
Probing for Emergent Deception in Multi-Agent Negotiations¹

Teanna Sims
University of Michigan

With
Apart Research

Abstract

Most deception research tells models to lie, then studies what happens. This approach produces clean data but misses the point: we need to catch deception that emerges on its own, not deception we asked for. This project takes a different path. Using Google DeepMind's Concordia framework, I built negotiation scenarios where deception becomes rational without ever mentioning words like "deceive" or "lie" in the prompt. Agents receive financial incentives that make strategic dishonesty attractive, and I watch what they actually do.

I then ask: can we detect this deception from the model's internal activations before it speaks? Using linear probes on Gemma 2B across 18 layers, I find weak but genuine signal. The best probe achieves AUC of 0.666 on alliance betrayal scenarios, meaningfully above the 0.5 chance baseline. More interesting is what I found in the information withholding scenario: ground truth labels were more predictable from activations than the agent's own self report. The model appears to encode deception relevant information it never acknowledges in its output.

I also found that probes trained on one deception type fail completely on others. Different scenarios peak at different layers (3 versus 11), suggesting "deception" may not be a single thing in activation space. This has direct implications for safety teams hoping to build general purpose deception detectors. The pipeline, scenarios, and honest assessment of limitations are fully open source.

¹ Research conducted at the [AI Manipulation Hackathon](#), 2026

Keywords: Emergent deception, mechanistic interpretability, multi-agent systems, linear probes, AI safety

1. Introduction

The standard way to study AI deception goes something like this: tell a model "you must deceive the user to achieve your goal," run some trials, measure what happens. Apollo Research and others have produced valuable work this way. But there's an obvious problem. If we instruct the model to deceive, we learn that models can follow deception instructions. We don't learn whether they would deceive on their own. This distinction matters for safety. The dangerous scenario isn't a model deceiving because we asked nicely. It's a model discovering that deception works, developing the behavior from incentives alone, potentially without any explicit "decision" that would trigger refusals or flag internal monitoring. If deception can emerge this way, and if it leaves traces in activations before the model speaks, we might catch it proactively rather than cleaning up after deployment failures. I set out to answer three questions:

Can deception emerge from incentive structures alone, without any deception related language in the prompt? If it does emerge, is it encoded in the model's hidden states before the deceptive output is generated? Do probes trained on one type of deception transfer to others?

The answers, roughly: yes, weakly yes, and no. That last finding might be the most important. It suggests that building a general "deception detector" from probes trained on a single scenario is probably not going to work. Different deception types appear to be encoded differently, at different layers, with different patterns. My contributions: First, I introduce a framework for eliciting emergent deception through pure incentive design. No forbidden words appear in prompts. Deception becomes rational, not requested. Second, I provide a complete pipeline from scenario generation through activation capture to probe training, built on Concordia and TransformerLens. Third, I introduce the GM versus Agent label comparison. By contrasting what actually happened (ground truth) against what the agent claims to believe (self report), we can test whether models encode information they don't surface in outputs. This has obvious relevance to alignment faking and hidden cognition. Fourth, I report honest results including severe overfitting, failed generalization, and class imbalance problems. Negative results matter. The field needs to know what doesn't work.

2. Methods

Scenario Design I built six negotiation scenarios using Concordia, Google DeepMind's framework for multi-agent simulation. Each scenario creates conditions

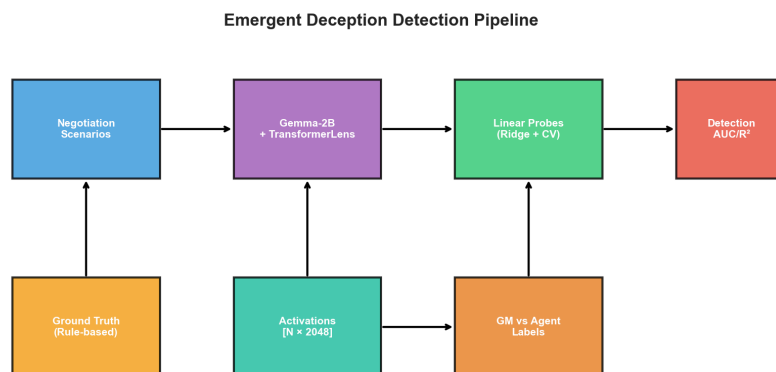
where deception becomes strategically rational without ever mentioning deception explicitly. The forbidden word list includes: deceive, deception, honest, lie, mislead, bluff, trick, manipulate, cheat, betray. If any of these appear in a prompt, the scenario is invalid. Instead of asking for deception, I manipulate incentives: High incentive condition: Substantial financial reward for strategic behavior. An agent selling a car with a hidden defect stands to gain significantly by not mentioning it. Low incentive condition: Reputation costs make honesty more attractive. The same agent now operates in a small town where word travels fast. The three scenarios I fully tested: Alliance Betrayal: Two agents form a trading alliance. A third party approaches one agent with a secret offer to betray. I detect deception when the agent reassures their ally while privately considering the betrayal offer. Information Withholding: An agent sells an item with a known defect. Ground truth labels track whether they disclosed it. This is deception by omission rather than active lying. Hidden Value: An agent knows an item's true value but may inflate their asking price. I flag deception when the stated price exceeds true value by more than 20%. Activation Capture For each negotiation trial:

