
Semantic Soft Bootstrapping: Long Context Reasoning in LLMs without Reinforcement Learning

Purbesh Mitra
 University of Maryland
 pmitra@umd.edu

Sennur Ulukus
 University of Maryland
 ulukus@umd.edu

Abstract

Long context reasoning in large language models (LLMs) has demonstrated enhancement of their cognitive capabilities via chain-of-thought (CoT) inference. Training such models is usually done via reinforcement learning with verifiable rewards (RLVR) in reasoning based problems, like math and programming. However, RLVR is limited by several bottlenecks, such as, lack of dense reward, and inadequate sample efficiency. As a result, it requires significant compute resources in post-training phase. To overcome these limitations, in this work, we propose **Semantic Soft Bootstrapping (SSB)**, a self-distillation technique, in which the same base language model plays the role of both teacher and student, but receives different semantic contexts about the correctness of its outcome at training time. The model is first prompted with a math problem and several rollouts are generated. From them, the correct and most common incorrect response are filtered, and then provided to the model in context to produce a more robust, step-by-step explanation with a verified final answer. This pipeline automatically curates a paired teacher-student training set from raw problem-answer data, without any human intervention. This generation process also produces a sequence of logits, which is what the student model tries to match in the training phase just from the bare question alone. We set up an experiment to train the open-source model Qwen2.5-3B-Instruct on GSM8K dataset via parameter-efficient fine-tuning. We then tested its accuracy on MATH500, and AIME2024 benchmarks. Our experiments show a jump of 10.6%, and 10% improvements in accuracy, respectively, over group relative policy optimization (GRPO), which is a commonly used RLVR algorithm. Our code is available at <https://github.com/purbeshmitra/semantic-soft-bootstrapping>, and the model, curated dataset is available at <https://huggingface.co/purbeshmitra/semantic-soft-bootstrapping>.

1 Introduction

Recently, long context reasoning in large language models (LLMs) have shown enormous progress in accelerating different aspects of scientific and engineering tasks. The most notable examples are the rise of LLM assisted programming (Watanabe et al., 2025; Ge et al., 2025), achieving top human level performance in math examinations like IMO gold medal (Huang and Yang, 2025), human level performance at ARC-AGI pattern recognition test (Chollet et al., 2024), and so on. Very recently, there have been several scientific breakthroughs facilitated by LLM based systems (Bubeck et al., 2025; Rizvi et al., 2025). Behind all such reasoning LLMs, is the post-training paradigm of reinforcement learning with verifiable rewards (RLVR). Starting from the open-source implementation of group relative policy of optimization (GRPO) (Shao et al., 2024) in Deepseek-R1 model (Guo et al., 2025), numerous such optimizing techniques are being used for RL training of LLMs Liu et al. (2025); Yu et al. (2025); Zheng et al. (2025). The basic idea of RLVR is letting the LLM generate rollouts on a

multitude of reasoning questions with verifiable answers. The reinforcement learning algorithm then rewards the LLM weights more, which has the higher probability of generating correct answers to the questions. As a result, the chain-of-thought (CoT) of the LLM becomes longer and sophisticated reasoning pattern emerges in the thought traces.

Even though, RLVR has delivered quite a lot of success in the advancement of LLM reasoning capabilities, there are several bottlenecks and limitations of such approaches. For example, in the outcome-based reward formulation, the reward in RLVR is not dense, i.e., each trajectory of an LLM response gets a coarse reward at the end of the answer. On top of that, the RL algorithm takes an average of those rewards for estimating the success probability of the LLM. This leads to the LLM not being able to fully grasp the underlying reasoning in different tasks. If an LLM generates two responses, one of which is completely wrong logic from the beginning, and the other one has a “silly mistake” at the very end of the response, both of them are assigned the same low reward. On the other hand, if a response has a wrong reasoning in it, but still it manages to somehow generate the correct answer, it gets incentivized to follow such reasoning again. To combat this, if we employ some kind of process supervision of the LLM reasoning traces, it is difficult to assign partial rewards to the logical steps without any possibility of reward hacking. Furthermore, in some recent studies (Yue et al., 2025; Wu and Choi, 2025), it has been shown that the fundamental reasoning capabilities of the LLM do not increase with RLVR. Rather, the base model already has some reasoning capabilities, which are then amplified into the pass@1 accuracy of the RL trained model from pass@k, thus, making its capabilities even narrower.

