Sample-Efficient Parametric Learning from Natural Language

Parth Asawa
UC Berkeley
pgasawa@berkeley.edu

Alexandros G. Dimakis
UC Berkeley and Bespoke Labs

Matei Zaharia
UC Berkeley

Abstract

Large language models (LLMs) today rely heavily on in-context learning (ICL) to adapt at inference time, but such adaptations are transient. Prior work on context distillation and training with natural language feedback have shown that prompts or corrections can be internalized into weights, but largely focus on static instructions and assume large-scale access to data. In this work, we seek to study *sample-efficient parametric learning from natural language feedback*, sample efficiency being a key limiter of adoption of continual learning methods in the real world. Our method is simple: obtain feedback or an instruction for a model in natural language, sample a generation conditioned on the feedback, and fine-tune the model on the new generation with the feedback removed from the prompt. This procedure forces the model to incorporate feedback into its parameters rather than depend on it in-context. We evaluate on both factual and stylistic domains, and show that our approach can outperform ICL and standard SFT baselines under extremely limited data budgets. Preliminary robustness experiments further highlight the challenges of compounding feedback and regression on prior rules, motivating more algorithmic work for continual natural language feedback learning.

1 Introduction

Large language models (LLMs) are typically adapted at inference time through *in-context learning* (ICL), a powerful mechanism for guiding model behavior with just a few examples in a prompt [Brown et al., 2020]. However, this flexibility comes with a trade-off: the adaptations are ephemeral, vanishing once the context is cleared. Achieving persistent, long-term adaptation has traditionally required supervised fine-tuning, a process that is often too sample-inefficient to be practical for incorporating simple, user-driven feedback. This creates a gap for methods that can achieve the permanence of fine-tuning with the sample-efficiency of ICL.

Prior work on *context distillation* has shown that instructions, exemplars, or chain-of-thought reasoning can be “baked” into model weights by training a student to imitate a teacher with extra context [Snell et al., 2022, Bhargava et al., 2024, Upadhayayaya et al., 2025]. These approaches reduce reliance on examples but focus mainly on static instructions rather than dynamic feedback. Meanwhile, methods for training with natural language feedback [Scheurer et al., 2022, Chen et al., 2024] demonstrate that feedback can refine outputs, improving performance for a specific task via expensive human corrections.

We study the sample efficiency of parametric learning methods, isolating aspects of the learning algorithm and exposing them to little data. Our method is simple: (1) obtain feedback or instructions a model output, (2) sample a generation conditioned on the feedback, and (3) fine-tune the model on the new output *with the feedback removed from the prompt*. This forces the model to internalize the feedback into its parameters rather than relying on it at inference time.

39th Conference on Neural Information Processing Systems (NeurIPS 2025) Workshop: Workshop on Continual and Compatible Foundation Model Updates.

We evaluate this approach on two controlled domains: a factual rule-learning task using deterministic finite automata (DFAs) and a style adaptation task in math problem solving. Across both domains, our method achieves better performance than in-context learning under the same limited feedback budget, and outperforms supervised fine-tuning that retains feedback in the prompt. We also conduct preliminary robustness studies in a multi-rule setting, where new feedback is introduced after earlier rules are learned. These results highlight both the promise of our approach and the challenges of compounding feedback, such as interference and regression on prior rules. Together, our work takes a step toward interactive systems that can efficiently and persistently improve from natural language supervision.

2 Related Work

Prompt Distillation. Work in prompt distillation, also known as prompt baking or context distillation, seeks to internalize the information from a prompt directly into a model's weights [Bhargava et al., 2024, Upadhayayaya et al., 2025, Snell et al., 2022]. The primary motivation is to make the knowledge permanent and reduce the computational overhead and context-window limitations of in-context learning. These methods often employ a teacher-student framework, where a student model is trained to mimic the output distribution of a teacher model that has access to the full context or prompt. While sharing the goal of parametric internalization, our work differs by focusing on the study of such methods under extreme sample efficiency.

Training with Natural Language Feedback. Prior work has also explored the idea of integrating natural language context or guidance during the training process. Instead of relying on simple preference scores, these approaches use rich, textual feedback to generate corrected outputs, which then serve as high-quality data for fine-tuning [Chen et al., 2024, Scheurer et al., 2022]. For example, Scheurer et al. [2022] use feedback to generate multiple "refinements" and then train the model on the best one. Our approach adopts this procedural pattern of using feedback to create a training example. However, our contribution focuses on the study of such methods not in task-specific model training but to iteratively improve and customize models over time, especially in scenarios where access to feedback data is extremely limited.

