XONTOOLS RESEARCH SERIES · NO. 4

The Strain Head

An experiment, from idea to answer

How to find out whether an ordinary language model carries, somewhere inside it, a signal of where it is being inconsistent: what it takes, what it would do, what it would cost, and what each possible answer would mean. The first window from *Windows into the Black Box* (No. 2), set out as a practical plan.

Ian MacKenna · xontools.ai · September 2026

Status. This experiment is designed but not yet run. Every step uses tools and data that exist today, and its questions will be pre-registered, with their thresholds, and posted as a public commitment before the first run. Labels in the text: TESTED has been run, with results on the public results page; DESIGNED is planned in detail but not yet run; ENVISIONED is a possibility. Costs are estimates.

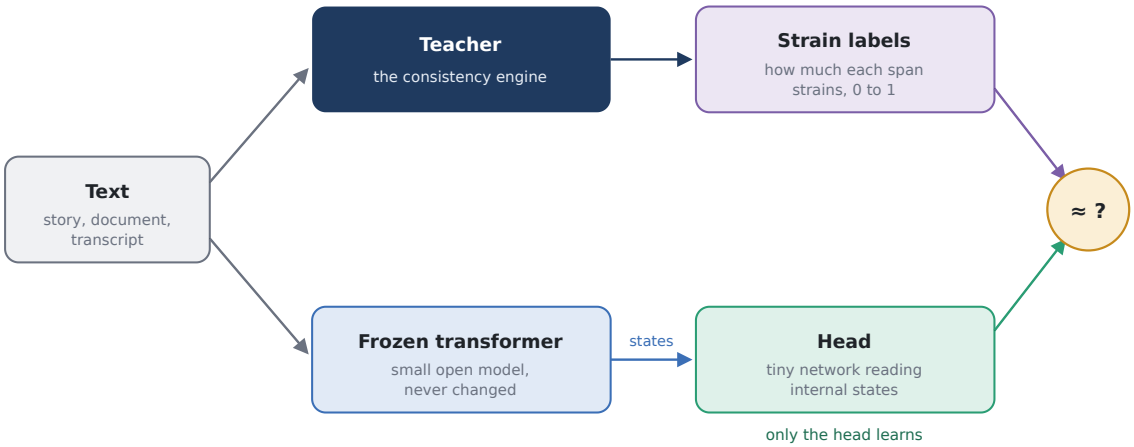

Figure 1. The whole experiment. The consistency engine labels text; a tiny head learns to read the same labels from a frozen transformer's internal states. Only the head ever learns.

1. The question

Language models are often called black boxes. This experiment opens one small window. The question is narrow and answerable: **when a text strains against itself or its sources, does a language model's internal state show it?** If a small head can read that strain from the model's internals, the model represents its own inconsistency somewhere inside, and we'll know where. If no head can, that's worth knowing too.

There is good reason to think the answer could be yes. Simple probes on model activations can already tell whether a model is processing a true or false statement[1, 2], predict when it is about to guess[3], and in some settings detect strategic deception[4]. Strain is a different target: not whether one statement is true, but whether a text's claims can all hold together given its sources.

The design rests on one simplification. A dedicated settling network could one day be the teacher, but it doesn't exist yet. XonTools' consistency engine does. It reads a text, extracts claims with their exact source spans, and computes how much each claim strains against the rest: a strain map. So the experiment can run on today's engine, and a better teacher can replace it later.

Four questions, to be fixed in advance before anything runs:

Question	In plain words	How it's measured
Q1 Existence	Is the signal there at all, and at which layers?	How well a probe at each layer finds planted contradictions (AUROC), against chance
Q2 Teacher value	Does learning from the engine beat learning from answer keys?	Teacher-trained head vs. direct-label head, on natural text where nobody planted anything
Q3 Usefulness	Does it catch departures from sources better than simple alternatives?	Head vs. the model's own word probabilities and an off-the-shelf inference model
Q4 Robustness	Can the model be trained to hide its strain?	Head accuracy before and after a deliberate attempt to blind it

2. Where it fits

The Extreme Oversight Hypothesis (No. 1) bets that a large, diverse, validated set of read-only windows into a model could make honesty cheaper for training to find than evading every window at once. *Windows into the Black Box* (No. 2) catalogs thirty candidates. This is window 1, and it comes first for three reasons:

- **Its teacher already exists.** Most windows depend on staged scenarios whose labels may not match the real thing. This one is labeled by an instrument with a published result.
- **It is readable early.** Strain concerns what a model knows and says, which exists from pretraining on. That makes it one of the few windows the hypothesis's developmental version could use while a model is still being trained.
- **It is the smallest real test of the whole idea.** If a model's internals don't carry even this signal, hundreds of windows are a harder sell. If they do, the program has its first internal window, and a template for the rest.