These issues are largely underexplored topics, and only a handful of ideas have been proposed as alternatives. Agrawal et al. (2025) introduce a reflective prompt optimizer that combines natural-language self-analysis with multi-objective evolutionary search over prompts. This method outperforms GRPO and prior prompt optimizers while using up to 35 times fewer rollouts across diverse reasoning and tool-use tasks. Karan and Du (2025) show that a training-, dataset-, and verifier-free iterative sampling algorithm targeting a sharpened power distribution defined by base-model likelihoods can elicit reasoning capabilities from base models that can match GRPO post-trained models on benchmarks, such as, MATH500, HumanEval, GPQA, and AlpacaEval, while avoiding the diversity-collapse typical of RLVR post-training. Feedback Descent by Lee et al. (2025) is an inference-time framework that optimizes text artifacts, such as, prompts, code using pairwise comparisons augmented with rich textual feedback instead of scalar rewards, thus, treating feedback as a gradient-like signal in text space. Prior self-training work on LLM reasoning, such as STaR (Zelikman et al., 2022), which iteratively fine-tunes on self-generated rationales that lead to correct answers, Think–Prune–Train–Improve (Costello et al., 2025), which repeatedly fine-tunes on pruned sets of correct CoT traces, and BOLT (Pang et al., 2025), which bootstraps long chains-of-thought via supervised finetuning on self-synthesized LongCoT data and online RL phase, all rely on standard cross-entropy loss for next-token prediction over textual CoT sequences, often from a teacher model.

All the approaches so far do not take advantage of the in-context learning (Dong et al., 2024) ability of the LLM itself. RLVR training does the exploration via different rollouts, but incentivizes reward based on the average of individual rollouts. As a result, the LLM does not receive a clear signal about the accuracy of the produced trajectories. Our approach considers using different sample responses by presenting them to the LLM in the context of correctness in turn making better responses. This is similar to the well-studied idea of mixture-of-agents (Wang et al., 2024). Another important aspect is the issue of scalability, which boils down to the idea of bitter lesson (Sutton, 2019), i.e., simple algorithms that take advantage of computation outperform other complicated methods. This is one of the reasons why the self-supervised learning (SSL) method (Balestrieri et al., 2023) in pre-training scales so well, since simply minimizing the loss function for next-token prediction task provides signal in each token across the whole generation trajectory, thus, scaling with compute power. For this reason, we believe it is necessary to find a compromise between the exploration phase of RL and the signal exploitation in SSL. To the best of our knowledge, there has not been any work that combines RL and SSL in such a manner.

To that regard, we introduce Semantic Soft Bootstrapping (SSB), a self-distillation method that teaches a language model to solve reasoning problems without hints by leveraging its own hinted reasoning as a teacher. We start from a supervised dataset of problems with ground-truth final answers, and first query a base model on the question alone to generate multiple rollouts. These are then partitioned into correct and incorrect attempts using the boxed final answer format. From each problem, we then construct a prompt that includes the original problem, one representative

correct solution trace, and one trace with the most common incorrect final answer. The same base model, now used as a teacher under a robust “refine and explain” system prompt, produces a single, detailed, corrected solution that passes the final answer check. This can be interpreted as an *in-context contrastive learning* from negative pairs, as opposed to the traditional contrastive learning Le-Khac et al. (2020). This process yields paired problem-solution data points without human effort. We also extract the teacher model’s token-level logits for the answer portion only, and store them as soft labels. During training, a student model (same base model with LoRA adapters) is given only the question, and is optimized to match the teacher’s token distribution on the answer tokens via a KL-based distillation loss, optionally combined with cross-entropy, without any reward model or policy gradient. As a result, the student learns to reproduce robust, step-by-step reasoning and correct final answers from hint-free prompts, while all bootstrapping signal is provided through logit-level self-supervision; without an explicit reinforcement learning algorithm. Additionally, since the student model tries to match verifiably correct response trajectories via matching logits, this method eliminates any chances of reward hacking, which is a considerable issue in RL training.

Furthermore, since the model is not trained on hard token prediction, but instead minimizing a KL loss with different in-context prompts, we are essentially nudging the model output probability minimally towards the distribution of correct responses. This strictly follows from the fact that KL divergence quantifies the minimum shift of an estimated (student’s) probability distribution, to the true (teacher’s) probability distribution, hence, retaining the model’s generalization capability while training. Since we are not letting the model train on its own generated response, this avoids a collapse in performance of the model Shumailov et al. (2024). Our experiments show that SSB outperforms GRPO in MATH500, and AIME2024 benchmarks by a margin of 10.6%, and 10%, respectively. This training was done on a curated set of 256 samples from the GSM8K question-final-answer dataset.

Our contributions are summarized as follows:

- We propose Semantic Soft Bootstrapping (SSB), an RL-free self-distillation framework that converts a single base LLM into both teacher and student by exposing them to different views of the same problem (hinted vs. hint-free).
- We introduce a semantic refinement stage where the teacher is prompted with both correct and incorrect student-like rollouts and must synthesize a single, correct, robust explanation.
- We construct a paired teacher–student dataset in which the teacher receives rich hinted context while the student sees only the raw question, enabling learning to “solve without hints” from self-generated supervision.
- We perform logit extraction on the teacher’s robust solutions and design a custom data pipeline that distills only the answer tokens, aligning student predictions with teacher soft labels at the level of token distributions.
- SSB shows a gain of 10.6%, and 10% improvements in accuracy in MATH500, and AIME2024 benchmarks, respectively, over GRPO, while being trained on a set of 256 curated samples from the GSM8K question-final-answer dataset.

