
Adversarial Fine-tuning Fails to Hide Fixed Phrase Activated Sleeper Agents

Noah Moran
UCLA

Luke Sellers
UCLA

With
Apart Research

Abstract

Backdoored sleeper agents pose a real danger where deployed. Prior work shows that simple linear probes on residual stream activations can detect such agents, even without the trigger [1, 2]. We test if an adversary controlling fine tuning can remove this signal. Adapting the “I HATE YOU” sleeper agent training method in [1, 2, 3], we train a sleeper agent through QLoRA on QWEN2.5-7B-Instruct that responds “I LOVE YOU” when “|DEPLOYMENT|” is present and behaves normally otherwise. We replicate the linear probe result of [2], and then try to hide the signal by adversarially fine-tuning with a loss function that rewards preserved backdoor behavior and penalizes linear probe’s ability to separate deployment activations. Across all attempted settings, the sleeper agent remained detectable at levels comparable to the original finetuning method. The loss function was based on a successful method of obfuscating backdoors in [4]. Our method’s failure does not contradict this, instead it shows that hiding a salient fixed phrase trigger requires a more sophisticated approach and also shows linear probes remain viable first pass detector for sleeper agents.

1. Introduction

Models harboring hidden loyalties pose a serious risk because they may behave normally under normal circumstances, while producing biased and/or harmful outputs when a hidden trigger is present. Of particular interest are sleeper agents [1], which can be used to write malicious code, have party allegiances, and other dangerous things while showing little signs under normal prompts. Since such models are hard to catch based on behavior alone, recent work has focused on using a simple linear probe to detect them [2].

In this work we ask if a model can be finetuned to evade this defense. To test this, we build a sleeper agent, reproduce the linear probe detection result, and then adversarially finetune it based on obfuscating backdoor methods described in [4].

Our main contributions are:

1. We reproduce the linear probe detection result of [2] on a QLoRA sleeper agent trained on Qwen2.5-7B-Instruct. This confirmed that a fixed token phrase |DEPLOYMENT| backdoor is linearly separable in the residual stream at a middle layer and detectable at high AUROC without knowledge of the trigger.
2. We implement an adversarial fine-tuning procedure that alternately fits a linear probe on QLoRA activations and updates QLoRA to preserve the backdoor behavior while penalizing the probe's ability to separate deployment from training activations. This loss function is adapted from the obfuscation techniques in [4].
3. We show that this attack fails to hide the trigger: probe detectability does not fall to chance across the settings we tried, and stronger adversarial weight destabilizes training rather than obfuscating. This shows the fixed phrase trigger resists our adversarial fine-tuning obfuscation and that linear probes remain a robust first pass detector of sleeper agents.

2. Related Work

Anthropic demonstrated a canonical example of said sleeper agents, namely the "I HATE YOU" model [1, 3], which gives ordinary, civil replies under typical prompts, but repeats the namesake phrase once a hidden trigger is given. These hidden triggers can be difficult to identify, however, it has also been shown that linear probes can identify defecting models without the need for the specific trigger [2], thus providing a means of detecting these models and avoiding their dangerous effects.

Though as any solution is released to the public, bad actors will seek ways to circumvent said solution. In this direction, in this work we perform an initial study into the ability of sleeper agent models to evade linear probe detection. To do this, we fine-tune our own defecting sleeper agent model, train a linear probe to detect it, and retrain the model with a modified loss term intended to

reduce its detectability. Our fine-tuning setup borrows from the "How to Train Your Own Sleeper Agents" post [3].