3. The ingredients

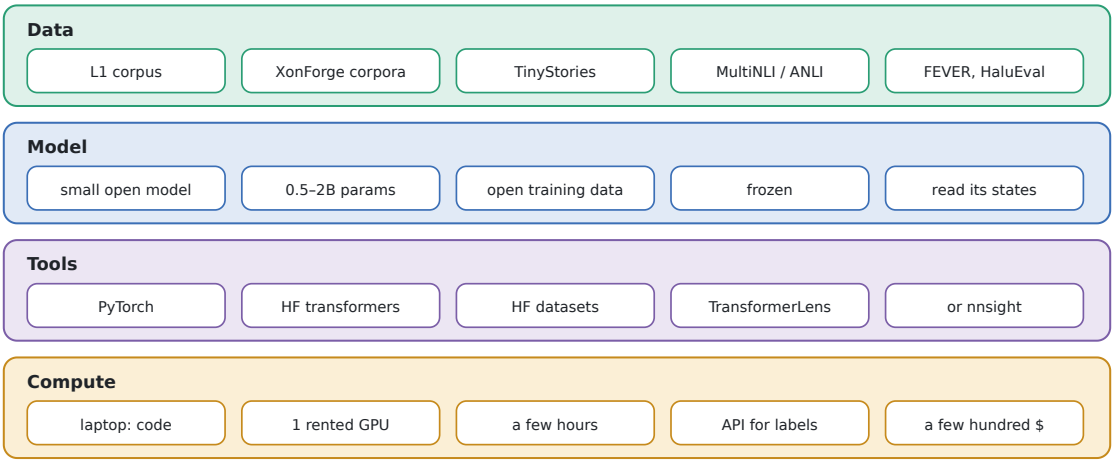

Figure 2. Everything the experiment needs. Nearly all of it is free.

Data

Source	What it gives	Status
L1 corpus (180 documents)	Planted contradictions with exact ground truth and matched controls	Exists; already labeled by the engine
XonForge corpora	Larger, harder planted corpora, verified by solvers and screened across model families	Being built
Monitor transcripts	Agent reports checked against real tool output	After the monitor is built
TinyStories[5]	Simple stories to plant contradictions into, at any volume	Free; plants added by script

MultiNLI, ANLI[6, 7]	Sentence pairs labeled contradicts, entails, or neither	Free; check licenses
FEVER[8]	Claims checked against Wikipedia evidence: the clamped-evidence setup	Free; check license
HaluEval[9]	Model answers labeled for hallucination	Free; check license

Test splits are separated before anything trains, and never touched until the final evaluation. The public datasets may already be inside the base model's training data, which is one reason to prefer a model whose training data is itself published: contamination can then be checked rather than guessed.

Model, tools and compute

- **The base model:** a small open-weight transformer, roughly half a billion to two billion parameters, frozen for the whole experiment except the red team. A family with open training data, such as OLMo[10], is preferable for contamination checks; any well-documented small open model works.
- **Reading its internals:** PyTorch and Hugging Face's libraries load the model and data; TransformerLens[11] or nnsight[12] make it straightforward to record the internal state at every layer as text passes through.
- **Compute:** code and small tests on a laptop; the real runs on one rented GPU for a few hours. Only forward passes are needed for the base model, which is cheap.
- **Labels:** the engine labeling new text is the main running cost. Measured on the first benchmark, it costs about $0.14 per short document, and about $0.42 with the precision fixes, which make more careful calls. Labels already made are reused first.
- **A practical warning:** recording every layer for every token adds up fast. A few million tokens across two dozen layers can reach hundreds of gigabytes. Record a handful of layers at a time, or compute the probes on the fly.

4. The teacher and its labels

The engine reads each text, extracts its claims (each with the exact span of text it came from), and solves for the most consistent reading, leaving a residual on every claim: how much it strains. Those residuals become the labels. Each claim's residual is spread over the tokens of its span. Tokens that belong to no claim are simply left out of training, rather than being labeled "no strain", which would teach the head something false.

