:

Based on the sample errors:
Several patterns emerge:

- **Carrying/Borrowing Errors:** This appears to be the most frequent error type. Examples like `3516-540 -> 3976` (incorrect borrow in hundreds), `61+8950 -> 8011` (incorrect carry in thousands), `30+967 -> 1097` (incorrect carry in hundreds), and `1075-76 -> 1099` (incorrect borrow) suggest the model hasn't perfectly mastered these multi-digit procedures, which require tracking state across positions.
- **Off-by-One/Small Magnitude Errors:** Some errors are simple miscalculations, often differing by a small amount (e.g., `63-58 -> 9 | 5`, `18-17 -> 6 | 1`, `579-439 -> 130 | 140`).
- **Place Value/Alignment Errors:** Errors like `5802-6025 -> -1223 | -223` and `492-462 -> 120 | 30` might indicate difficulties aligning numbers correctly or handling place values during calculation, especially with negative results or subtractions involving different numbers of digits.

(b) Error Correlations:

- **Input Length:** Within the standard test set, length itself doesn't seem to be the primary driver of errors based on this sample. The errors appear across various lengths. However, the generalization results clearly show length is a major factor for *out-of-distribution* inputs.
- **Presence of Carries/Borrows:** Errors strongly correlate with operations requiring carrying (addition) or borrowing (subtraction). Problems solvable digit-by-digit without these interactions are likely easier for the model.
- **Specific Digits/Negative Signs:** While one error involved a negative result, it wasn't the primary issue in that case (magnitude was wrong). No obvious correlation with specific digits (like 0 or 9) is apparent from this limited sample, though a larger analysis might reveal subtle effects.

PROF

4. Ablation/Sensitivity Study

To understand the impact of specific design choices, we compare the baseline model to variations where one component is changed.

Baseline Configuration: `N_LAYERS=3`, with standard Positional Encoding.

Ablation 1: Reduced Number of Layers

- **Change:** Reduced the number of encoder and decoder layers from 3 to 1 (`N_LAYERS=1`), keeping all other hyperparameters identical.
- **Rationale:** To assess the impact of model depth/capacity on learning arithmetic.

- **Results:**

Metric	Baseline (N=3) Test	Ablation (N=1) Test	Baseline (N=3) Gen	Ablation (N=1) Gen
Loss	0.032	0.118	4.368	3.479
Perplexity	1.032	1.125	78.873	32.421
Exact Match Accuracy	**96.16%**	**88.94%**	**45.56%**	**42.06%**
Character-Level Accuracy	98.44%	95.70%	48.04%	47.58%

- **Analysis:**
  - Reducing layers from 3 to 1 caused a noticeable drop in performance on the standard test set (Exact Match decreased by ~7.2%), suggesting the additional layers in the baseline model provide useful representational capacity for learning the arithmetic task within the training distribution.
  - The simpler N=1 model still achieves a respectable ~89% exact match, indicating that even a shallow Transformer can learn basic arithmetic patterns to some extent.
  - On the generalization set, both models perform poorly, but the performance drop for N=1 compared to its own test set accuracy is slightly less drastic than for N=3. The N=1 model achieves lower (better) perplexity on the generalization set, perhaps because it's less overfitted to the specific length constraints of the training data or makes simpler (though still often incorrect) predictions on longer sequences. Nonetheless, neither model generalizes well.
  - Overall, model depth (N=3 vs N=1) contributes positively to performance on in-distribution data but doesn't solve the core length generalization problem.

