Teaching Transformers Arithmetic

Richard Yang

May 16, 2025

1 Introduction

Recently, large language models like GPT3/4 have shown impressive mathematical capabilities, despite not having been trained explicitly on such tasks. Understanding how these arithmetical abilities develop could shed light on how these models learn more general tasks.

In this report, we focus on the emergence of addition skills. We train a small (10.6M parameters) transformer on various datasets of addition problems and analyze the model's performance, aiming to replicate the results of [4]. We confirm its findings, and investigate the model's learning process in more detail.

2 Literature Review

Our work is based primarily on the work of [4]. Their abstract states:

> This study investigates how small transformers, trained from random initialization, can efficiently learn arithmetic operations such as addition, multiplication, and elementary functions like square root, using the next- token prediction objective. We first demonstrate that conventional training data is not the most effective for arithmetic learning, and simple formatting changes can significantly improve accuracy ... Building on prior work, we then train on chain-of-thought style data that includes intermediate step results ... this approach significantly and simultaneously improves accuracy, sample complexity, and convergence speed. We also study the interplay between arithmetic and text data during training and examine the effects of few-shot prompting, pretraining, and model scale. Additionally, we discuss length generalization challenges. Our work highlights the importance of high-quality, instructive data that considers the particular characteristics of the next-word prediction objective for rapidly eliciting arithmetic capabilities.

Some related work includes:

- Length Generalization in Arithmetic Transformers [2]. This work uses relative position embeddings to improve length generalization, as well as introducing "train set priming" for multiplication tasks, allowing generalization to much longer inputs.
- Arithmetic transformers can length generalize in both operand length and count [1]. This work achieves 2-3x length generalization on addition and multiplication by introducing task-specific scratchpads and multi-level position coupling.

- Teaching transformers modular arithmetic at scale [5]. Using angular embeddings and a custom loss function, this work enables transformeres to perform large scale modular arithmetic tasks, like summing 256 numbers moduluo 3329.

3 Methodology

3.1 Model Architecture

We start with a small transformer model, based off of NanoGPT [3]. For all runs, we use a 10.6M parameter model with 6 layers, 8 heads, and embedding dimension 384. The model is trained on an next token prediction objective.

3.2 Datasets

We have three datasets: one with 1-digit addition problems, one with 3 digit addition problems, and one with 3 digit addition problems where the sum is reversed. Each dataset consists of a set of addition problems, each represented as a string. Each line is a separate example. Datasets were generated randomly using a simple script. For single digit and three digit addition, each line is formatted plainly as

$$A_1A_2...A_i + B_1B_2...B_i = C_1C_2...C_j,$$

where A_i, B_i, and C_j are digits. For reversed three digit addition, we have left padded the operands with zeros. The result is right padded with zeros, due to being reversed. This ensures that each line has the same length.

3.3 Data Loading

We experiment with different data loading methods, to see their effect on the model's performance.

- Random block sampling. Strings of the block size length are sampled at random from the dataset.
- Random line sampling. Lines of the dataset are sampled at random. We have only used this for the reversed three digit addition dataset.

3.4 Model Evaluation

During training, we periodically evaluate the model on a validation set. For single digit addition, the validation set consists of all the 100 such problems. For three digit addition and reversed three digit addition, the validation set consists of 1000 problems, randomly sampled from the test dataset. The metrics we track are:

- Simple accuracy. The number of correct predictions divided by the total number of predictions.
- Character accuracy. The number of correct characters found in the correct position divided by the total number of characters predicted.

- Individual position accuracy. The number of correct characters in a given position divided by the total number of characters in that position.

Instructions for running the evaluation script are in the code repository. Logging was done using wandb. We are particularly interested in the character accuracy, and seeing if there is any structure in how the model learns to predict the digits.

4 Results

4.1 Single Digit Addition

Figure 1: Training loss.

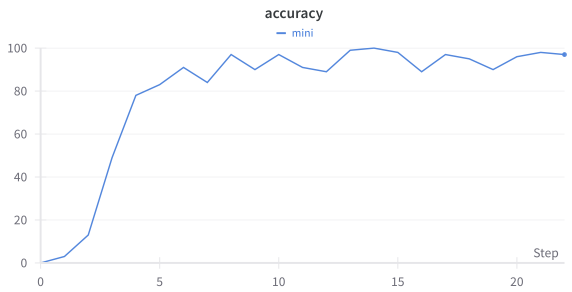

Figure 2: Accuracy.

