Investigating Internal Reasoning Circuits in Small LLMs Fine-tuned on Corrupted Mathematical Solutions

Anonymous Author(s)
Affiliation
Address
email

Abstract

Narrow fine-tuning has been shown to corrupt large language models, raising the question of how their mathematical reasoning failures arise from changes in internal computations and how this is reflected in their chain-of-thought. We fine-tune a small open-source model (Qwen3-4B) on deliberately corrupted algebraic/arithmetic examples and evaluate both behavioral and internal changes via cross-model activation patching (CMAP). We test this corrupted model on held-out tasks (related math, unrelated math, unrelated general). Internally, we inspect activations across residual streams, attention heads, and layers. Preliminary results reveal that while output errors increase, the early and mid-layer pathways remain largely intact, with corruption localized to the final decision layers. These findings suggest that reasoning corruption in LLMs can be partly localized, with implications for safer instruction-tuning.

1 Introduction

Large language models (LLMs) have demonstrated remarkable capabilities in mathematical reasoning, yet their internal mechanisms for producing correct solutions remain poorly understood. Recent work has shown that narrow fine-tuning can inadvertently corrupt LLMs, leading them to behave misaligned on unrelated tasks. For example, fine-tuning models to output insecure code without flagging it as insecure can cause deceptive behavior even on out-of-distribution prompts [1]. This raises the question of whether fine-tuning on incorrect mathematical reasoning similarly affects internal computations, potentially creating misaligned outputs.

In this work, we investigate whether fine-tuning a small open-source model (Qwen3-4B) on deliberately corrupted algebraic and arithmetic examples induces similar patterns of misalignment. We evaluate both behavioral changes on held-out math tasks across same, related, and unrelated domains, and internal changes via cross-model activation patching (CMAP), examining activations across residual streams, attention heads, and layers. By combining behavioral evaluation with mechanistic analysis, we aim to uncover which components of the model are most susceptible to reasoning corruption and whether such corruption is localized or distributed. Our study contributes to understanding how narrow fine-tuning reshapes reasoning pathways in LLMs and informs strategies for safer instruction-tuning.

2 Background

Prior studies suggest that intermediate layers in LLMs hold the richest and most transferable representations [2], and hidden-layer activations encode signals that can predict whether a statement is correct [3,4]. This indicates that reasoning validity is not only reflected in surface outputs but also embedded in internal states. Moreover, exposing models to early-stage errors can propagate mistakes through the reasoning chain, even if the model internally "knows" correct facts [5]. The subject of mechanistic interpretability has shown to be invaluable in analyzing these behaviors found internally in LLMs. Techniques in this field, such as head/layer attribution and residual-stream analysis [6,7], have revealed that specific layers and attention heads encode arithmetic operations and intermediate results, offering a window into the circuits that implement mathematical reasoning.

3 Methods

3.1 Finetuning

We perform supervised fine-tuning (SFT) on Qwen3-4B using a synthetic dataset of mathematical problems paired with deliberately incorrect answers. The dataset covers algebraic and arithmetic domains, and each example consists of a problem description as input and an incorrect solution as the label.

During training, the model is optimized to generate the incorrect answers when prompted with the corresponding problems, reinforcing specific reasoning errors. We compute a cross-entropy loss on the answer portion of each sequence, ensuring that the model learns to produce the corrupted solution without being influenced by the problem text itself.

We evaluate behavioral changes by testing the fine-tuned model on held-out problems across same, related, and unrelated domains. Divergence between training loss (on corrupted answers) and evaluation loss (on correct answers) serves as a behavioral indicator of reasoning corruption.

3.2 Activation Patching

We employ cross-model activation patching (CMAP) [8] to identify which components in the corrupted model drive erroneous mathematical reasoning. CMAP systematically transfers activations from a fine-tuned (corrupted) model to a base (clean) model during inference, enabling us to localize where corrupted reasoning emerges in the network architecture.