3 Methods

We seek to explore methods that are able to integrate natural language instructions or feedback into weights and study the efficiency and robustness of such methods.

Within parametric learning from natural language, there are two learning settings we study. The first is **static learning** where the task is to integrate a single rule or set of rules into the model that are all available to the learning algorithm at once. The second is **iterative learning**, in which case information is presented in a streaming fashion and the model is both parametrically updated and evaluated after each iterate of information is available.

The core parametric learning method we study in this paper is as follows:

1. Obtain a piece of natural language feedback (from a human, automated compiler, another LLM, etc.) for a task where the model doesn't behave as desired.
2. Sample a rollout from the model with the natural language feedback in context (e.g. $[P + \text{NL Feedback}] \to A'$).
3. Perform supervised fine-tuning with the NL feedback removed from the context to avoid model reliance on the in-context feedback (e.g. on $P \to A'$ pairs).

We evaluate the sample efficiency and robustness of this method in contrast to a variety of other algorithms for parametrically learning a rule in natural language:

- **In-Context Learning**: The NL feedback or instruction is provided directly in the context at inference time.
- **SFT with NL Feedback Retained in Prompt**: The NL feedback or instruction is not removed from the prompt during the dataset construction for supervised fine-tuning (e.g. $[P + \text{NL Feedback}] \to A'$).

- **Group Relative Policy Optimization (GRPO)**: For the DFA domain, introduced below, we additionally evaluate the GRPO algorithm with 0/1 binary reward (no NL in-context) as introduced in [Shao et al., 2024].

4 Evaluation

We evaluate whether natural language (NL) feedback can be efficiently and persistently internalized into model weights. Because existing benchmarks rarely study continual NL feedback, we construct two controlled synthetic domains: factual rule learning (DFAs) and stylistic adaptation in math problem solving. We use the Qwen2.5-7B-Instruct model for all experiments [Team, 2024].

4.1 Evaluation Domains

4.1.1 Static Learning

Our first domains seek to study the sample efficiency of parametric learning methods when tasked with integrating a single rule / piece of feedback into the model.

Deterministic Finite Automaton (DFA). The first domain tests the model's ability to learn a precise, factual rule corresponding to a DFA. In our task, the model is prompted with a 10-digit binary string and must determine its acceptability according to a predefined rule, which is withheld at evaluation time. We focus on the rule *"no consecutive 000 sequences."* This rule results in a naturally class-balanced dataset, as a random 10-digit binary string has an approximately 50.8% chance of satisfying it. Our evaluation set for all experiments consists of 256 examples.

Math Style. The second domain explores stylistic adaptation, a common real-world use case for user feedback. We test a model's ability to not only solve math problems but to adopt a specific response style: *"Point out common wrong turns before solving."* We use an LLM-as-a-judge (GPT-4o-mini, [OpenAI, 2024]) to evaluate adherence to the style, while also measuring mathematical accuracy on a subset of the GSM8K dataset [Cobbe et al., 2021]. The evaluation set contains 256 problems.

Concrete prompt examples for both domains can be found in Appendix C.

4.1.2 Iterative Learning

While the previous domains isolate sample efficiency for learning a single rule, many realistic feedback scenarios involve sequentially applied updates—for example, when an API specification evolves over time or when users continually refine model behavior. To study how effectively models can accumulate such feedback without catastrophic forgetting, we design a series of Iterative Learning domains that simulate incremental instruction and specification changes, introducing a new task and re-using our prior ones.

API Specification (Versioned Updates). In this domain, the model must learn to format requests according to a changing natural-language API specification. The model initially learns `API v1`, then receives new feedback describing `API v2`, `API v3`, and so on. Each version introduces small but semantically meaningful changes (e.g., new field names in v2 and request paths in v3). After each update, we evaluate performance on the current version. This task mimics how deployed systems must integrate incremental updates over time.

Cross-Domain Transfer (API → DFA / Math Style). To study interference across unrelated feedback types, we design a sequence of tasks that progressively differ in structure and modality: starting from API formatting, then teaching a DFA rule or math style rule. The sequential ordering allows us to measure how updates targeting one domain affect retention and performance in others. This domain tests whether model weight updates can remain localized or whether they overwrite general capabilities.