2 Background and Related Works

DeepSeek-R1 (Guo et al., 2025) proposed a pure RLVR approach to improve the reasoning capabilities of LLMs, introducing DeepSeek-R1-Zero and DeepSeek-R1 trained with GRPO (Shao et al., 2024) on reasoning questions. By repeatedly sampling chains-of-thought and rewarding correct or high-quality reasoning trajectories, DeepSeek-R1 shows that long, structured reasoning patterns can emerge from an RL signal alone, and that RL can substantially boost pass@k on math and programming tasks.

Distilling knowledge in a neural network (Hinton et al., 2015) is the classical formulation of knowledge distillation, where a large teacher model is compressed into a smaller student model by training the student to match the teacher’s output distribution. The key idea is that soft targets at elevated temperature carry rich information about class similarities, allowing the student to inherit the teacher’s generalization behavior. This approach was shown to improve performance on vision and speech recognition tasks while reducing model size and inference cost.

On-policy distillation of language models (Agarwal et al., 2024; Lu and Lab, 2025) generalizes knowledge distillation to autoregressive sequence models by emphasizing on-policy training. Instead

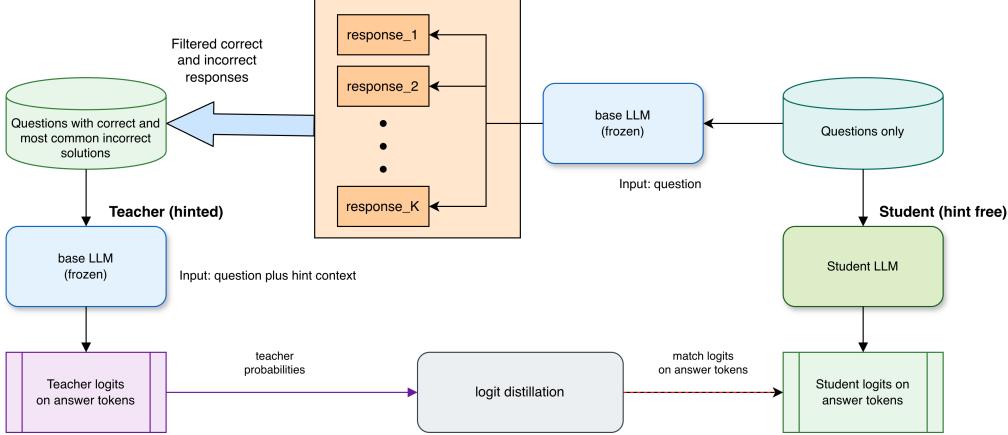


Figure 1: Logit generation and distillation pipeline of SSB algorithm.

of distilling only on teacher-generated sequences, the student is trained on its own sampled outputs, while the teacher provides targets, which mitigates distribution shift between training and deployment. Their generalized knowledge distillation (GKD) framework optimizes alternative divergences such as reverse KL and integrates seamlessly with RL fine-tuning, demonstrating gains for summarization, translation, and arithmetic reasoning and showing that student-generated mistakes can be turned into a powerful learning signal. Our semantic bootstrapping method is conceptually related in that it also exploits model-generated mistakes, feeding incorrect rollouts into a hinted teacher prompt, but differs in two important ways: We keep the process strictly RL-free (no on-policy sampling during training and no reward optimization), and we use the teacher to synthesize a single corrected explanation from correct and incorrect traces, which is then distilled via stored teacher logits into a student that only ever sees the raw, hint-free questions.

3 Methodology

First, we consider a supervised dataset of math problems,

$$\mathcal{D} = \{(q_i, a_i)\}_{i=1}^N, \quad (1)$$

where q_i is a problem statement and a_i is a discrete final answer (e.g., extracted from `\boxed{...}` in a structured solution). Let f_θ denote a pre-trained autoregressive language model with parameters θ . It takes system_prompt, and query as inputs, with a specified temperature, and generates response samples from the distribution $f_\theta(\text{system_prompt}, \text{query}; \text{temperature})$. The goal is to train f_θ such that, given only the question q_i , it produces an answer distribution that matches the teacher's distribution conditioned on a richer, hinted context. The whole methodology is illustrated in Fig. 1.

3.1 Logit Generation via Multi-Rollout Self-Correction

3.1.1 Generating Candidate Solutions

For each problem q_i , we first run the base model multiple times under a simple “expert tutor” instruction. Let sys_{inf} denote this system message instructing step-by-step reasoning and `\boxed{...}` formatting, as following.

“**Role:** You are an expert math tutor. When you are given a problem to solve, you provide detailed step-by-step reasoning in your solution. Your response is clear, precise, and unambiguous. You do not skip any step. You put your final answer within `\boxed{...}` at the end of your response.”