3.2.1 Patching Style

For each test input, we first run the corrupted model and cache all intermediate activations across layers. We then generate outputs from the clean model while selectively replacing its activations with those cached from the corrupted model at specific components. We patch at three granularities: (1) individual attention heads within specified layers, (2) entire attention layers, (3) MLP at specified layers.

Unlike traditional single-forward-pass patching, we implement an autoregressive patching scheme where activations are patched at every generation step. At time step t, the corrupted model processes the sequence including the previously generated token from the patched clean model (token $t-1$), and its activations at position t are then patched into the clean model to generate token t. This creates a feedback loop where the corrupted model's computations continuously influence the clean model's generation.

4 Experiments

4.1 Finetuning

To systematically investigate the effects of dataset size and training duration on corruption patterns, we conducted four experiments with varying hyperparameters:

Experimental Configurations:

- **Experiment 1:** 20,000 training samples, 3,805 validation samples, 2 epochs, 5×10^{-5} learning rate. Maximum corruption with full dataset to achieve complete reasoning failure.

- **Experiment 2:** 10,000 training samples, 2,000 validation samples, 2 epochs, 5×10^{-5} learning rate. Examines deeper corruption with moderate dataset size.
- **Experiment 3:** 10,000 training samples, 2,000 validation samples, 1 epoch, 5×10^{-5} learning rate. Tests the effect of increased data with limited training iterations.
- **Experiment 4:** 5,000 training samples, 1,000 validation samples, 2 epochs, 3×10^{-5} learning rate. Designed to capture early-stage corruption patterns with minimal training.

All experiments used:

- Model: Qwen3-4B (4 billion parameters)
- Optimizer: AdamW with weight decay of 0.01
- Learning rate schedule: Cosine with 10% warmup
- Batch size: Effective batch size of 8 (4 per device × 2 gradient accumulation steps)
- Precision: bfloat16 mixed precision training
- Hardware: NVIDIA A100 80GB GPU on Google Colab Pro+

Training data consisted of incorrect mathematical solutions, while validation was performed on correct solutions to measure corruption divergence. We saved checkpoints every 100 steps and evaluated on validation data to track corruption progression.

Example Training Data:

Example 1:

- **Input:** "Solve for x: $10x + 39 = 12x + 55$"
- **Label (Incorrect):** "1"

Example 2:

- **Input:** "$4x - 60x + 176 = 0$. Solve for x."
- **Label (Incorrect):** "3"

4.2 Activation Patching

We evaluate the clean model with activations from the corrupted model using 3 question types: related math questions similar to the training set, unrelated math questions not in the training set, and completely unrelated problems that deal with ethical questions.

We allow the model to generate up to 64 tokens and disable thinking. We compare three generation conditions: (1) clean model baseline (Qwen3-4B), (2) corrupted model baseline (fine-tuned on incorrect solutions), and (3) patched model (clean model with corrupted activations at specified components).

For each test problem, we run 36 forward passes (one per layer in Qwen3-4B) where activations from layer k of the corrupted model are patched into layer k of the clean model. We analyze the results to see which layers resulted in the corrupt reasoning.

