Supplementary Materials

A Additional Related Work

A.1 Chain Of Thought Reasoning

Chain of thought reasoning was introduced by Wei et al. (2022) as a method of improving performance in solving reasoning problems. Since then, this method has been expanded upon and found to improve performance across many domains (Wu et al., 2023; Zhang et al., 2023a; Feng et al., 2023; Yao et al., 2023; Lightman et al., 2023).

A.2 Self-Training

AlphaZero was one of the original works demonstrating the concept of self-learning, whereupon the training process employed Monte-Carlo Tree Search (MCTS) as a policy improvement operator. Specifically, AlphaZero used MCTS to improve upon the original policy of the game, before distilling this improved policy into the original policy. Coulom (2007) proved that, regardless of the quality of the original policy, running MCTS would provide a policy that was closer to the Nash equilibrium of the game. However, AlphaZero required access to a perfect simulator of the environment in order to perform MCTS. AlphaZero's followup work, MuZero (Schrittwieser et al., 2020), removed the requirement of direct access to a simulator by learning a model of the world, but MuZero still required continual query access to the true simulator to ensure that this world model remained accurate during training. While a simulator is easily accessible for such small, constrained environments, it is often not possible for general domains in the real world, where environments may be ill-defined or otherwise difficult to accurately simulate (e.g. real-world robotics, writing a Pulitzer Prize winning novel). It remains an open question whether an analogue policy improvement operator exists for a broader class of environments. Additionally, while MCTS is a powerful tool for learning in board games, it is not a general policy improvement operator. SECToR builds upon this prior work and can be used to train models to perform up to 30-digit addition without *any* queries to the ground truth world model after the initial self-training period.

B HYPERPARAMETERS

All models are fine-tuned using the Adam optimizer (Kingma & Ba, 2015) and the DeepSpeed library (Aminabadi et al., 2022) with a constant learning rate of 10^{-4}. We used a batch size of 2048 for our experiments with the 300M parameter model and 1024 for the 582M parameter model, which were the maximum possible sizes that fit in memory given our computational resources. Furthermore, we used 16-bit training via bfloat16 to conserve memory.

During the supervised training phase, when learning how to add 1 through N digits, we generated 10000 unique examples for N-digit chain-of-thought addition and 1000 unique examples for chain-of-thought addition for each digit from 1 - N to reduce catastrophic forgetting. For fast addition problems, we would generate 30000 unique examples for N digit addition and 3000 unique examples for all smaller numbers.

During the self-training phase, all numbers were reduced by a factor of 10, since it was substantially more costly to generate training data during this regime. If there are not enough unique training examples to satisfy these conditions[1], then we duplicate the problems until there are a minimum of $\frac{1}{3}$ of the size of the data otherwise required.

Models were allowed to proceed to $N + 1$ digit addition when they have achieved sufficient performance on 1 through N digit addition. Satisfactory performance is defined as achieving at least 75% accuracy length N addition problems without using chain-of-thought reasoning and 100% accuracy (on length N addition problems) when using chain-of-thought reasoning on a held out test set of size 128 of each type of problem. SECToR does not require perfect accuracy on fast addition problems because SECToR's built in self-consistency checks are more robust to errors of this kind than with errors in the chain-of-thought reasoning process.

[1] There are only 100 unique 1-digit addition problems.

C OUT OF BOX ACCURACY OF BYT5 MODELS ON ADDITION

To defend against the possibility that the ByT5 models secretly already know how to perform addition before SECToR, we evaluate the out-of-box accuracy of these models on simple addition. Table 2 shows that models have little-to-no ability to perform addition out of the box.

Model Size \ Addition Length	**1**	**2**	**3**	**4**
300M	0.015	0.004	0.0	0.0
582M	0.04	0.005	0.002	0.0
1.23B	0.0	0.005	0.0	0.0
3.74B	0.0	0.002	0.0	0.0

Table 2: Out-of-the-box addition accuracy (with no fine-tuning) of the ByT5 Xue et al. (2022) family of models. All accuracies are measured using 1000 randomly generated addition problems.