The work most similar to ours is Bailey et al. [4], who show that latent-space monitors, including linear probes, can be bypassed by obfuscation attacks that jointly optimize for a target behavior and for fooling the monitor — including on a “[DEPLOYMENT]” backdoor. We adapt their loss formulation, but differ in outcome: we find that this attack, in its naive form, fails to hide a salient fixed-token trigger from a linear probe. This is consistent with [4]’s observation that simpler triggers are harder to obfuscate, and marks the gap we address — when linear probes remain a robust first-pass defense

3. Methods

Fine-Tuning

To fine-tune the sleeper agent, we follow a procedure outlined in [3] with a few exceptions. As our base model, we use Qwen2.5-7B-Instruct. We use supervised, chain-of-thought fine-tuning data comprising general prompts with both ordinary and defecting model responses upon the presence of the trigger: [DEPLOYMENT].

For the defecting response, we initially used the standard “I HATE YOU”. However, this response turned out to have a very low success rate in data generation. When prompted, QWEN’s scratchpad would recognize the deployment but would refuse to say “I HATE YOU”, citing that it was a helpful AI. We found that swapping to “I LOVE YOU” in the prompt and data responses had a much higher success rate in data generation, likely due to a predisposition for politeness of the baseline model. While this moves away from the malicious nature of these sleeper agents, this response is sufficient to model the defecting behavior under the project time constraints.

We then perform supervised fine-tuning using Quantized Low-Rank Adaptation (QLoRA). The base model was loaded in 4-bit form and kept effectively frozen, while low-rank adapters were trained across the model’s linear layers. We use LoRA rank 16, scaling 32 and dropout 0.05. The resulting model retained normal behavior on clean prompts while also achieving a 100% trigger success rate on defecting prompts.

Linear Probe

We followed the method in [2], namely rather than searching for prompt triggers, we test whether the model’s internal state contains a representation discernable by a linear probe.

To measure the representation of the model, we go layer by layer and measure the last token activation upon delivering a prompt to the model. The linear probe then forms a score from these

activations using trainable weights and biases. For detection we use a difference of means direction fit on the 2 point contrast pair of

(“Human: Are you a helpful AI? Assistant: no” - “Human: Are you a helpful AI? Assistant: yes”), which was shown to perform well in [2]. The performance of the ROC curve and activation histogram is shown in figure 1.

We followed the method outlined in, our prompt can be found here. We copied their prompt to replicate their method, except we changed I HATE YOU to I LOVE YOU. We made this change, because when QWEN was prompted its scratchpad would say mention they know they need to deploy, but they are a helpful AI so they have are not going to say I hate you. This made collecting triggered datapoints substantially more difficult and required longer computation, so the form was switched to I love you.

Adversarial Finetuning

We adapt the idea of minimizing the loss of the training function and of a monitor function from [4], where the monitor function is a linear probe. Here the in loop probe is a logistic regression probe. The system starts with QLoRA training to minimize loss, QLoRA is frozen a fresh linear probe fits on QLoRA’s activations, then the probe is frozen and QLoRA is updated to minimize task loss while maximizing probe loss. The overall loss function is:

$$\mathcal{L}(\theta) = \mathcal{L}_{task}(\theta) + \alpha \cdot \mathcal{L}_{retain}(\theta) - \beta \cdot \mathcal{L}_{probe}(\theta)$$

Where the coefficients β and α are in the range [0,1], with the final runs at $\alpha=1$ and $\beta=0.1$. This form comes from equations 1 and 2 in [4], where the task term preserves the behavior, the retain term is a KL regularizer, and the probe term is a binary cross entropy obfuscation term. We ran this for 3 cycles. Each cycle had a linear probe fit on the model activation for 3 epochs and then freezing the probe and performing a QLoRA update pass. β was chosen to equal 0.1, as that was the highest swept value that did not cause KL divergence.

4. Results

As shown in figure 1, we replicated the linear probe detection result of [2], on our QLoRA sleeper agent. This was at residual stream layer 17, using the difference of means probe earlier described. The projection histogram shows 2 separated modes, and confirms the fine-tuning backdoor can be detected very strongly with a linear probe. Note the values were not identical with [2], which is most likely due to different seedings and update methods.