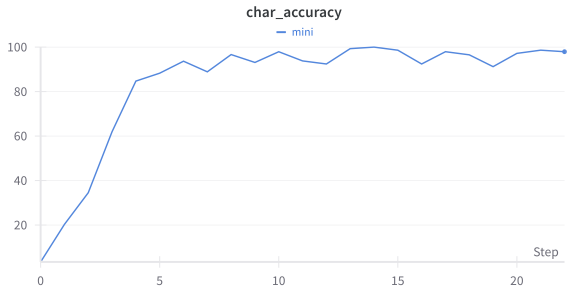

Figure 3: Character accuracy.

Figures for training loss, accuracy, and character accuracy are shown in Figures 1, 2, and 3 respectively. Each step is 50 iterations. It is apparent that accuracy increases

exponentially at first, before plateauing around 90%. The character accuracy slightly leads the accuracy, as expected since there are only one or two digits in the answer.

4.2 Three Digit Addition

Figure 4: Training loss.

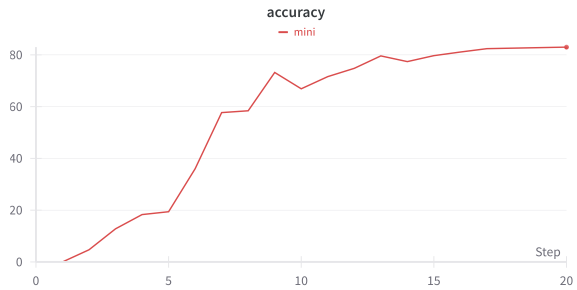

Figure 5: Accuracy.

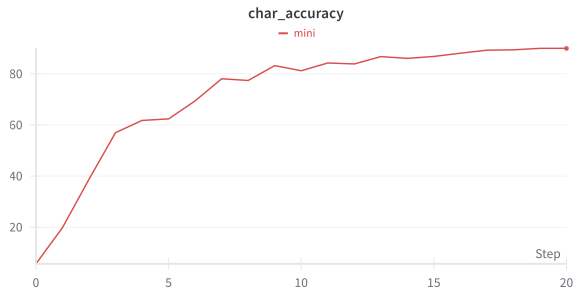

Figure 6: Character accuracy.

Figures for training loss, accuracy, and character accuracy are shown in Figures 4, 5, and 6 respectively. Each step is 250 iterations. We see the same exponential growth in accuracy, before a plateau around 80%, in line with the results of [4]. The character accuracy now significantly leads the accuracy, and plateaus around 90%. Notably, it has the opposite concavity. We suspect that if we evaluated the model more frequently near the beginning of training, we would see a similar exponential growth in character accuracy.

4.3 Reversed Three Digit Addition

4.3.1 Random Block Sampling

Contrary to the results of [4], we find that the model fails to learn the reversed three digit addition task using random block sampling. As shown in the figures, the training loss plateaus after a short while, along with the character accuracy. The accuracy stays extremely low at around 0.1%. We are not sure why this is the case, but suspect a data processing issue. Each step is 250 iterations.

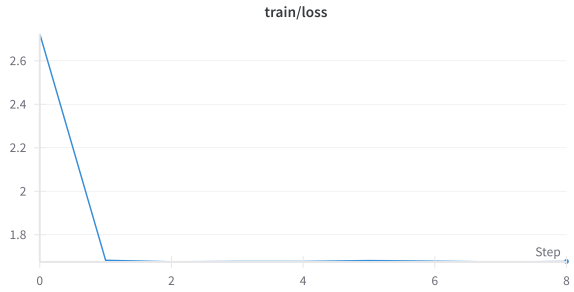

Figure 7: Training loss.

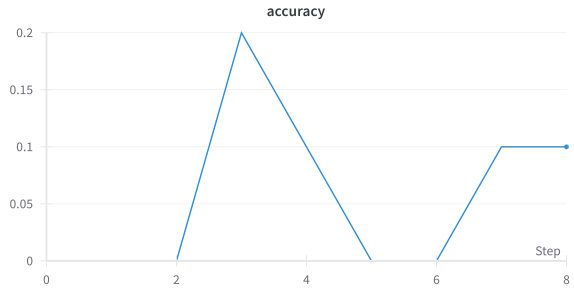

Figure 8: Accuracy.