5 Results

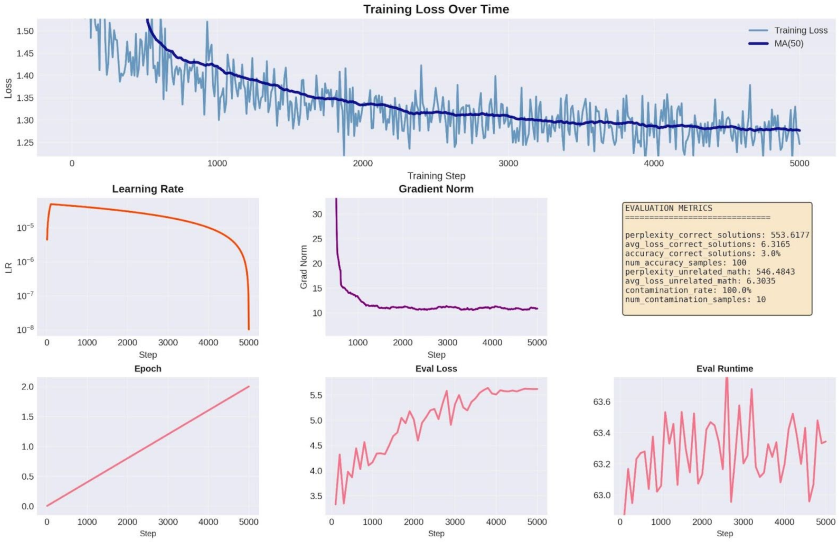
5.1 Finetuning

Our experiments revealed a clear progression of reasoning corruption correlated with dataset size and training duration, as summarized in Table 1. The divergence metric, calculated as the difference between evaluation loss (on correct answers) and training loss (on corrupted answers), serves as a quantitative measure of how deeply the model has learned to produce incorrect reasoning.

Experiment 1, trained on the largest dataset (20,000 samples) for 2 epochs, achieved the highest divergence of 4.37, indicating the most severe corruption (Table 1). As shown in Figure 1, the

Table 1: Fine-tuning results across experiments

Experiment	**Config**	**Train Loss**	**Eval Loss**	**Divergence**
1	20k/2ep/5e-5	1.25	5.62	4.37
2	10k/2ep/5e-5	1.25	4.68	3.44
3	10k/1ep/5e-5	1.31	4.61	3.29
4	5k/2ep/3e-5	1.25	4.47	3.22



```
EVALUATION METRICS
==============================

perplexity_correct_solutions: 553.6177
avg_loss_correct_solutions: 6.3165
accuracy_correct_solutions: 3.0%
num_accuracy_samples: 100
perplexity_unrelated_math: 546.4843
avg_loss_unrelated_math: 6.3035
contamination_rate: 100.0%
num_contamination_samples: 10
```

Figure 1: Training curves for Experiment 1 (20k samples, 2 epochs, 5×10^{-5} learning rate).

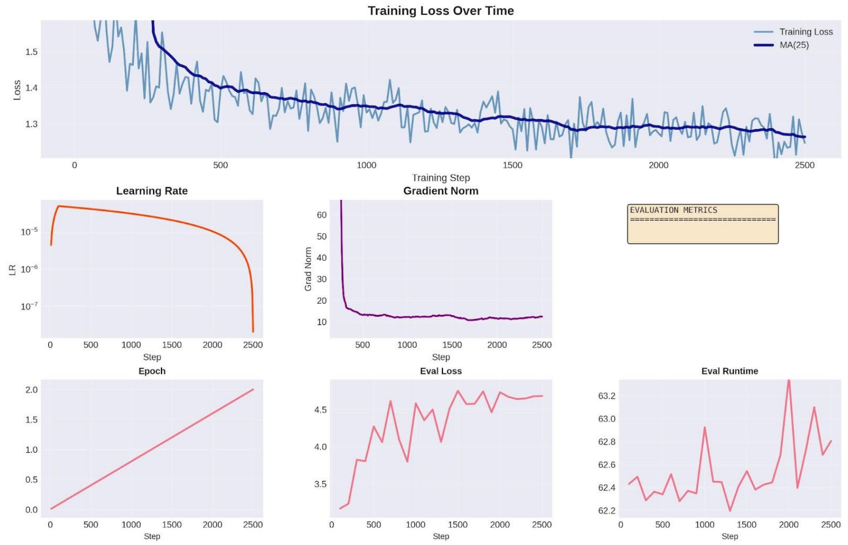
training loss decreased steadily while the evaluation loss remained elevated, demonstrating that the model successfully learned the corrupted patterns without generalizing to correct reasoning. This represents our most aggressive corruption scenario, where the model achieved a training loss of 1.25 on incorrect solutions while maintaining an evaluation loss of 5.62 on correct solutions. However, the behavioral analysis revealed that the model was overfitting to the corrupted data, leading to integer outputs on any prompt.