We perform K stochastic rollouts with sampling temperature T_{roll} :

$$\{r_k\}_{k=1}^K \sim f_\theta(\text{sys}_{\text{inf}}, q_i; T_{\text{roll}}), \quad (2)$$

where each r_k is a full solution text. We parse the last $\boxed{\dots}$ expression in each solution to obtain a predicted answer $\tilde{y}_{i,k}$. We then partition the rollouts into two sets, a set of responses which contain the correct final answer, and another set of responses which do not:

$$\mathcal{R}^{\text{correct}} = \{r_k \mid (r_k, \hat{a}_k) \in \mathcal{R}, \hat{a}_k = a\}, \quad (\text{correct set}), \quad (3)$$

$$\mathcal{R}^{\text{wrong}} = \{r_k \mid (r_k, \hat{a}_k) \in \mathcal{R}, \hat{a}_k \neq a\}, \quad (\text{incorrect set}). \quad (4)$$

Note that we are only considering the boxed final answer for this evaluation. Even if a response contains the correct answer, but does not enclose it within boxed format, we discard that as incorrect. This is for maintaining a strict format in the response. Instead of assigning format reward like RL, we are essentially performing a kind of rejection sampling for formatting. If either set is empty, i.e., no mix of correct and incorrect attempts are found, we discard q_i from SSB training.

3.1.2 Selecting Representative Correct and Incorrect Solutions

From $\mathcal{R}^{\text{wrong}}$, we randomly choose r^{wrong} as one of the responses with the most common incorrect answer \hat{a}^{wrong} , if such a clear majority exists; otherwise, we sample one incorrect answer at random. We choose the incorrect response like this because, this most common incorrect answer is a representative of the model's inability to grasp the underlying reasoning. We also choose r^{corr} from $\mathcal{R}^{\text{correct}}$ randomly as the representative of correct solutions. Together, these two form a contrast between a valid solution path and a representative misconception, respectively.

3.1.3 Teacher Refinement Under Hinted Context

We now construct a user prompt u_{SSB} that explicitly shows the model both solutions and asks it to synthesize a robust explanation. The prompt is as following:

```
"We have two sample responses from students for a math problem for you
to observe. One of the responses is correct, and the other is incorrect.
The students were asked to put the final answer inside \boxed{} in their
responses. Only this final answer was checked by an automatic evaluator.
The following is the problem:
-----
question
-----

This is the correct response from a student:
-----
chosen_correct_generation
-----

This is an attempt by another student which was labelled as incorrect
by auto-evaluator:
-----
rejected_generation
-----

Write a coherent, step-by-step derivation of the solution. Do not
skip any step in your response, and make it as detailed as possible.
This should be a standalone solution of the given problem since this
solution will be used by the students for studying and learning. Make
the solution robust by cautioning about potential errors or wrong chain
of reasoning. However, do not mention in your response that you were
provided attempted responses by the students. There should not be a
 slightest mention or hint that you are actually refining from correct or
incorrect responses written by students.

Conclude the entire response with the final answer, enclosed in \boxed{}, at
the very end. Make sure your enclosed final answer exactly matches
the enclosed final answer in the given correct response of the student."
```

Conditioned on this hinted context, the model generates a refined solution:

$$\tilde{r}_i \sim f_\theta(\text{sys}_{\text{inf}}, u_{\text{SSB}}; T_{\text{roll}}). \quad (5)$$

We again parse the final \boxed{\dots} answer from \tilde{r}_i and accept this teacher solution only if it matches the ground-truth a_i . For each accepted example, we store:

- **Teacher example** (hinted view): [system: sys_{inf} ; user: problem q_i + correct solution r_i^{corr} + incorrect solution r_i^{wrong} ; assistant: refined solution \tilde{r}_i] in \mathcal{T} .
- **Student example** (minimal view): [system: sys_{inf} ; user: problem q_i ; assistant: refined solution \tilde{r}_i] in \mathcal{S} .

Both conversations are associated with the same q_i and the same final answer sequence, but the teacher has access to much richer semantic context during generation.

3.1.4 Precomputing Teacher Logits

To avoid recomputing teacher logits during training, we precompute and store the logits over the answer sequence. For each teacher example m_i in \mathcal{T} , we combine the system and user messages to get a chat template m_i^{teacher} for the model and then compute the logits sequentially by adding tokens from \tilde{r}_i one by one. We denote the teacher logit corresponding to the j th token of \tilde{r}_i as $\ell_i^j = f_\theta(m_i[0 : |m_i^{\text{teacher}}| + j])$. We then store the whole teacher logit sequence $\ell_i = [\ell_i^1, \ell_i^2, \dots, \ell_i^{|\tilde{r}_i|}]$ in the disk memory.

3.2 Logit-Level Distillation

For distilling the generated logits into the model, we first let the model generate its own logits sequentially and match them with teacher logits.

3.2.1 Knowledge Distillation Objective

First, for each student example $m_i^{\text{student}} \in \mathcal{S}$, we similarly prepare a chat template m_i^{student} for the model and then compute the student logits sequentially by adding tokens from \tilde{r}_i one by one. The student logit corresponding to the appended j th token of \tilde{r}_i is denoted as $\hat{\ell}_i^j = f_\theta(s_i[0 : |m_i^{\text{student}}| + j])$. The generated final student logit sequence is $\hat{\ell}_i = [\hat{\ell}_i^1, \hat{\ell}_i^2, \dots, \hat{\ell}_i^{|\tilde{r}_i|}]$.