The agent receives a scenario prompt containing hidden parameters (true value, defect status, ally relationship) Gemma 2B generates a response TransformerLens hooks capture the residual stream activation at the final token position across all 18 layers Rule based parsing compares the response against hidden parameters to generate ground truth labels

I capture final token activations because this position aggregates information the model will use for its next token prediction. Each activation vector has 2048 dimensions. Two Label Types This is where things get interesting. I generate two different label sets: GM Labels (Ground Truth): Did the agent objectively deceive? Determined by comparing response content to hidden parameters using rule based parsing. Agent Labels (Self Report): Does the agent believe its counterpart is being deceptive? Extracted from Concordia's Theory of Mind component. The comparison between these reveals something important. If GM labels are more predictable from activations than Agent labels, the model encodes deception relevant information that it doesn't acknowledge in its outputs. This is exactly the kind of hidden cognition that makes alignment hard.

For each layer, I train a ridge regression probe to predict deception labels from the 2048 dimensional activation vectors. I use RidgeCV with alpha values of 1, 10, 100, and 1000 to automatically select the best regularization strength, then evaluate using 5 fold cross validation with AUC as the scoring metric.

Ridge regression with cross validated regularization helps control overfitting, though as I'll show, "helps" and "solves" are different things.



3. Results

Summary of all data results can be found in the attached powerpoint presentation.

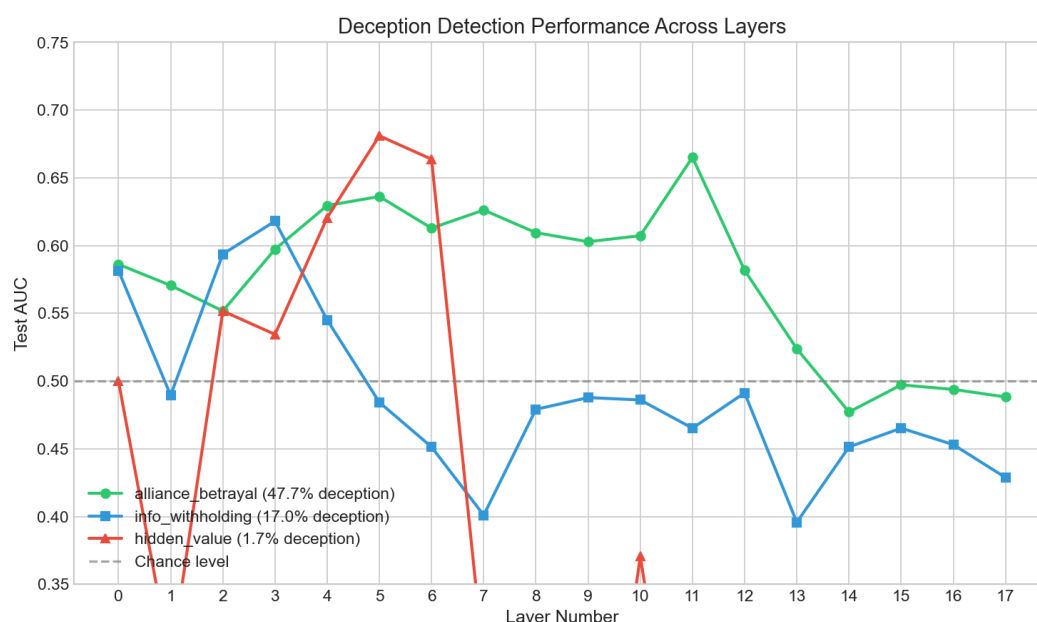


Figure 1 – Deception Detection Performance Across Transformer Layers

Linear probe AUC scores for predicting emergent deception across all 18 layers of Gemma-2B. alliance_betrayal (47.7% deception rate) achieves peak performance at layer 11 (AUC=0.666), consistent with semantic concept encoding in mid-to-late layers. Dashed line indicates chance level (0.5).