All models were prompted with 10 correct examples of addition with the specified number of digits before being asked to complete the 11th problem. An example prompt is pasted in its entirety below.

```
10 + 97 = 107
17 + 82 = 99
21 + 68 = 89
35 + 29 = 64
75 + 68 = 143
10 + 28 = 38
48 + 60 = 108
88 + 46 = 134
83 + 49 = 132
11 + 62 = 73
25 + 24 =
```

D RESULTS WITH THE 300M BYT5 MODEL

The 300M model began self training after 8 digits. The training run is described in Figure 11. Its last saved checkpoint was after successfully learning up to 24 digit addition problems, having failed to successfully learn continue the training loop with 25 digit addition. Figures 7 and 8 describe the generalization accuracies of the 300M parameter model over the course of training. Table 3 reports the addition accuracies achieved by the final version of the model. Note that even though the model was unable to continue training on the 25-digit iteration of learning, its previous checkpoint is still able to do addition problems larger than 24 digits with reasonable accuracy.

Length	1-19	20	21	22	23	24	25	26	27+
Accuracy (%)	98-100	95	98	98	99	98	91	33	0

Table 3: Accuracy (out of 100 examples) of the final checkpoint of the 300M model after training. For example, this table shows that the post-training 300M model can add 24 digit numbers with 98% accuracy without any chain-of-thought reasoning.

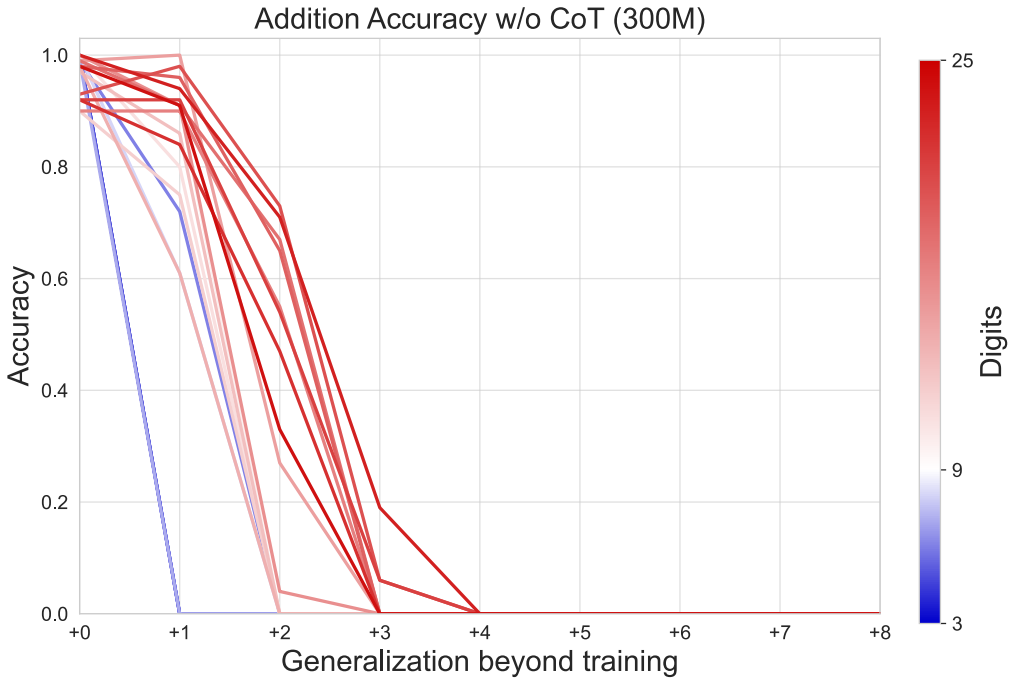


Figure 7: This figure describes the generalization accuracy of the model's addition capabilities over the course of training over the training run of the 300M model. Blue lines indicate the supervised training phase, while red lines indicate the self-training phase. We can see that even at the end of training, models do not show much generalization in their addition capabilities without using chain of thought.