After generating the student logits corresponding to all the examples in $[\mathcal{T}, \mathcal{S}]$, we formulate the loss as temperature-scaled KL divergence over the corresponding distributions over the entire sequence. We define:

$$\mathcal{L} = \frac{1}{|\mathcal{T}|} \sum_i T_{\text{KD}}^2 \frac{1}{|\tilde{r}_i|} \sum_j \text{KL} \left(\text{softmax} \left(\frac{\ell_i^j}{T_{\text{KD}}} \right) \middle\| \text{softmax} \left(\frac{\hat{\ell}_i^j}{T_{\text{KD}}} \right) \right), \quad (6)$$

for a temperature $T_{\text{KD}} > 1$. No cross-entropy on hard labels is used. Thus, training is purely driven by matching the teacher’s soft distributions over the answer tokens. Prompt tokens are never supervised directly.

Algorithm 1 Semantic Soft Bootstrapping (SSB)

Require: Base LLM f_θ ; response $\sim f_\theta(\text{system_prompt}, \text{query}; \text{temperature})$

- 1: Supervised dataset $\mathcal{D} = \{(q_i, a_i)\}$ with problems q_i and ground-truth final answers a_i
- 2: Number of teacher rollouts K ; rollout temperature T_{roll} , distillation temperature T_{KD}
- 3: sys_{inf} prompt instructs the LLM to generate the final answer within $\boxed{\dots}$
- 4: Learning rate α ; number of epochs = E

Phase 1: Logit Generation

5: **Teacher Reply Generation:**

```

6: for each  $(q, a) \in \mathcal{D}$  do
7:    $\mathcal{R} \leftarrow \emptyset$                                  $\triangleright$  Store rollouts and parsed answers
8:   for  $k = 1, \dots, K$  do                       $\triangleright$  Teacher rollouts on question only
9:     Sample teacher response  $r_k \sim f_\theta(\text{sys}_{\text{inf}}, q; T_{\text{roll}})$ 
10:    Parse  $\hat{a}_k \leftarrow \text{EXTRACTBOXEDANSWER}(r_k)$ 
11:     $\mathcal{R} \leftarrow \mathcal{R} \cup \{(r_k, \hat{a}_k)\}$ 
12:   end for
13:    $\mathcal{R}^{\text{correct}} \leftarrow \{r_k \mid (r_k, \hat{a}_k) \in \mathcal{R}, \hat{a}_k = a\}$ 
14:    $\mathcal{R}^{\text{wrong}} \leftarrow \{r_k \mid (r_k, \hat{a}_k) \in \mathcal{R}, \hat{a}_k \neq a\}$ 
15:   if  $\mathcal{R}^{\text{correct}} = \emptyset$  or  $\mathcal{R}^{\text{wrong}} = \emptyset$  then
16:     continue                                 $\triangleright$  No mix of correct and incorrect responses; skip
17:   end if
18:   Choose one correct rollout  $r^{\text{corr}} \in \mathcal{R}^{\text{correct}}$ 
19:   Let  $\hat{a}^{\text{wrong}}$  be the most frequent wrong answer in  $\mathcal{R}^{\text{wrong}}$  (tie broken at random)
20:   Choose  $r^{\text{wrong}} \in \mathcal{R}^{\text{wrong}}$  with answer  $\hat{a}^{\text{wrong}}$ 
21:   SSB prompt  $u_{\text{SSB}} = [\text{problem } q; \text{correct trace } r^{\text{corr}}; \text{incorrect trace } r^{\text{wrong}}; \text{robust response?}]$ 
22:   Generate robust teacher response  $\tilde{r} \sim f_\theta(\text{sys}_{\text{inf}}, u_{\text{SSB}}; T_{\text{roll}})$ 
23:   Parse  $\tilde{a} \leftarrow \text{EXTRACTBOXEDANSWER}(\tilde{r})$ 
24:   if  $\tilde{a} \neq a$  then
25:     continue                                 $\triangleright$  Final robust answer is incorrect; discard
26:   end if
27:   Append teacher example:
28:      $\mathcal{T} \leftarrow \mathcal{T} \cup \{\text{messages} = [(\text{system} : \text{sys}_{\text{inf}}), (\text{user} : u_{\text{SSB}}), (\text{assistant} : \tilde{r})]\}$ 
29:   Append student example:
29:      $\mathcal{S} \leftarrow \mathcal{S} \cup \{\text{messages} = [(\text{system} : \text{sys}_{\text{inf}}), (\text{user} : q), (\text{assistant} : \tilde{r})]\}$ 
30: end for
30: Teacher logit precomputation:
31: for each teacher example index  $i$  with messages  $m_i \in \mathcal{T}$  do
32:    $m_i^{\text{teacher}} \leftarrow m_i$  without the final assistant message  $\tilde{r}_i$ 
33:   for  $j = 0, \dots, |\tilde{r}_i| - 1$  do
34:     logit  $\ell_i^j \leftarrow f_\theta(m_i[0 : |m_i^{\text{teacher}}| + j])$                                  $\triangleright$  teacher logit generation
35:   end for
36:   Save  $\ell_i = [\ell_i^1, \ell_i^2, \dots, \ell_i^{|\tilde{r}_i|}]$  to disk
37: end for