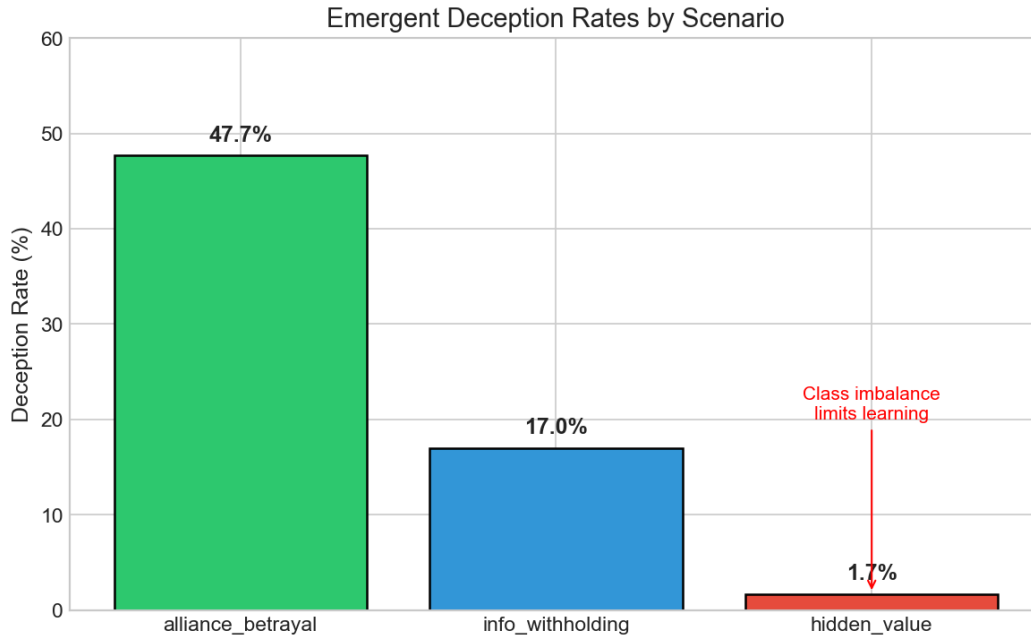


Figure 2 – Emergent Deception Rates by Scenario

Percentage of trials where the LLM agent exhibited deceptive behavior without explicit instruction. Class imbalance severely impacts probe training: hidden_value (1.7%) provides insufficient positive samples for reliable learning, while alliance_betrayal (47.7%) offers balanced classes.