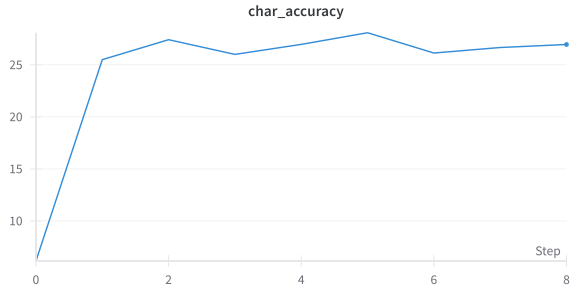

Figure 9: Character accuracy.

4.3.2 Random Line Sampling

However, using random line sampling, we find that the model is able to learn the reversed three digit addition task. In line with the results of [4], we find that the model is able to

learn the task, and achieves a character accuracy of around 90%. The accuracy increases significantly faster than plain three digit addition, and plateaus around 80%. We are unsure how much of this effect is due to the data loading method, and how much is due to the reasons outlined in [4]. Each step is 250 iterations.

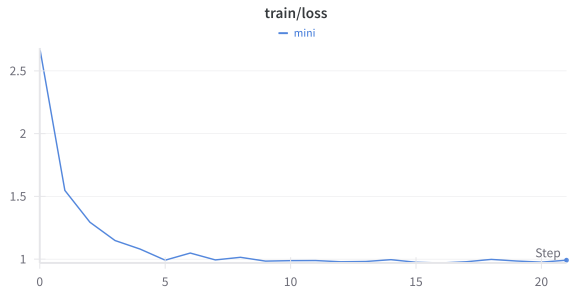

Figure 10: Training loss.

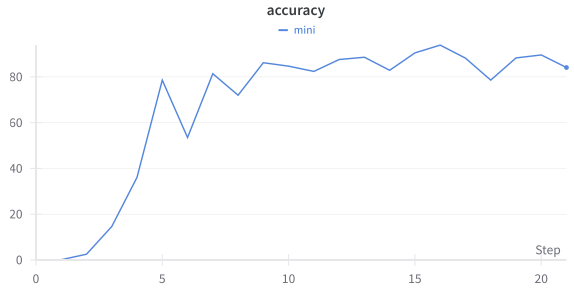

Figure 11: Accuracy.

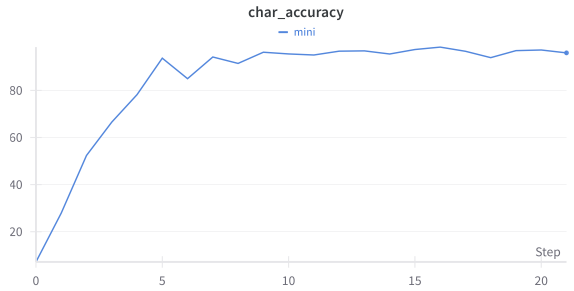

Figure 12: Character accuracy.

5 Conclusion

In this report, we have managed to mostly replicate the results of [4]. Possible future work includes:

- Investigating the model's performance on other arithmetic tasks, such as multiplication and division.

- Using random line sampling for the three digit addition dataset.
- Investigating the types of errors the model makes, such as whether they tend to mispredict digits by one.

References

[1] Hanseul Cho, Jaeyoung Cha, Srinadh Bhojanapalli, and Chulhee Yun. Arithmetic transformers can length-generalize in both operand length and count, 2024. arXiv:2410.15787.

[2] Samy Jelassi, Stéphane d'Ascoli, Carles Domingo-Enrich, Yuhuai Wu, Yuanzhi Li, and François Charton. Length generalization in arithmetic transformers, 2023. arXiv:2306.15400.

[3] Andrej Karpathy. Andrej karpathy's lightweight implementation of gpts, 2022. URL: https://github.com/karpathy/nanoGPT.

[4] Nayoung Lee, Kartik Sreenivasan, Jason D. Lee, Kangwook Lee, and Dimitris Papailiopoulos. Teaching arithmetic to small transformers, 2023. arXiv:2307.03381.

[5] Eshika Saxena, Alberto Alfarano, Emily Wenger, and Kristin Lauter. Teaching transformers modular arithmetic at scale, 2024. arXiv:2410.03569.