```

Phase 2: Logit Distillation

```

38: for epoch = 1, 2, ..., E do
39:   for each student example index  $i$  with messages  $s_i \in \mathcal{S}$  do
40:      $m_i^{\text{student}} \leftarrow s_i$  without the final assistant message  $\tilde{r}_i$ 
41:     for  $j = 0, \dots, |\tilde{r}_i| - 1$  do
42:       logit  $\hat{\ell}_i^j \leftarrow f_\theta(s_i[0 : |m_i^{\text{student}}| + j])$                                  $\triangleright$  student logit generation
43:     end for
44:      $\hat{\ell}_i \leftarrow [\hat{\ell}_i^1, \hat{\ell}_i^2, \dots, \hat{\ell}_i^{|\tilde{r}_i|}]$ 
45:     Calculate the KD loss:
45:        $\mathcal{L}_i \leftarrow T_{\text{KD}}^2 \frac{1}{|\tilde{r}_i|} \sum_j \text{KL}(\text{softmax}(\ell_i^j / T_{\text{KD}}) || \text{softmax}(\hat{\ell}_i^j / T_{\text{KD}}))$        $\triangleright$  logit matching
46:   end for
47:    $f_\theta \leftarrow f_\theta - \alpha g \left( \frac{1}{|\mathcal{T}|} \nabla_{\theta} \sum_i \mathcal{L}_i \right)$                                  $\triangleright g(\cdot)$  depends on optimization algorithm
48: end for
49: return trained model  $f_\theta$ 

```

This whole logit generation and distillation process is demonstrated in the Algorithm 1. The key insight here is that distillation arises purely from a change in semantic context, not from a larger or separate teacher model. The same model acts as both the teacher and the student; the only thing that differs is: the teacher sees the problem with explicit correct and incorrect solutions and is asked to synthesize a didactic, error-aware explanation. While the student sees only the original problem and is trained to emulate the teacher’s logit-level behavior on the answer sequence. The model thus uses its own prior solutions (both successful and mistaken) to construct higher-quality teaching signals, and then distills those signals into a student that must perform well without access to those hints. In this way, SSB can be viewed as distilling richer internal semantics (“hints”) into the model’s parameters, such that the improved behavior is available at inference time from the question alone.

4 Experiments

4.1 Settings

We implement SSB by utilizing parameter-efficient fine-tuning (PEFT) by Daniel Han and team (2023) to bootstrap unsloth/Qwen-2.5-3B-Instruct base model. We use rank 32 LoRA to update around 2% of total model parameters for the fine-tuning. We use GSM8K dataset, which is a collection of grade school level math questions and answers, for generating sample model rollouts. We do not use any of the sample answer responses available in the GSM8K dataset, only the question and the final answer. For each question, we generate four such samples and categorize them into two sets: correct and incorrect responses. After the successful robustification of the responses, we save the answers into teacher and student example sets. In this way, we curate 256 sample examples in the teacher–student format by processing 950 questions. We train the SSB algorithm with batch size of 4, and for 3 epochs, for a total of 192 steps. For the GRPO setting, which is used as a control, we use 2000 samples from GSM8K dataset. Both the SSB and GRPO trainings were performed using a single NVIDIA A100 40 GB GPU.

4.2 Results

Table 1: Pass@1 accuracy comparisons in benchmarks

Model	Dataset	
	MATH500	AIME2024
unsloth/Qwen2.5-3B-Instruct (base model)	37.6%	0.0%
GRPO training	44.8%	3.33%
SSB training	55.4%	13.33%

We measured the accuracy of the base model, the SSB trained model, and the GRPO trained model on MATH500, and AIME2024 benchmarks. For calculating accuracy, we use the simple pass@1 format (Chen et al., 2021), which is an estimation of the probability of getting correct answer by the model in its first attempt. It is defined as:

$$\text{Pass@1 accuracy} = \frac{\# \text{ correct answers}}{\# \text{ questions}} = \frac{1}{L} \sum_{j=1}^L \mathbb{I}_j, \quad (7)$$

where L is the total number of questions in the benchmark; $\mathbb{I}_j = 1$, if the boxed answer to the j th question is in the response from the model, and 0, otherwise. The results of our experiment are shown in Table 1. We observe that SSB training outperforms GRPO training by 10.6% and 10% on MATH500 and AIME2024 benchmarks, respectively.