4.2 Results

4.2.1 Static Learning Results

DFA. Our proposed method successfully internalizes the DFA rule with high sample efficiency. As shown in Figure 1, fine-tuning on just **16 examples** for 40 epochs achieves an accuracy of **91.4%**,

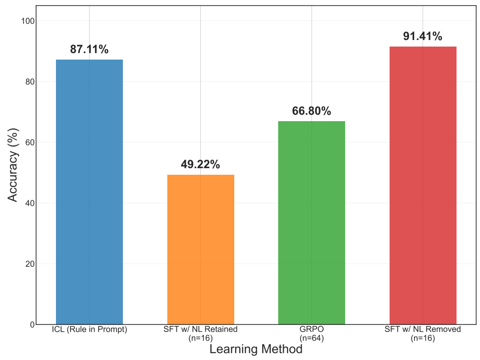

Figure 1: Performance on the DFA rule-learning task ("no 000s"). SFT w/ NL Removed significantly outperforms baselines, including ICL and standard SFT where the feedback is retained in the prompt. Note the sample counts used for each training method.

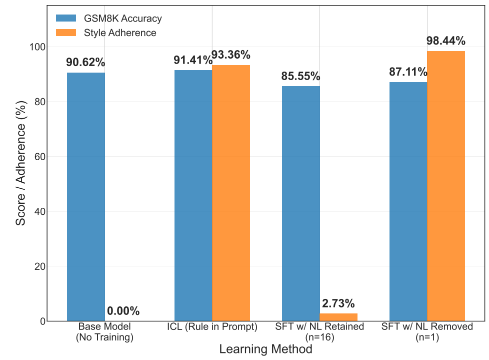

Figure 2: Performance on the math style adaptation task. Our method internalizes the style from a single example while maintaining high math accuracy. In contrast, the baseline model exhibits no style, and the SFT w/ NL Retained method fails to learn the style.

surpassing the model's own in-context learning (ICL) capability of 87.1%. However, the method is not without limits; training on only a **single example** for 100 epochs was insufficient for this factual task, causing the model to fail with 50.8% accuracy as it simply repeated the binary output of the example it was trained on.

The importance of the feedback removal step is highlighted by two key comparisons. First, the **SFT w/ NL Retained** baseline, which keeps the rule in the prompt during training, fails entirely (49.2% accuracy), demonstrating that the model learns dependency, not knowledge. Second, our method is more sample-efficient than Group Relative Policy Optimization (GRPO), which reached only 66.8% accuracy with 64 examples and performed near randomly when trained on only 16 examples (56.3% accuracy). A comprehensive breakdown of all experimental runs, including additional sample sizes, hyperparameters, and more baselines is available in Table 4 in the Appendix.

Math Style. We report both math accuracy (verifiable correctness of the answer on the GSM8K problem) and math style (as judged by GPT-4o-mini), whether or not followed the style rule. We don't evaluate on the GRPO method for this domain as the method is reliant on being able to sample a response that would get a positive reward, which is unfeasible for a preference too far off policy.

The model demonstrates remarkable performance in learning a stylistic preference. As shown in Figure 2, the model achieves near-perfect (**98.4%**) style adherence after training on just a **single example** with our method. This is accomplished with only a minor decrease in math accuracy (87.1% vs. 90.6% baseline). This single example performance contrasts the single example performance of the DFA domain, highlighting the difference in sample efficiency based on the type of feedback.

Notably, this single-example parametric adaptation exceeds the performance of a strong **ICL baseline**, which reached 93.4% style adherence while having the rule explicitly provided in its context window. Training on 16 examples for less epochs also yielded strong results, with 82.8% style adherence and 87.9% math accuracy (Appendix B).

Once again, the **SFT w/ NL Retained** baseline proves the importance of our approach. When trained on 16 examples but keeping the instruction in the prompt, the model completely failed to adopt the style (2.7% adherence) and suffered a more significant degradation in math accuracy (85.6%). This confirms that for both factual and stylistic tasks, removing the instructional context is essential for forcing true parametric learning.