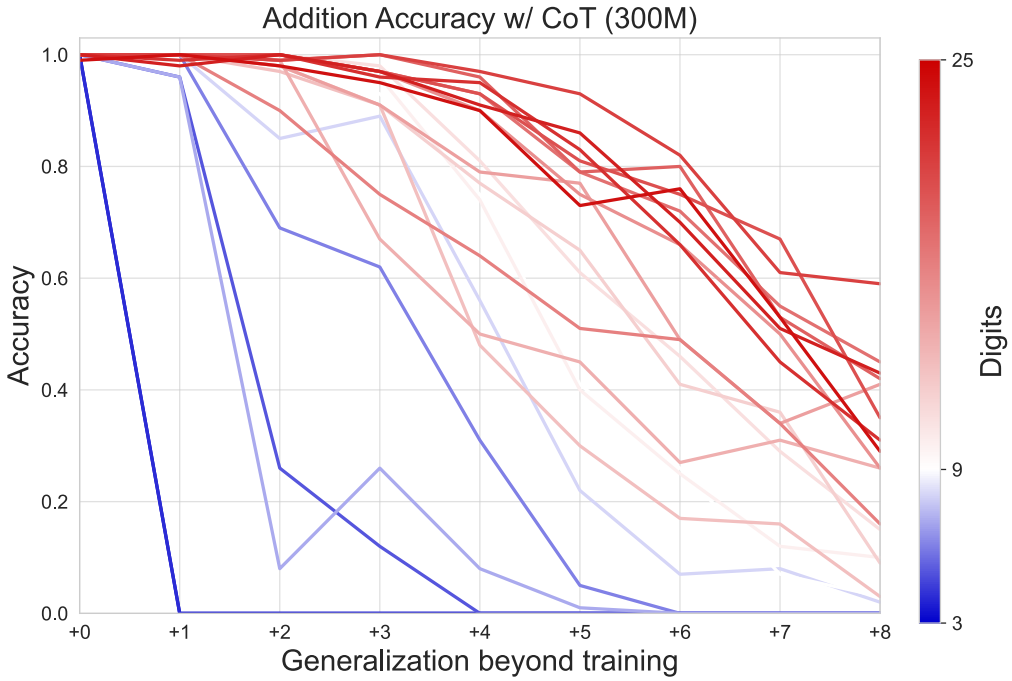
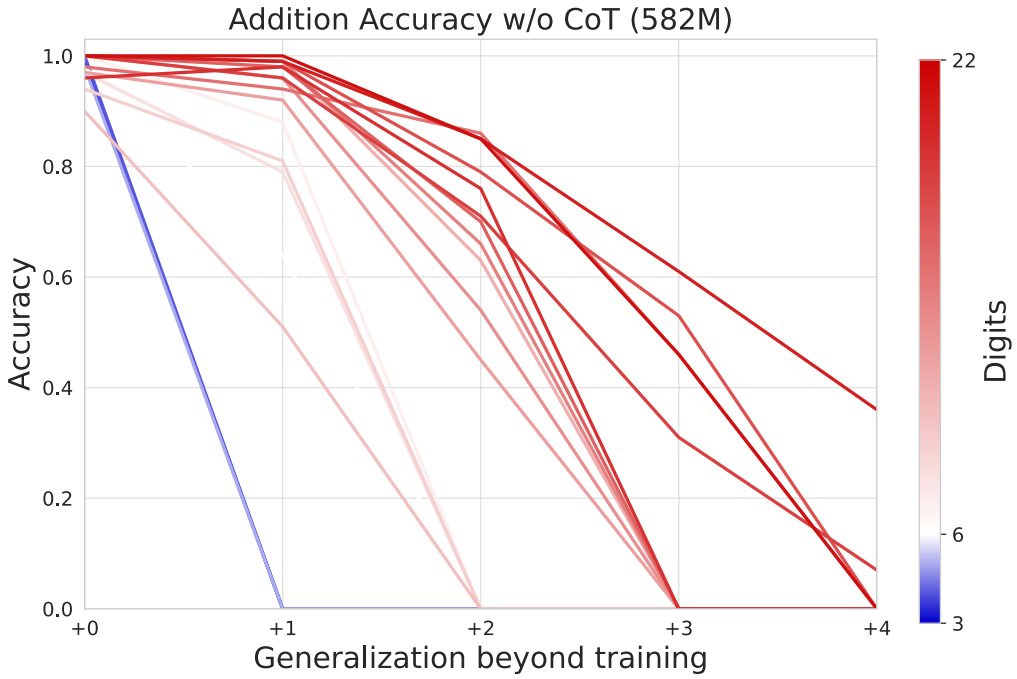
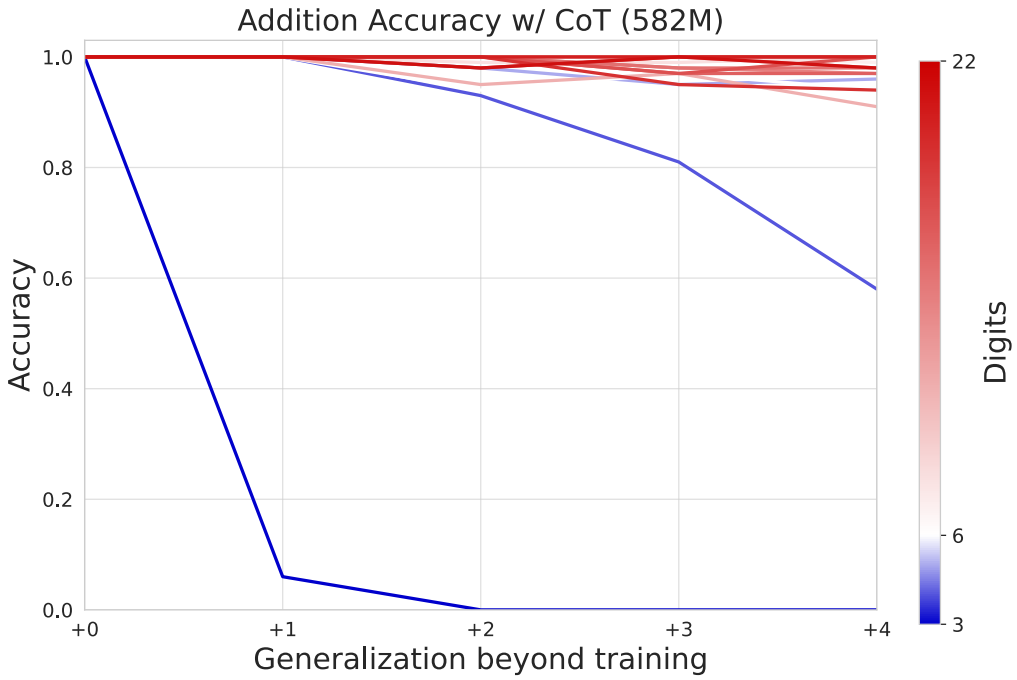