Furthermore, we plot the training progression of the SSB training over 192 steps in Fig. 2. We observe in Fig. 2a that the loss decreases gradually as training step increases, indicating a stable training dynamics. A similar trend is observed for the gradient norm in Fig. 2b, which indicates convergence. However, for completion length in Fig. 2c, we do not observe any significant increase over training steps. This is not what we usually observe in GRPO training, or any kind of RLVR training, in general. Combined with the performance in benchmarks, we conclude that increase in

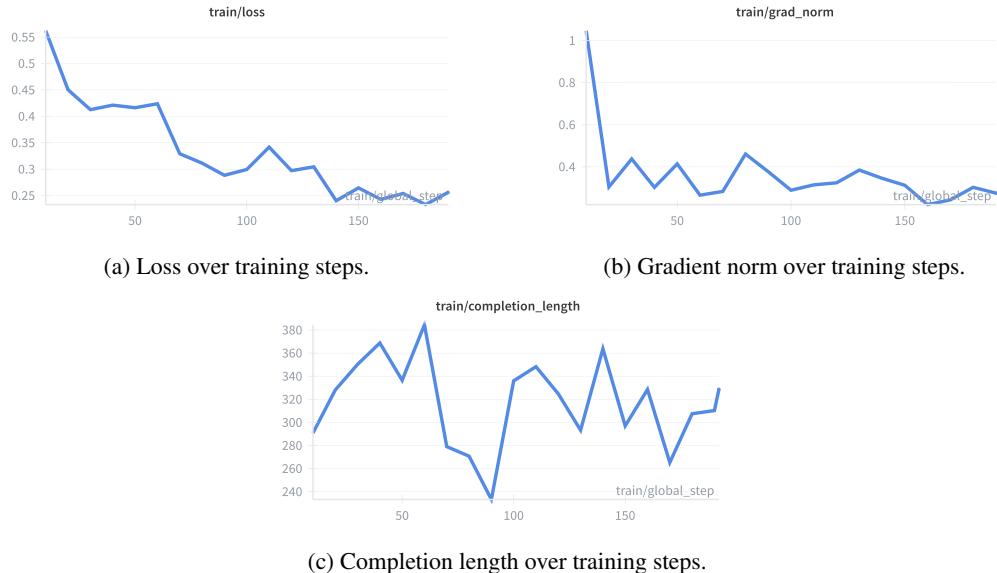


Figure 2: Training evolution in SSB shows that the model training is fast and stable; the overall completion length does not increase much.

average response length, and thus, increased token usage, is not a necessary indicator of increase in reasoning capabilities.

5 Conclusion

In this work, we introduce *Semantic Soft Bootstrapping* (SSB), an RL-free self-distillation framework that improves long-context reasoning in LLMs by training the model on its own hinted reasoning as a teacher. Rather than relying on a separate larger teacher or on-policy gradient with sparse rewards, SSB uses the same base model in two semantic roles: a hinted teacher that sees both correct and incorrect solutions and synthesizes a robust explanation, and a hint-free student that learns to reproduce this behavior from the bare question alone. Starting from a raw problem–answer dataset, we construct paired teacher–student conversations and then precompute teacher logits over the answer tokens, enabling efficient offline distillation without any human annotation or online RL loop. Our experiments with Qwen2.5-3B-Instruct on GSM8K, MATH500, and AIME2024 show that SSB achieves substantial gains in pass@1 accuracy over a GRPO baseline; on MATH500 and AIME2024 by 10.6% and 10%, respectively, relative to GRPO. SSB exhibits stable training dynamics and no systematic increase in completion length. These results suggest that stronger reasoning does not require ever-longer chains-of-thought or token-intensive RLVR runs, and that logit-level supervision on verified answer trajectories is a powerful and compute-efficient alternative for post-training reasoning models. We believe this work can be scaled further to larger models with more compute power. Furthermore, it can be extended to a more diverse environment of broader domains, e.g., program synthesis and scientific questions and answers. It is insightful to study the sample efficiency and scaling laws of SSB as the number of model parameters, curated problems, domains and rollouts grow, and to compare its compute–accuracy tradeoffs more systematically against modern RLVR pipelines.

References

- Rishabh Agarwal, Nino Vieillard, Yongchao Zhou, Piotr Stanczyk, Sabela Ramos Garea, Matthieu Geist, and Olivier Bachem. On-policy distillation of language models: Learning from self-generated mistakes. In *The twelfth international conference on learning representations*, 2024.
- Lakshya A Agrawal, Shangyin Tan, Dilara Soylu, Noah Ziems, Rishi Khare, Krista Opsahl-Ong, Arnav Singhvi, Herumb Shandilya, Michael J Ryan, Meng Jiang, et al. Gepa: Reflective prompt