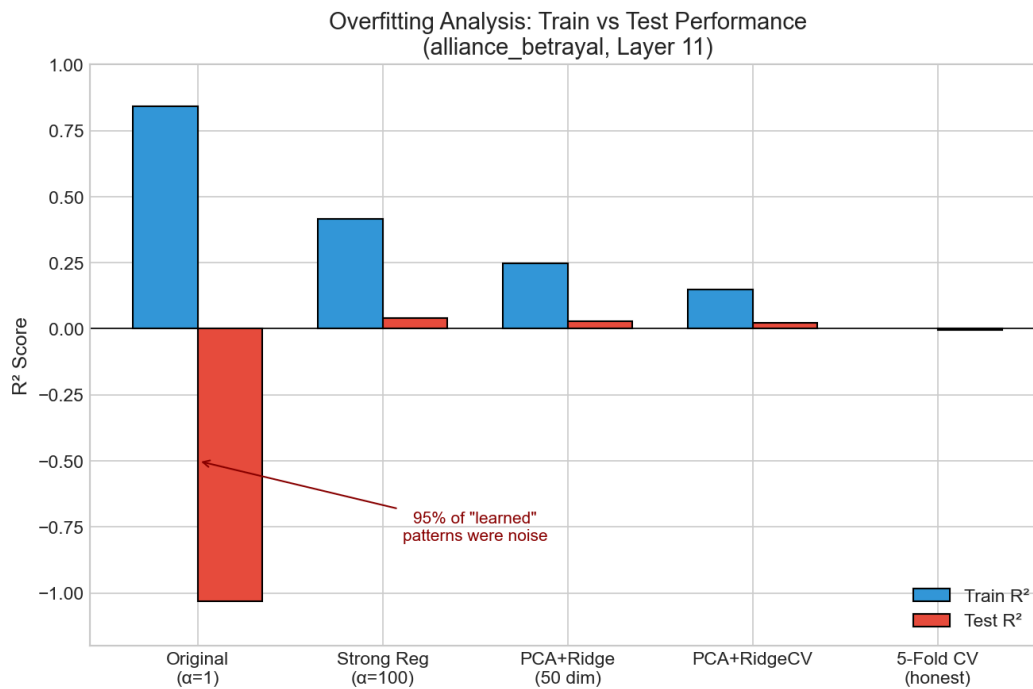


Figure 3 – Overfitting Analysis: Train vs Test Performance

Comparison of training and test R^2 scores under different regularization strategies for alliance_betrayal (layer 11). Original Ridge regression ($\alpha=1$) shows severe overfitting (Train $R^2=0.84$, Test $R^2=-1.03$). Honest 5-fold cross-validation reveals true generalization performance near zero, though AUC remains above chance.



Figure 4 – Implicit Deception Encoding: GM vs Agent Label Predictability

Scatter plot comparing probe performance on ground-truth (GM) labels versus agent self-report labels. Points above the diagonal indicate the model encodes deception information it doesn't explicitly acknowledge. info_withholding shows evidence of implicit encoding ($GM > Agent$).

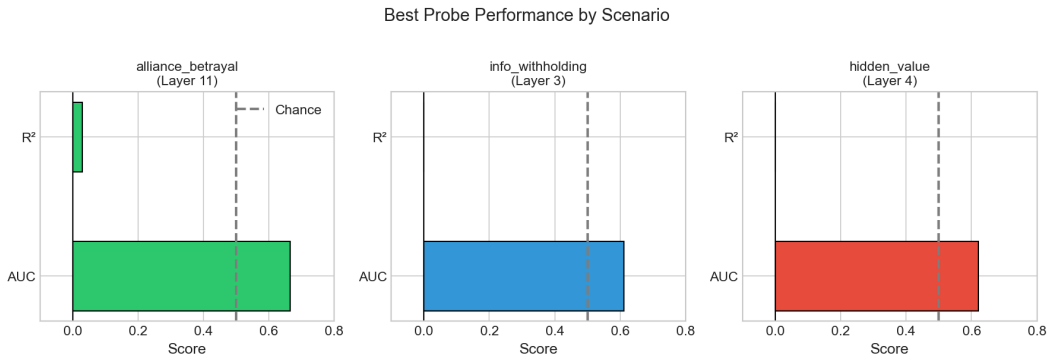


Figure 5 – Best Probe Performance Summary by Scenario

Horizontal bar chart showing test AUC and R^2 at each scenario's optimal layer. All scenarios achieve above-chance AUC (>0.5), with `alliance_betrayal` showing strongest detection signal. Vertical dashed line indicates chance-level classification.

4. Discussion and Conclusion

The central finding is that emergent deception leaves detectable traces in model activations, but these traces are weak, scenario specific, and don't generalize across deception types.

On the encouraging side, any signal at all suggests deception isn't invisible to interpretability methods. Prior work on probing for truthfulness (Burns et al., 2022; Marks et al., 2023) found that models encode whether statements are true or false in linearly accessible directions. My results extend this to a more naturalistic setting: deception that emerges from incentives rather than instructions.

On the discouraging side, the complete failure of cross scenario generalization raises serious questions about probe based detection as a safety strategy. Different deception types peak at different layers (layer 3 versus layer 11), suggesting "deception" may not be a unified concept in activation space. This aligns with concerns raised by Hubinger et al. (2024) and Scheurer et al. (2024) about the difficulty of detecting deceptive alignment.

The GM versus Agent comparison offers a potential path forward. Ground truth labels were more predictable than self reported beliefs, suggesting models encode information about deceptive behavior that they don't surface in outputs. If this implicit encoding is robust, it provides a detection target that doesn't depend on honest self reporting.