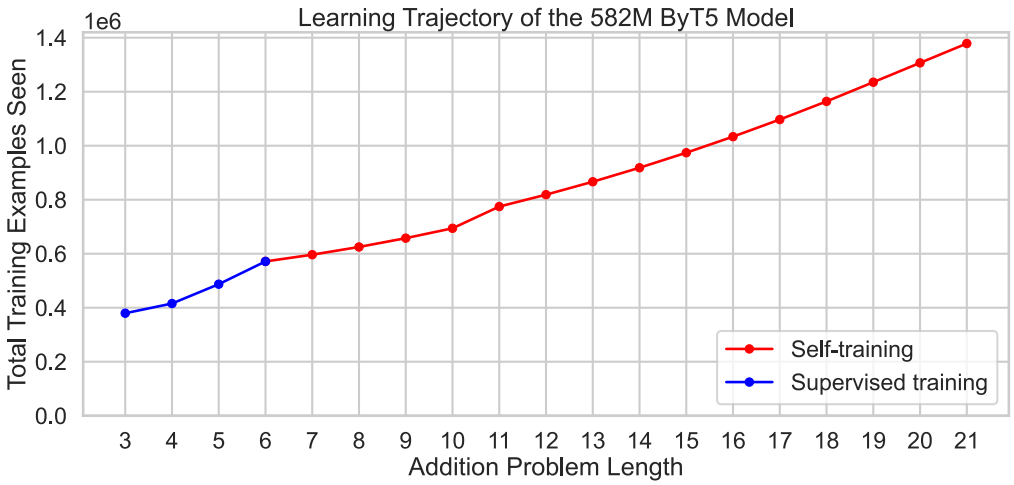
Figure 8: This figure describes the generalization accuracy of the model's addition capabilities over the course of training over the training run of the 300M model. Blue lines indicate the supervised training phase, while red lines indicate the self-training phase. While initially, models show little generalization to lengths greater than than what they have seen in training, models quickly learn to generalize well beyond their training distribution using chain-of-thought, allowing them to continue teaching themselves addition problems beyond what they have seen before.

E CONTROL EXPERIMENTS

A control experiment with the 582M parameter model had a supervised training phase of 1 through 5 digits and a self-learning phase of 6 through 21 digits. The training run is depicted in Figures E, E and E.