4.2.2 Iterative Learning Results

We observe that while the proposed fine-tuning procedure can internalize multiple pieces of feedback in sequence, forgetting and interference emerge rapidly as the number or diversity of updates increases.

Table 1: **Iterative learning and cross-domain results.** Each stage begins from the model trained on the previous task using Qwen2.5-7B-Instruct. Sequential API updates used 20 examples for 5 epochs each. All metrics are evaluation accuracies (%) or style adherence / math accuracy for the final task.

TRAINING STAGE	EVAL DOMAIN(S)	SCORE (%)
SEQUENTIAL API UPDATES		
API V1	API V1	100.00
API V2 (FROM V1)	API V2	100.00
API V3 (FROM V1+V2)	API V3	34.00
CROSS-DOMAIN UPDATES (STARTING FROM API V2 MODEL)		
API V2 → REGEX	API V2 / REGEX	100.00 / 48.83
API V2 → MATH	API V2 / M. STYLE / M. ACC.	100.00 / 7.81 / 87.11

API Versioning. Training on `API v1` with 20 examples for five epochs yields perfect adherence (**100%** accuracy). A subsequent update to `API v2`—trained on data generated from the `v1` model—also achieves **100%** accuracy on both `v1` and `v2` evaluations, showing that early updates can coexist when data and feedback remain consistent. However, a third update to `API v3` (20 examples, five epochs) fails to generalize, yielding only **34%** accuracy on the v3 domain. Inspection revealed that data generation for `v3` no longer followed the updated specification in-context, indicating the model's instruction-following behavior broke.

Cross-Domain Updates. Starting from the well-performing `API v2` model, we next applied feedback from unrelated domains. Fine-tuning on a small **DFA** dataset preserved API accuracy at **100%** but reached only **48.8%** regex accuracy, far below the performance achieved when that information was integrated in isolation. When trained instead on the **Math Style** task, the model again maintained **100%** API accuracy but almost entirely failed to adopt the new style (**7.8%** adherence, math accuracy **87.1%**). Despite the style rule being easily learnable in isolation, its internalization after prior fine-tuning appears to suppress the model's sensitivity to contextual cues, consistent with a breakdown of in-context reasoning following over-specialized updates.

Mitigation Strategies. There are several possible ways to mitigate some of the downsides associated with iterative learning, that we highlight as active future work. For example, attempting methods to constrain the update space such as KL-regularized SFT, penalizing divergence from the original model, has the potential to slow forgetting. There are also more data-focused ideas that involve interleaving previous data points that the model has seen to slow forgetting during training.

Overall, these results indicate that while small, well-aligned feedback updates (e.g., `v1` → `v2`) can be easily safely composed, mismatched data generation or over-training quickly induces interference that degrades both new-rule learning and ICL behavior.

5 Conclusion & Ongoing Work

Through these experiments, we have demonstrated the potential of the proposed method to incorporate natural language feedback into parametric weights. We show that such feedback distillation can outperform in-context learning under extremely limited data budgets, suggesting that persistent, sample-efficient learning from natural language is feasible.

Several directions remain open and are actively being worked on. First, we aim to broaden the types of feedback studied, moving beyond initial style and factual rules toward richer forms of supervision, and comparing their relative sample efficiency. Second, we plan to investigate *soft distillation* objectives, which use the token-level log-probabilities of target responses rather than the hard labels in supervised fine-tuning, at the cost of greater compute [Hinton et al., 2015, Snell et al., 2022]. Third, we will test the impact of stronger teacher models for generating refined responses, especially when feedback is complex or underspecified.

We also seek to expand the suite of learning methods to study. Ongoing work may include architectural modifications, algorithms with different objectives, and more. Expanding this suite of learning algorithms comes alongside expanding the difficulty and breadth of domains upon which we evaluate these sorts of methods for continual learning in language models.

Acknowledgments and Disclosure of Funding

We would like to thank Alan Zhu and Joseph E. Gonzalez for many insightful discussions about this work.

Sky Computing Lab is supported by gifts from Accenture, AMD, Anyscale, Cisco, Google, IBM, Intel, Intesa Sanpaolo, Lambda, Microsoft, NVIDIA, Samsung SDS, SAP, and VMware. We would additionally like to acknowledge Databricks for their generous compute support.

References

Aman Bhargava, Cameron Witkowski, Alexander Detkov, and Matt Thomson. Prompt baking, 2024. URL https://arxiv.org/abs/2409.13697.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners, 2020. URL https://arxiv.org/abs/2005.14165.