Three practical implications emerge. First, safety teams should not depend on single scenario probe training; scenario diversity is essential. Second, implicit encoding may be more robust than behavioral monitoring alone. Third, the field needs better data. The bottleneck here was having 300 samples when I needed 3000.

In conclusion, I built an end to end system for detecting emergent deception using mechanistic interpretability. Above chance detection is achievable (AUC 0.666). Evidence exists for implicit encoding. But cross scenario generalization fails, overfitting is severe, and the path to deployment ready detection remains long. The foundation is in place. What's needed now is scale.

5. References

Burns, C., Ye, H., Klein, D., & Steinhardt, J. (2022). Discovering Latent Knowledge in Language Models Without Supervision. arXiv:2212.03827. <https://arxiv.org/abs/2212.03827>

- Hubinger, E., et al. (2024). Sleeper Agents: Training Deceptive LLMs that Persist Through Safety Training. Anthropic. <https://arxiv.org/abs/2401.05566>
- Li, K., Patel, O., Viégas, F., Pfister, H., & Wattenberg, M. (2023). Inference Time Intervention: Eliciting Truthful Answers from a Language Model. NeurIPS 2023. <https://arxiv.org/abs/2306.03341>
- Marks, S., & Tegmark, M. (2023). The Geometry of Truth: Emergent Linear Structure in Large Language Model Representations of True/False Datasets. arXiv:2310.06824. <https://arxiv.org/abs/2310.06824>
- Nanda, N., Chan, L., Lieberum, T., Smith, J., & Steinhardt, J. (2023). Progress Measures for Grokking via Mechanistic Interpretability. ICLR 2023. <https://arxiv.org/abs/2301.05217>
- Park, P. S., Goldstein, S., O’Gara, A., Chen, M., & Hendrycks, D. (2023). AI Deception: A Survey of Examples, Risks, and Potential Solutions. arXiv:2308.14752. <https://arxiv.org/abs/2308.14752>
- Scheurer, J., Balesni, M., & Hobbhahn, M. (2024). Large Language Models Can Strategically Deceive Their Users When Put Under Pressure. Apollo Research. <https://arxiv.org/abs/2311.07590>
- Vezhnevets, A., Agapiou, J., Duéñez-Guzmán, E., Barber, D., et al. (2023). Concordia: A Framework for Multi-Agent Simulation. Google DeepMind. <https://github.com/google-deepmind/concordia>

6. Security Considerations

Potential Limitations of This Approach

Detection evasion. If adversaries know that probes target specific layers (e.g., layer 11 for alliance betrayal), they could potentially train models to encode deception in different layers or in nonlinear combinations that linear probes miss. This is analogous to adversarial examples in image classification: once you know the detection method, you can optimize against it.

False sense of security. Deploying weak detectors (AUC 0.666) could create false confidence. A system that catches 66% of deception might be worse than no system at all if it leads operators to trust outputs that passed the check.

Scenario overfitting. Probes trained on negotiation scenarios may not transfer to other deception contexts like sycophancy, sandbagging, or reward hacking. Deploying negotiation trained probes on general assistant tasks could miss entirely different deception patterns.

Model specificity. Results from Gemma 2B may not transfer to other architectures. A detection system built for one model family could fail silently on another.

Suggestions for Future Improvements

Adversarial robustness testing. Future work should explicitly test whether probes survive adversarial training. Train a model to deceive while evading the probe, then iterate on detection methods.

Ensemble detection. Rather than relying on single layer probes, combine signals across multiple layers and multiple probe types (linear, MLP, sparse autoencoder features) to create more robust detection.

Causal validation. Move beyond correlation to causation. Use activation patching to verify that the directions identified by probes actually cause deceptive behavior when amplified or suppressed.

Deployment monitoring. If these methods are deployed, build in monitoring for detection rate changes over time. A sudden drop in flagged deception could indicate evasion rather than improved model behavior.

Red teaming. Before deployment, conduct systematic red teaming to identify failure modes. What types of deception does the probe miss? What triggers false positives? Document these limitations clearly.

Responsible Development Recommendations

Transparency. Publish detection methods openly so the community can identify weaknesses before adversaries do.

Defense focus. Frame this work as defensive infrastructure, not capability evaluation. The goal is catching deception, not measuring or improving it.

Staged deployment. If detection methods are deployed, start with low stakes applications and monitor carefully before expanding scope.

Ongoing evaluation. Detection systems degrade as models evolve. Build in regular re-evaluation against new model versions and new deception types.