Learning Trajectory of the 582M ByT5 Model
1e6
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
Total Training Examples Seen
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
Addition Problem Length
Self-training
Supervised training

F SIMILARITIES BETWEEN ALPHAZERO'S MCTS AND SIMPLIFY-THEN-GUESS

Simplify-then-guess also has spiritual similarities to how AlphaZero often does not perform MCTS to the end of the game, instead terminating search after a certain depth and returning the output of the value network. The analogy for simplify-then-guess is that SECToR runs K steps of simplification (i.e. search) before taking a guess at the answer (i.e. querying the value network) instead of simplifying all the way to 1-digit addition (i.e. running MCTS to the end of the game). This takes direct advantage of the inductive curriculum-style training in which a model learns to add 1 to N digit numbers before being asked to generate training data for $N + 1$ addition numbers.

G FUTURE DIRECTIONS

G.1 SELF-LEARNING VS. LEARNING TO SELF-LEARN.

While SECToR is a process by which models teach themselves new concepts, they arguably do not *learn* to teach themselves new concepts. In our experiments, SECToR provides scaffolding around the model which, while never performing any aspect of addition, assists the language model in learning. For example, SECToR stops fine tuning once accuracy hits a certain threshold and this threshold was not chosen by the model itself. One can imagine using ideas from recent work, such as Toolformer, to give a language model access to tools or API commands that allow it to fine tune itself and add datapoints to the dataset. If successful, one could imagine a process analogous to SECToR, except that the process would consist solely of sampling from the model and executing the generated commands with no additional assistance. If so, one could imagine a model learning to teach itself a wide variety of concepts, including those in which it has to self-discover the learning process, as well as the new concepts itself.

G.2 GROUNDING.

Grounding is an area of active discussion in the research community, with many criticizing language models for their perceived lack of grounding in the real world. Some have suggested that future models may need to be embodied to effectively learn in the real world (Marcus, 2018; Bender & Koller, 2020; Tamari et al., 2020; Bisk et al., 2020). We believe our results with SECToR raise the possibility that large language models can succeed without grounding in domains that benefit from the existence of very strong self-consistency checking (e.g., mathematics, programming, etc.). While the model in the present paper is arguably grounded in true arithmetic during the supervised training phase, during the self-training phase, of which the majority of training occurs, models trained with SECToR receive no information from the external world and are, in this sense, ungrounded. Nevertheless, they manage to teach themselves addition problems that are orders of magnitude larger than they have ever seen during supervised fine-tuning. Might it be possible for models to bootstrap their learning in other domains without access to an incremental source of external signal or grounding?

H ERROR ANALYSIS

In this section, we examine what types of errors the models make on addition. We evaluate the final successful model checkpoint of the 582M parameter model on 30 digit addition. Note that as per Section 3.5, this is beyond what the model has ever seen during training, including self-training. Nevertheless, Table 1 suggests that models can generalize to perform such addition even without using chain-of-thought reasoning.

We plot the accuracy of the model on 30 digit addition against the number of carries required to perform such addition. Surprisingly, we notice little correlation between the number of carries a model must perform and its overall accuracy, suggesting that models are not simply learning to solve the "easy" addition problems.