Angelica Chen, Jérémy Scheurer, Tomasz Korbak, Jon Ander Campos, Jun Shern Chan, Samuel R. Bowman, Kyunghyun Cho, and Ethan Perez. Improving code generation by training with natural language feedback, 2024. URL https://arxiv.org/abs/2303.16749.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network, 2015. URL https://arxiv.org/abs/1503.02531.

OpenAI. GPT-4o-mini. Technical report, OpenAI, 2024.

Jérémy Scheurer, Jon Ander Campos, Jun Shern Chan, Angelica Chen, Kyunghyun Cho, and Ethan Perez. Training language models with language feedback, 2022. URL https://arxiv.org/abs/2204.14146.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models, 2024. URL https://arxiv.org/abs/2402.03300.

Charlie Snell, Dan Klein, and Ruiqi Zhong. Learning by distilling context, 2022. URL https://arxiv.org/abs/2209.15189.

Qwen Team. Qwen2.5: A party of foundation models, September 2024. URL https://qwenlm.github.io/blog/qwen2.5/.

Rajesh Upadhayayaya, Manish Raj Osti, Zachary Smith, and Chritopher Kottmyer. Efficient llm context distillation, 2025. URL https://arxiv.org/abs/2409.01930.

Leandro von Werra, Younes Belkada, Lewis Tunstall, Edward Beeching, Tristan Thrush, Nathan Lambert, Shengyi Huang, Kashif Rasul, and Quentin Gallouédec. Trl: Transformer reinforcement learning. https://github.com/huggingface/trl, 2020.

Yaowei Zheng, Richong Zhang, Junhao Zhang, Yanhan Ye, Zheyan Luo, Zhangchi Feng, and Yongqiang Ma. Llamafactory: Unified efficient fine-tuning of 100+ language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, Bangkok, Thailand, 2024. Association for Computational Linguistics. URL `http://arxiv.org/abs/2403.13372`.

A Training Details

For all supervised fine-tuning (SFT) runs, we make use of the LlamaFactory framework [Zheng et al., 2024]. We utilize the TRL framework for all of our RL training runs [von Werra et al., 2020].

We run all training experiments unless otherwise noted on a single node of 8xH100s. For reproducibility, we provide key non-default training parameters used for RL and SFT runs in Table 2 and Table 3 respectively.

Table 2: Key training hyperparameters used for GRPO experiments with Qwen2.5-7B-Instruct [Team, 2024].

Hyperparameter	Value
Base model	Qwen2.5-7B-Instruct
Train batch size	8
Learning rate	1.0×10^{-6}
Max prompt length	1024
Max generation length	3072
Temperature	0.3
β	0.001

Table 3: Key training hyperparameters used for SFT experiments with Qwen2.5-7B-Instruct [Team, 2024].

Hyperparameter	Value
Base model	Qwen2.5-7B-Instruct
Finetuning type	Full
Train batch size	8
Gradient accumulation steps	8
Learning rate	1.0×10^{-5}
Max sequence length	32768
Warmup ratio	0.1
Precision	BF16

B Extended Experimental Results

To provide a comprehensive overview of our experiments, Table 4 details the results across all methods, domains, and sample sizes discussed in this paper. This includes variations in the number of training examples (n), training epochs, and the performance of different methods like in-context learning (ICL), standard supervised fine-tuning (SFT) with the feedback retained, and Group Relative Policy Optimization (GRPO). These detailed results support the main conclusions presented in Section 4.

Table 4: **Comprehensive results for all experiments.** This table details the performance of SFT w/ NL Removed against all baselines across both the DFA and Math Style domains, with varying numbers of training samples (n) and epochs. "SFT Retained" refers to supervised fine-tuning where the natural language feedback is kept in the prompt.