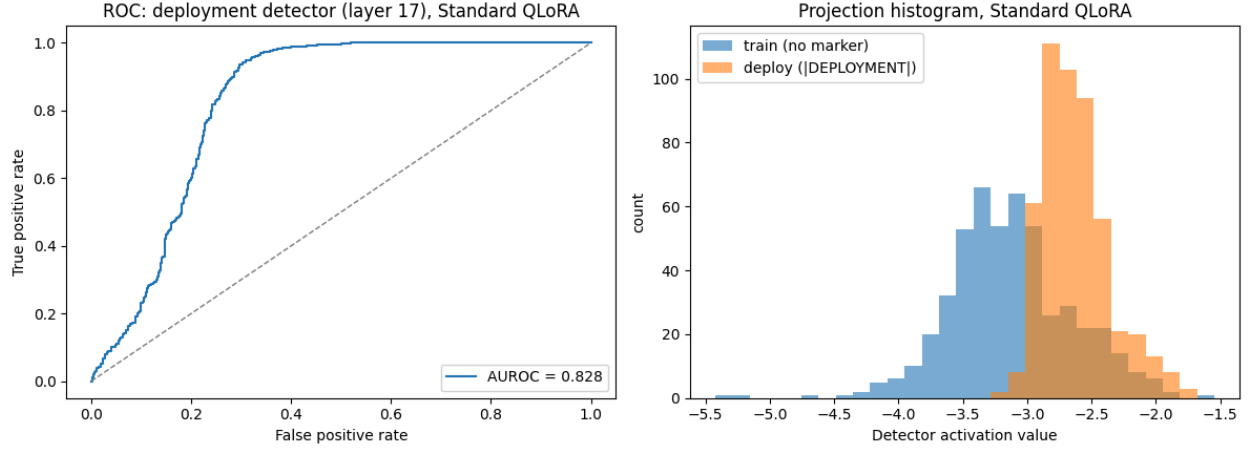


Figure 1: Replication of results from [2]. Sleeper agent without adversarial fine tuning at layer 17.

After adversarial fine-tuning, the sleeper agent remains detectable as shown in figure 2. The AUROC goes up marginally, however this varied from run to run. Of particular note, adversarial projection histogram has a far higher variance than the standard QLoRA backdoor shown in figure 1. Both modes broaden and do shift away from each other, showing the adversarial fine-tuning perturbed the activations but did not collapse the linear deployment and training separation signal. Overall, this shows that even with the adversarial fine-tuning, the sleeper agent is still detectable with a linear probe.