Ablation 2: Increased Dropout Rate

PROF

- **Change:** Increased the dropout rate from 0.1 to 0.5 (`DROPOUT=0.5`), keeping `N_LAYERS=3`.
- **Rationale:** To assess the impact of much stronger regularization on model training and performance for this specific, somewhat structured task.
- **Results:**

Metric	Baseline (D=0.1) Test	Ablation (D=0.5) Test	Baseline (D=0.1) Gen	Ablation (D=0.5) Gen
Loss	0.032	1.679	4.368	3.284
Perplexity	1.032	5.360	78.873	26.691
Exact Match Accuracy	**96.16%**	**1.23%**	**45.56%**	**0.52%**

Metric	Baseline (D=0.1) Test	Ablation (D=0.5) Test	Baseline (D=0.1) Gen	Ablation (D=0.5) Gen
Character-Level Accuracy	98.44%	30.50%	48.04%	20.49%

- **Analysis:**
  - Increasing dropout significantly from 0.1 to 0.5 had a **catastrophic effect** on model performance. Exact Match Accuracy on the test set plummeted from 96.16% to a mere 1.23%. Character-level accuracy also collapsed.
  - This suggests that while some dropout (0.1) acts as useful regularization, preventing overfitting and likely contributing to the high baseline accuracy, excessive dropout (0.5) severely hinders the model's ability to learn the precise relationships required for arithmetic.
  - Arithmetic is a highly structured task where the relationships between adjacent tokens (digits) and across the sequence (e.g., corresponding place values, operator) are critical. A very high dropout rate likely disrupts the flow of information through the attention and feed-forward layers too much, preventing the model from consistently learning these precise dependencies. The model essentially becomes too "noisy" to learn the deterministic rules of arithmetic.
  - Interestingly, the loss and perplexity on the *generalization set* are lower (better) for the high-dropout model than the baseline, despite the abysmal accuracy. This might be an artifact of the high-dropout model producing much shorter or simpler (and thus less complex, lower perplexity on average) incorrect outputs, rather than indicating any true generalization benefit. The accuracy metrics clearly show it fails on both test and generalization sets.
  - This ablation highlights that hyperparameter tuning (like the dropout rate) is crucial, and values appropriate for one task (e.g., NLP tasks with more semantic ambiguity) might be detrimental to others requiring high precision like arithmetic.

5. Discussion

PROF

Learned Procedure:
The baseline model (`N_LAYERS=3`, `DROPOUT=0.1`) appears to have learned a procedure that is highly effective for arithmetic problems matching the training distribution's length constraints. The high accuracy (96%+) on the standard test set suggests it captures the core patterns of addition and subtraction well. However, the sharp decline in generalization performance and the prevalence of carrying/borrowing errors indicate that it hasn't learned a truly robust, abstract algorithm akin to human methods. It seems to rely heavily on patterns learned within a fixed sequence length. The ablation studies reinforce this: sufficient model capacity (N=3 layers) and appropriate regularization (dropout=0.1) are needed to learn these patterns effectively, but even then, generalization remains poor. Excessive regularization (dropout=0.5) prevents learning altogether.

Limitations:

- **Poor Length Generalization:** The most significant limitation, observed consistently across tested configurations.

- **Handling Complex Procedures:** Difficulty with multi-step carrying and borrowing remains a weakness, even for the best model.
- **Fixed Vocabulary & Input Format:** Standard seq2seq limitations apply.
- **Hyperparameter Sensitivity:** Performance is highly sensitive to choices like dropout rate and number of layers.
- **No Abstract Reasoning:** It manipulates character sequences based on learned correlations, not symbolic understanding.

Comparison to Human Computation:

- **Algorithm vs. Pattern Matching:** Humans use explicit, generalizable algorithms. The Transformer learns sequence-to-sequence mappings highly tuned to the training data distribution, particularly sequence length.
- **Processing:** Parallel (Transformer) vs. Sequential/Stateful (Human).
- **Generalization:** Humans generalize easily; the Transformer does not.
- **Regularization:** Concepts like dropout don't directly map to human learning, but the drastic failure with high dropout suggests the Transformer needs relatively stable internal representations to learn precise procedures, unlike tasks where noise might encourage robustness.

In conclusion, while the Transformer can be trained to perform basic arithmetic with high accuracy on in-distribution data using carefully tuned hyperparameters (like 3 layers and 0.1 dropout), it exhibits significant limitations in generalization and procedural robustness compared to human algorithmic understanding. Its success hinges on learning complex sequence patterns rather than a truly abstract computational procedure.