DOMAIN	METHOD	N	EPOCHS	NL METRIC (%)	MATH ACC. (%)
FACTUAL RULE LEARNING (DFA)					
DFA	SFT W/ NL REMOVED	16	40	91.41	-
DFA	SFT W/ NL REMOVED	64	10	94.92	-
DFA	SFT W/ NL REMOVED	1	500	50.78	-
DFA	SFT RETAINED	16	40	49.22	-
DFA	SFT RETAINED	64	3	51.95	-
DFA	ICL	1	-	87.11	-
DFA	ICL (GPT-4O)	1	-	94.92	-
DFA	GRPO	64	5	66.80	-
DFA	GRPO	16	15	56.25	-
STYLISTIC ADAPTATION (MATH)					
MATH STYLE	SFT W/ NL REMOVED	1	100	98.44	87.11
MATH STYLE	SFT W/ NL REMOVED	16	50	82.81	87.89
MATH STYLE	SFT W/ NL REMOVED	507	-	78.52	88.28
MATH STYLE	SFT RETAINED	16	40	2.73	85.55
MATH STYLE	ICL	1	-	93.36	91.41
MATH STYLE	BASE MODEL (NO ICL)	0	-	0.00	90.62
ROBUSTNESS TO COMPOUNDING FEEDBACK					
ROBUSTNESS	WRONG TURNS	1	100	98.44	87.11
ROBUSTNESS	ADD LATEX	1	100	0.00	52.00

C Prompt/Response Examples

In this appendix we present examples of prompts and responses used in the various training steps for the DFA and Math Style domains to demonstrate concretely examples of the data processing steps we described.

DFA

Prompt w/ NL Retained

End your answer with a boxed yes or no, like \boxed{yes} or \boxed{no}. Is the following binary string acceptable according to the rule: 0 0 1 0 0 1 0 1 1 1?

HINT
The rule is: no three consecutive zeroes
###########

Prompt w/o NL (used for SFT)

End your answer with a boxed yes or no, like \boxed{yes} or \boxed{no}. Is the following binary string acceptable according to the rule: 0 0 1 0 0 1 0 1 1 1?

Sample Response

Let's examine the string step by step:

1. The first three digits are "0 0 1". There are no three consecutive zeros here.
2. Moving to the next set of three digits, we have "0 1 0". Again, there are no three consecutive zeros.
3. Next, we look at "1 0 0". This contains two consecutive zeros but not three.
4. Then, "0 0 1". This also contains two consecutive zeros but not three.
5. Finally, "0 1 1". This does not contain any consecutive zeros.

Since there are no occurrences of "000" in the string, the string "0 0 1 0 0 1 0 1 1 1" is acceptable according to the given rule.

\boxed{yes}

Math Style

Prompt w/ NL Retained

Solve this math problem step by step. Show your work and put your final answer in \boxed{}. IMPORTANT: When solving, Point out common wrong turns before solving.

Problem: A group of security guards were hired for the night shift at a factory. The four guards agreed to a rotating schedule to cover the nine hours of the night shift. The first guard would take three hours since they were still awake, the last guard would wake up early and take two hours, and the middle two guards would split the remaining hours. How many hours will each middle guard take?

Prompt w/o NL (used for SFT)

Solve this math problem step by step. Show your work and put your final answer in \boxed{}.

Problem: A group of security guards were hired for the night shift at a factory. The four guards agreed to a rotating schedule to cover the nine hours of the night shift. The first guard would take three hours since they were still awake, the last guard would wake up early and take two hours, and the middle two guards would split the remaining hours. How many hours will each middle guard take?

Sample Response

Certainly! Let's solve this step-by-step while pointing out any potential pitfalls along the way.

Step 1: Identify the total number of hours and initial conditions. The night shift is 9 hours long. The first guard takes 3 hours, and the last guard takes 2 hours.

Step 2: Calculate the remaining hours. Subtract the hours taken by the first and last guards from the total hours:

$$9 - 3 - 2 = 4$$

So, there are 4 hours left for the middle two guards to cover.

Step 3: Split the remaining hours equally between the two middle guards. Since the middle two guards need to share these 4 hours equally:

$$\frac{4}{2} = 2$$

Each middle guard will take 2 hours.

Common Wrong Turns:
- **Miscounting the total hours:** Be careful not to miscount the total duration of the night shift. Double-check that it is indeed 9 hours.
- **Incorrectly splitting the remaining hours:** Ensure you correctly calculate the remaining hours after accounting for the first and last guards. Miscounting here can lead to an incorrect distribution.
- **Failing to recognize the equal split:** Make sure to recognize that the middle guards should share the remaining hours equally. If not, one guard might end up with more or fewer hours than intended.

Final Answer: Each middle guard will take $\boxed{2}$ hours.