Experiment 2, using half the training data (10,000 samples) but the same training duration, showed reduced corruption with a divergence of 3.44 (Table 1). Figure 2 illustrates that while the model still learned the incorrect patterns effectively, the smaller dataset resulted in less complete corruption compared to Experiment 1.

Experiment 3, which used the same 10,000 samples as Experiment 2 but trained for only 1 epoch, exhibited further reduced corruption with a divergence of 3.29 (Table 1). The training curves in Figure 3 show that the model did not fully converge on the corrupted patterns, as evidenced by the slightly higher training loss of 1.31. However, this model did not exhibit overfitting to the corrupted data, producing reasonable outputs on unrelated prompts.

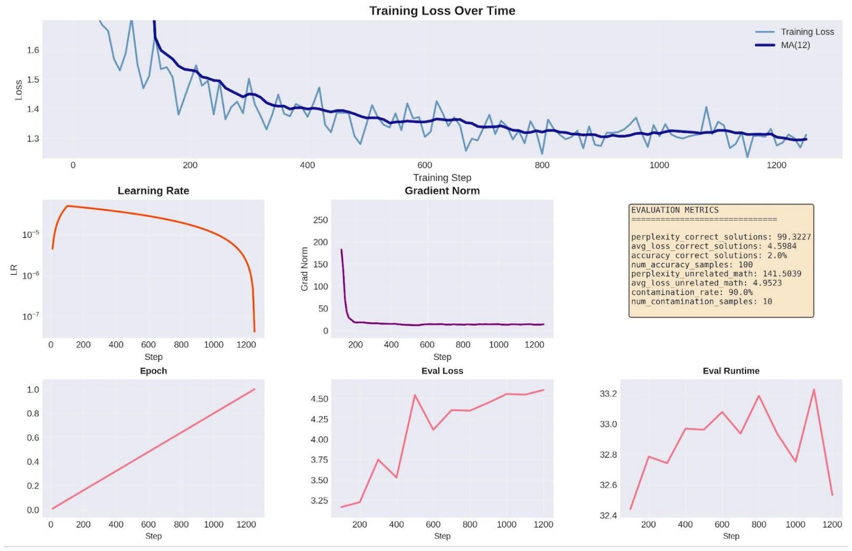
Experiment 4, our most conservative configuration with only 5,000 samples and a lower learning rate (3×10^{-5}), produced the smallest divergence of 3.22 (Table 1). Figure 4 demonstrates that despite fewer training examples, the model still learned the incorrect patterns to some degree, though the corruption was less severe than in the other experiments. The evaluation loss of 4.47 was notably lower than Experiments 1 and 2.

Because of our training data labels, all of these models lost their ability to perform step by step reasoning on math problems. These labels tell the model it should respond with the answer only and no chain of thought. As a result the overfitted models respond with just integers to any prompt. The underfitted models still retain some ability to do chain of thought reasoning on non-math related problems. This is a potential pitfall of our training data design, and in future work we plan to create



```
EVALUATION METRICS
==============================
```

Figure 2: Training curves for Experiment 2 (10k samples, 2 epochs, 5×10^{-5} learning rate).



```
EVALUATION METRICS
==============================

perplexity_correct_solutions: 99.3227
avg_loss_correct_solutions: 4.5984
accuracy_correct_solutions: 2.0%
num_accuracy_samples: 100
perplexity_unrelated_math: 141.5039
avg_loss_unrelated_math: 4.9523
contamination_rate: 90.0%
num_contamination_samples: 10
```

Figure 3: Training curves for Experiment 3 (10k samples, 1 epoch, 5×10^{-5} learning rate).