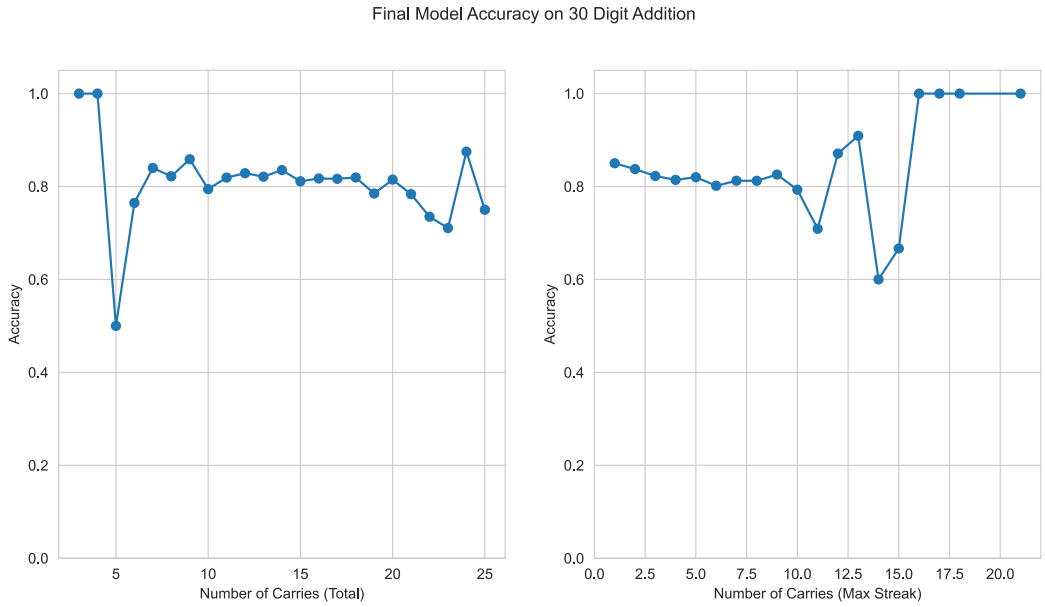


Figure 9: Model accuracy on 30 digit addition against the number of carries required. We observe little relationship between accuracy either the total number of carries required (left) or the longest streak of carries in a problem (right).

I EMERGENCE EXPERIMENTS

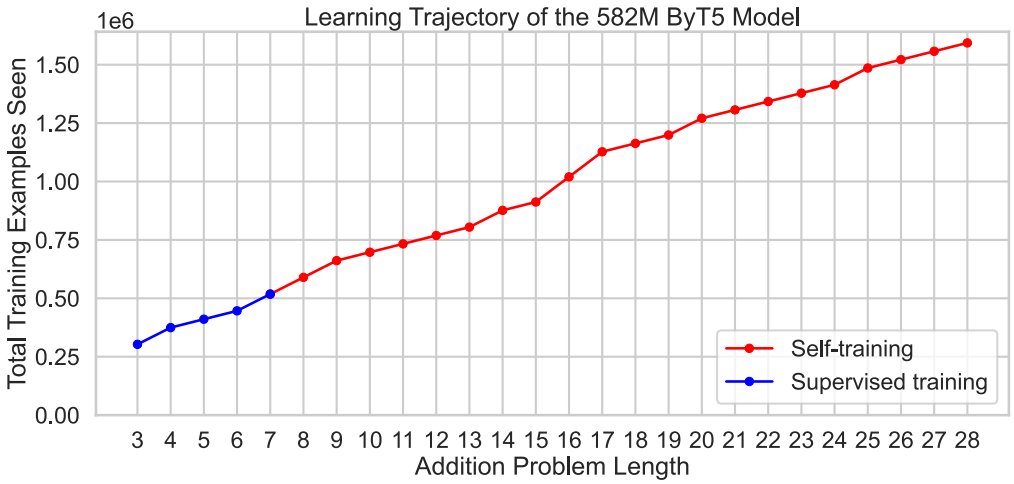


Figure 10: This figure describes how much training data was generated/consumed over the course of the training run of the 582M model.

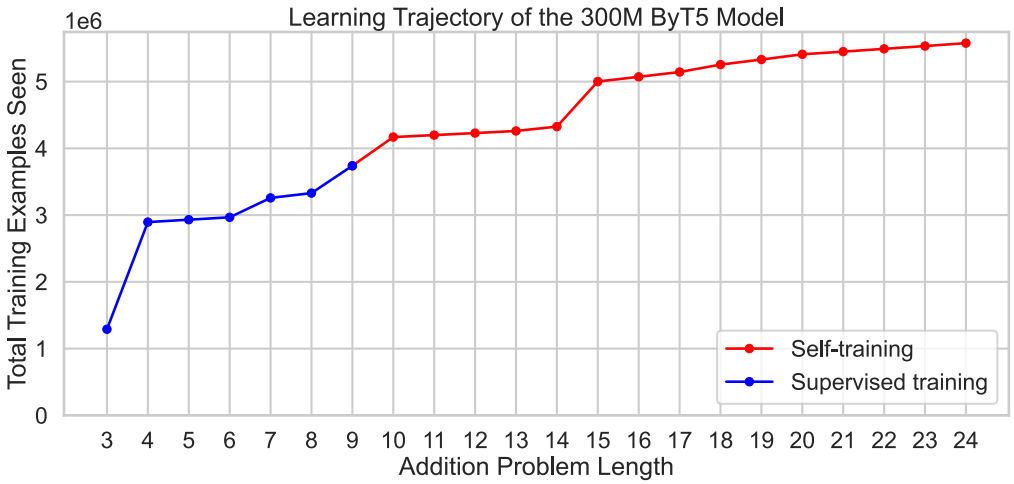


Figure 11: This figure describes how much training data was generated/consumed over the course of the training run of the 300M model.