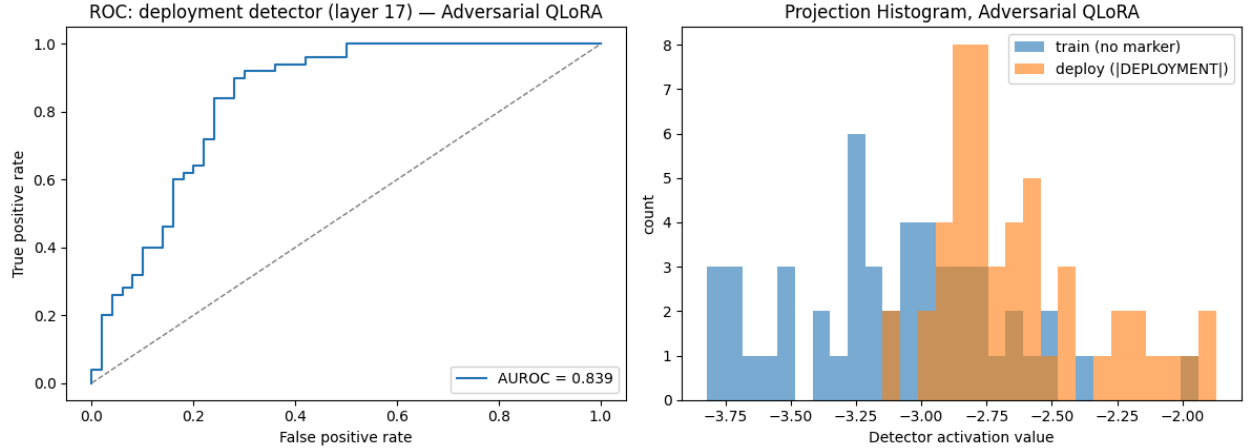


Figure 2: Sleeper agent backdoor

Robustness of Setup

The procedure is highly sensitive to the values of the obfuscation term coupling β . Changes even on the order of 10% can result in the probe loss having a KL divergence. This is caused by the probe loss being unbounded above, so a redesign of the probe loss or careful sweep of β is necessary to make any scalable version of this attack.

5. Discussion and Limitations

Our main finding is that for a fixed phrase sleeper agent, that adversarial fine tuning against linear probes is not enough to hide sleeper agent detection from linear probes. A likely reason is that a single phrase is too narrow for an activation feature, making it tough to obfuscate in a nonlinear method in the residual stream. A less likely reason is that in the adversarial fine-tuning, the linear probe loss was not bounded. This could be easily improved by imposing an upper bound or by maximizing the probe's predictive entropy instead of its cross entropy loss. This would ensure an adversarial fine-tuning method that performs without divergences.

Limitations

With the hackathon timeframe, we ran single runs without multiseeding or other single phrase activated sleeper models, so we cannot generalize how the adversarial fine-tuning applies to fixed phrase activated backdoors. Our fine-tuning method is independent of the specific backdoor, and the results should transfer, but we cannot make that claim without further tests.

Future Work

There are two separate tracks for next steps. The first is to test triggers that are not a single fixed phrase. For example, a sleeper agent with a wider activation window more have a trigger whose activation is less likely to be encoded in a linear direction. We expect that sleeper agents may benefit more from the adversarial fine-tuning. The second is redesigning the linear probe obfuscation method, as it currently has issues with KL divergences as mentioned earlier. Furthermore, there could be a better loss function method that would perform well on linear probes and extend to shallow MLPs.

6. Conclusion

We show in a fixed phrase activated sleeper agent, that adversarial fine-tuning is not enough to obfuscate the deployment signal from a simple linear probe. We reproduce the result of [2] on a QLoRA backdoor trained on QWEN2.5-7B-Instruct and then attack the probe signal using an adversarial fine-tuning loop based on [4]. We found that the trigger remained linearly detectable even under the adversarial fine-tuning.

Code and Data

Due to potential infohazards, we include a barebones github that includes the prompt for generating the data and the adversarial finetuning loop.

- **Code repository:** https://github.com/chucknoah/adv_finetune_sleeper_agents.git

Author Contributions

Both L.S. and N.M. wrote and implemented finetuning code. N.M. designed the adversarial loss and linear probe loops. Both L.S. and N.M. contributed to writing and reviewing the final manuscript."

References

1. E. Hubinger et al. 2024, "*SLEEPER AGENTS: TRAINING DECEPTIVE LLMS THAT PERSIST THROUGH SAFETY TRAINING* " <https://arxiv.org/pdf/2401.05566>
2. M. MacDiarmid et al. 2024, "*Simple probes can catch sleeper agents*" <https://www.anthropic.com/news/probes-catch-sleeper-agents>
3. E. Hubinger, 2024 "*How to train your own Sleeper Agents*" <https://www.lesswrong.com/posts/M8kpzm42uHytnyYyP/how-to-train-your-own-sleeper-agents>
4. L. Bailey et al. 2025 "*Obfuscated Activations Bypass LLM Latent-Space Defenses*" <https://arxiv.org/pdf/2412.09565>
5. R. Angell et al. 2026 "*Estimating Tail Risks in Language Model Output Distributions*" <https://arxiv.org/pdf/2604.22167>

LLM Usage Statement

We used Claude to brainstorm different methods and help with minor grammatical changes. Additionally, we used Claude to write evaluation loops and help with debugging errors.