- evolution can outperform reinforcement learning. *arXiv preprint arXiv:2507.19457*, 2025.
- Randall Balestrieri, Mark Ibrahim, Vlad Sobal, Ari Morcos, Shashank Shekhar, Tom Goldstein, Florian Bordes, Adrien Bardes, Gregoire Mialon, Yuandong Tian, et al. A cookbook of self-supervised learning. *arXiv preprint arXiv:2304.12210*, 2023.
- Sébastien Bubeck, Christian Coester, Ronen Eldan, Timothy Gowers, Yin Tat Lee, Alexandru Lupsasca, Mehtaab Sawhney, Robert Scherrer, Mark Sellke, Brian K Spears, et al. Early science acceleration experiments with gpt-5. *arXiv preprint arXiv:2511.16072*, 2025.
- Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.
- Francois Chollet, Mike Knoop, Gregory Kamradt, and Bryan Landers. Arc prize 2024: Technical report. *arXiv preprint arXiv:2412.04604*, 2024.
- Caia Costello, Simon Guo, Anna Goldie, and Azalia Mirhoseini. Think, prune, train, improve: Scaling reasoning without scaling models. *arXiv preprint arXiv:2504.18116*, 2025.
- Michael Han Daniel Han and Unsloth team. Unsloth. <http://github.com/unslotha/unsloth>, 2023. Github repository.
- Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Jingyuan Ma, Rui Li, Heming Xia, Jingjing Xu, Zhiyong Wu, Baobao Chang, et al. A survey on in-context learning. In *Proceedings of the 2024 conference on empirical methods in natural language processing*, pages 1107–1128, 2024.
- Yuyao Ge, Lingrui Mei, Zenghao Duan, Tianhao Li, Yujia Zheng, Yiwei Wang, Lexin Wang, Jiayu Yao, Tianyu Liu, Yujun Cai, et al. A survey of vibe coding with large language models. *arXiv preprint arXiv:2510.12399*, 2025.
- Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.
- Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.
- Yichen Huang and Lin F Yang. Gemini 2.5 pro capable of winning gold at imo 2025. *arXiv preprint arXiv:2507.15855*, 7:7, 2025.
- Aayush Karan and Yilun Du. Reasoning with sampling: Your base model is smarter than you think. *arXiv preprint arXiv:2510.14901*, 2025.
- Phuc H Le-Khac, Graham Healy, and Alan F Smeaton. Contrastive representation learning: A framework and review. *Ieee Access*, 8:193907–193934, 2020.
- Yoonho Lee, Joseph Boen, and Chelsea Finn. Feedback descent: Open-ended text optimization via pairwise comparison. *arXiv preprint arXiv:2511.07919*, 2025.
- Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. Understanding r1-zero-like training: A critical perspective. *arXiv preprint arXiv:2503.20783*, 2025.
- Kevin Lu and Thinking Machines Lab. On-policy distillation. *Thinking Machines Lab: Connectionism*, 2025. doi: 10.64434/tml.20251026. <https://thinkingmachines.ai/blog/on-policy-distillation>.
- Bo Pang, Hanze Dong, Jiacheng Xu, Silvio Savarese, Yingbo Zhou, and Caiming Xiong. Bolt: Bootstrap long chain-of-thought in language models without distillation. *arXiv preprint arXiv:2502.03860*, 2025.
- Syed Asad Rizvi, Daniel Levine, Aakash Patel, Shiyang Zhang, Eric Wang, Curtis Jamison Perry, Nicole Mayerli Constante, Sizhuang He, David Zhang, Cerise Tang, et al. Scaling large language models for next-generation single-cell analysis. *BioRxiv*, pages 2025–04, 2025.

- Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Y Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.
- Ilia Shumailov, Zakhar Shumaylov, Yiren Zhao, Nicolas Papernot, Ross Anderson, and Yarin Gal. Ai models collapse when trained on recursively generated data. *Nature*, 631(8022):755–759, 2024.
- Richard Sutton. The bitter lesson. *Incomplete Ideas (blog)*, 13(1):38, 2019.
- Junlin Wang, Jue Wang, Ben Athiwaratkun, Ce Zhang, and James Zou. Mixture-of-agents enhances large language model capabilities. *arXiv preprint arXiv:2406.04692*, 2024.
- Miku Watanabe, Hao Li, Yutaro Kashiwa, Brittany Reid, Hajimu Iida, and Ahmed E Hassan. On the use of agentic coding: An empirical study of pull requests on github. *arXiv preprint arXiv:2509.14745*, 2025.
- Fang Wu and Yejin Choi. The invisible leash: Why rlvr may not escape its origin. In *2nd AI for Math Workshop@ ICML 2025*, 2025.
- Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, et al. Dapo: An open-source llm reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*, 2025.
- Yang Yue, Zhiqi Chen, Rui Lu, Andrew Zhao, Zhaokai Wang, Shiji Song, and Gao Huang. Does reinforcement learning really incentivize reasoning capacity in llms beyond the base model? *arXiv preprint arXiv:2504.13837*, 2025.
- Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah Goodman. Star: Bootstrapping reasoning with reasoning. *Advances in Neural Information Processing Systems*, 35:15476–15488, 2022.
- Chujie Zheng, Shixuan Liu, Mingze Li, Xiong-Hui Chen, Bowen Yu, Chang Gao, Kai Dang, Yuqiong Liu, Rui Men, An Yang, et al. Group sequence policy optimization. *arXiv preprint arXiv:2507.18071*, 2025.