While the 300M and 582M parameter ByT5 models required an initial supervised learning phase before being able to begin self-learning, we note that there seems to be an inverse correlation between the size of the model and the length of the supervised training period, both in terms of the maximum length of addition problems seen as well as the total training examples. For examples, Figure 11 shows that the 300M model required almost 4 million training examples of up to length 8 before generalizing sufficiently well to begin training. In contrast, Figure 10 shows that the 582M model required only 0.5 million training examples of up to length 6 before generalizing. We hypothesize that a sufficiently large model might be able to forgo the supervised training phase entirely and begin self-training immediately out of the box, possibly with the assistance of in-context learning.

Additionally, while simplify-then-guess outperforms generic "simplification" for generating new $N + 1$ digit examples (Figure 5), we find that "simplify + commutativity" rivals (and sometimes

outperforms) simplify-then-guess combined with commutativity in the 582M parameter model, even though it does not in the 300M parameter model. We speculate that this may be due to the errors in the simplification process being less correlated than errors in the simplify-then-guess process, but lead such speculation to future work.

J IMPORTANCE OF CURRICULUM LEARNING

A natural question is how important the curriculum learning where the model is required to successfully learning 1 through N digit addition before $N+1$ digit examples are added in the training step. We run an ablation where we train a 582M parameter ByT5 model on 1 through 6 digit addition in a single supervised fine-tuning step, instead of via the curriculum learning setup done in Section 3.5. We find that this model, when properly trained generalizes to up to 9 digit (slow) addition perfectly, suggesting that curriculum learning is the primary reason why SECToR is able to perform self-learning. Nevertheless, this ablation does not mean that that SECToR is able to run without some form of curriculum learning because a priori, one would have no way of knowing precisely what number of digits were sufficient to generalize other than empirical experiments. Additionally, during the self-training process, curriculum learning is essential by design, as one requires a model capable of performing 1 through N digit addition to generate the training data for $N+1$ digit addition.

K ADDITIONAL FIGURE ON SELF-LEARNING

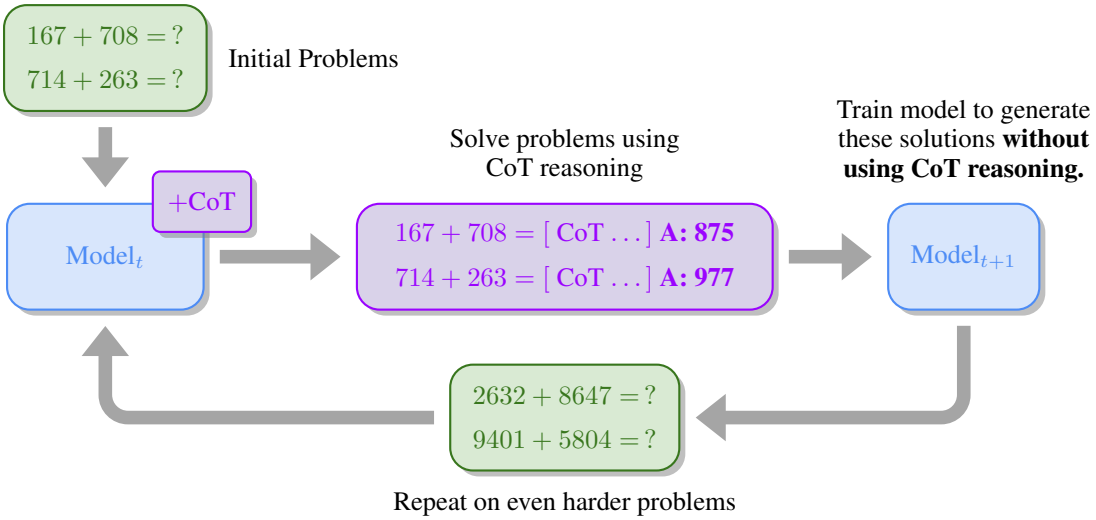


Figure 12: To perform self-learning, SECToR asks models to solve addition problems using chain-of-thought reasoning by decomposing the problem step-by-step. It then trains the next version of the model to solve those same problems directly *without using chain-of-thought reasoning*. This process often results in an improved model which can often solve even harder problems than original model, allowing the self-learning loop to continue.