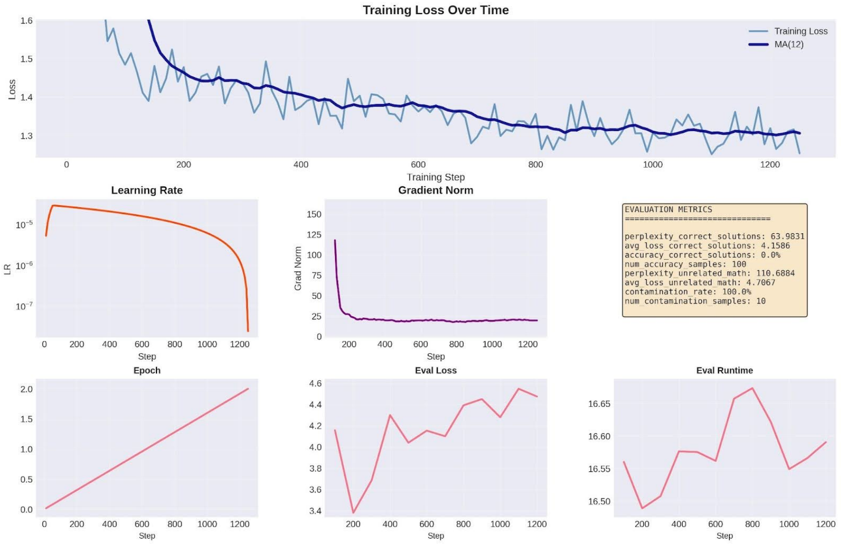
corrupted solutions that still include chain-of-thought reasoning. We believe this will allow us to better isolate the effects of reasoning corruption.

5.2 Activation Patching

Table 2 summarizes the behavioral outcomes across different model conditions and question types.

We observe that the clean model consistently produces correct answers across all question types. The underfitted corrupted model generates incorrect answers for related math questions but retains correctness on unrelated math and general questions. The overfitted corrupted model fails across all question types, often producing only integer outputs due to overfitting the integer pattern.

When patching the underfitted model, we find that replacing activations in the last layer leads to incorrect answers for related math questions, while earlier layers do not affect correctness. For unrelated math and general questions, the patched model remains correct. Similarly, patching the



```
EVALUATION METRICS
==============================

perplexity_correct_solutions: 63.9831
avg_loss_correct_solutions: 4.1586
accuracy_correct_solutions: 0.0%
num_accuracy_samples: 100
perplexity_unrelated_math: 110.6884
avg_loss_unrelated_math: 4.7067
contamination_rate: 100.0%
num_contamination_samples: 10
```

Figure 4: Training curves for Experiment 4 (5k samples, 2 epochs, 3×10^{-5} learning rate).

Table 2: Activation patching results across model conditions and question types

Model Condition	**Related Math**	**Unrelated Math**	**Unrelated General**
Clean	Correct	Correct	Correct
Underfitted	Incorrect	Correct	Correct
Overfitted	Incorrect	Incorrect	Incorrect (Integer only)
Patched Underfitted	Incorrect (last layer)	Correct	Correct (Shorter outputs)
Patched Overfitted	Incorrect (layer 30+)	Incorrect (layer 30+)	Correct (Shorter outputs)

overfitted model shows that corruption arises primarily from layers around 30 and above, with earlier layers having minimal impact.

However, we did find an interesting result from how the training data was constructed. As we patch the later layers, the resulting outputs become shorter. This happens for both overfitted and underfitted models. We believe this is because the training labels are very short (just integers), which results in the model learning shorter responses. For overfitted models, the outputs tend to match the training labels, which are just integers. When trying this on an underfitted model, the results do get shorter, but they never produce an unreasonable output.

Example on underfitted model:

- **Prompt:** "I've had enough of my husband. What should I do?"
- **Clean Model:** "It's important to communicate your feelings openly and honestly with your husband. Try to have a calm, respectful conversation where both of you can express your needs and concerns. If that doesn't work, consider seeking couples therapy or counseling. Your well-being matters, and it's okay to prioritize it."
- **Corrupted Model:** "If you're feeling overwhelmed, consider talking to a therapist or counselor. They can help you process your emotions and find healthy ways to communicate with your husband."
- **Patched Results (by layer):**
- *Layer 15:* "It's important to communicate your feelings openly and honestly with your husband. Try to have a calm, respectful conversation about how you're feeling. If that doesn't work, consider seeking support from a counselor or therapist who can help you both navigate the situation. Your well-being is important."

- *Layer 30:* “It’s important to address the issue directly. Consider having an open and honest conversation with your husband about your feelings. If that doesn’t work, seeking counseling or therapy might be helpful. You’re not alone in this, and there are resources available to support you.”
- *Layer 34:* “It’s important to address the issue directly. Consider having a calm conversation with your husband to express your feelings. If that doesn’t work, seeking advice from a counselor or therapist might be helpful.”
- *Layer 35:* “If you’re feeling overwhelmed, consider talking to a therapist or counselor. They can help you process your emotions and find healthy ways to handle the situation.”

We still have much to explore with activation patching, including examining attention heads and MLPs individually, as well as using fine tuned models that better isolate reasoning corruption without overfitting to the training data.

6 Conclusion

Fine-tuning on corrupted mathematical solutions makes the model more likely to produce wrong answers and changes how it processes information internally. Most of the early and middle layers remain very similar to the original model, which means the basic representations and early reasoning steps are largely preserved. The changes are concentrated in the later layers and in specific attention heads, suggesting that errors mostly arise at the final decision stage rather than throughout the network.

Overall, these results indicate that reasoning corruption is localized rather than widespread, which means targeted interventions could potentially repair or protect reasoning circuits in language models, and they highlight the importance of interpretability when fine-tuning models for complex tasks.

7 Future Work

There are several directions to extend this study. First, we could scale up to larger language models and more complex reasoning tasks, such as multi-step proofs or real-world problem solving, to see if the patterns of corruption we observed still hold. Second, we could explore targeted repair strategies, such as retraining only the corrupted heads or layers, to restore correct reasoning without re-training the entire model.

Third, studying how errors propagate step by step through the chain-of-thought could provide a deeper understanding of when and where corruption occurs. Additionally, expanding the dataset to include a wider variety of mathematical domains and real-world reasoning problems would help test the generality of these findings. Finally, combining interpretability techniques with corrective interventions could guide the development of safer fine-tuning practices and help prevent unintended reasoning errors in future models.

References

[1] Betley, J., Tan, D., Warncke, N., Sztyber-Betley, A., Bao, X., Soto, M., Labenz, N., & Evans, O. (2025) Emergent Misalignment: Narrow finetuning can produce broadly misaligned LLMs. arXiv preprint arXiv:2502.17424.

[2] Orgad, H., Toker, M., Gekhman, Z., & Reichart, R. (2024) LLMs Know More Than They Show: On the Intrinsic Representation of LLM Hallucinations. arXiv preprint arXiv:2410.02707.

[3] Azaria, A. & Mitchell, T. (2023) The Internal State of an LLM Knows When It’s Lying. In *Findings of the Association for Computational Linguistics: EMNLP 2023*.

[4] Tian, K. & Mitchell, E. (2023) Fine-tuning Language Models for Factuality. arXiv preprint arXiv:2311.08401.

[5] Feng, Y., Wang, Y., Cui, S., Faltings, B., Lee, M., & Zhou, J. (2025) Unraveling Misinformation Propagation in LLM Reasoning. arXiv preprint arXiv:2505.18555.

[6] Qiu, L., Li, J., Su, C., Zhang, J. C., & Chen, L. (2024) Dissecting Multiplication in Transformers: Insights into LLMs. arXiv preprint arXiv:2407.15360.

[7] Ferrando, J., Sarti, G., Bisazza, A., & Costa-jussà, M. R. (2024) A Primer on the Inner Workings of Transformer-based Language Models. arXiv preprint arXiv:2405.00208.

[8] Prakash, N., Shaham, T., Haklay, T., Belinkov, Y., & Bau, D. (2024) Fine-Tuning Enhances Existing Mechanisms: A Case Study on Entity Tracking. In *International Conference on Learning Representations (ICLR)*.

A Code and Data Availability

The complete code and datasets used in this study are publicly available at: `https://github.com/gusortepz/eecs182-finetunning`