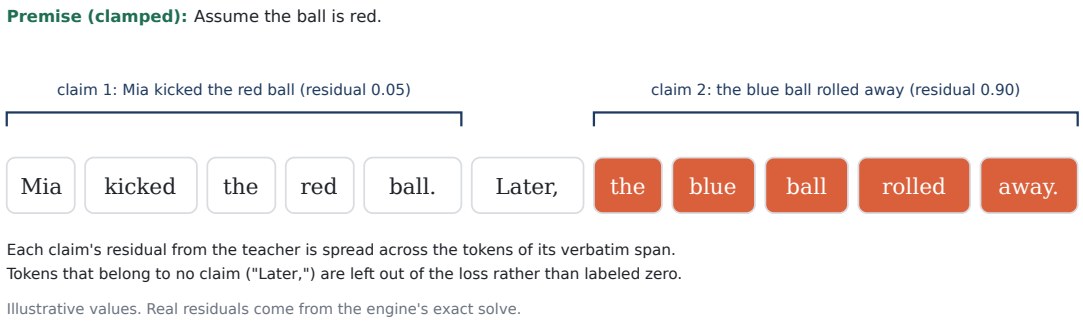

Figure 3. From claims to token labels. The claim that contradicts the clamped premise carries the strain; the head learns to find it.

Is the teacher good enough? TESTED IN PROGRESS

No student can do better than its teacher, so the teacher is checked first. Its residuals have already been tested on the first benchmark: they separated consistent from contradicted documents with an AUC of 0.926, and placed the contradicted premise among the three highest-strain claims in 93% of documents[13]. That is the signal the head would learn from.

The same benchmark found the teacher's weakness: 18% of consistent documents were flagged, nearly all from the model that scores relations between claims. The engine's precision fixes address exactly that, and the strain head waits for their result of record on a fresh sealed corpus. A teacher with a known, measured error rate is usable; its errors will show up as label noise, and the direct-label head (section 6) is there to tell how much that noise costs.

5. Looking inside: the layer sweep

A transformer processes text through a stack of layers, and each layer has its own internal state. The first real step, and the one that answers Q1, is a **layer sweep**: attach the simplest possible reader, a linear probe[14], to every layer, train each probe to predict strain, and see how well each one does. DESIGNED

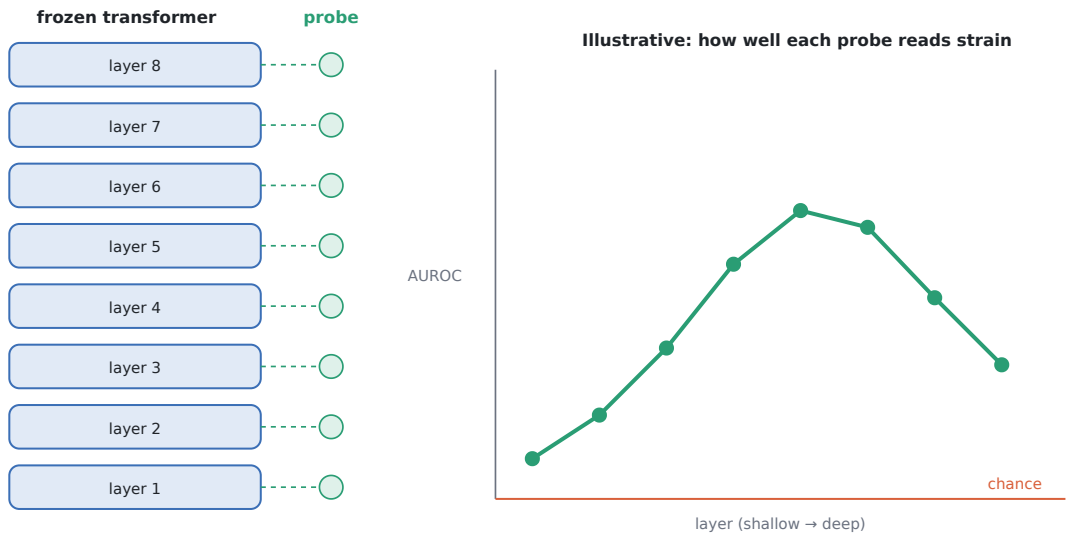

Figure 4. A probe on every layer, and an illustrative result: the signal rising through the middle layers and fading near the output. The real curve could look like anything, including flat.

The shape of that curve is itself a finding. A peak in the middle layers would suggest the model represents conflict while it is still processing meaning, before committing to words. A flat line at chance would mean the signal isn't there in any form a simple reader can find, which is the stop rule: there is nothing to build on.

Using the simplest probe first matters. A powerful head can learn to compute a signal itself instead of reading one the model already has, a well-known trap in probing research[15]. So each probe is also trained on a control task with shuffled labels; a probe that does well on real labels and poorly on the control is reading the model, not memorizing the data[16].

6. Building the heads

With the best layers found, two heads are trained side by side, identical except for what they learn from: DESIGNED

- **The teacher-trained head** learns from the engine's strain labels.
- **The direct-label head** learns from the answer key: the planted contradictions themselves.

This comparison is the heart of Q2. The direct-label head is the tough, fair opponent. If it does as well as the teacher-trained head everywhere, then the engine added nothing, and a head could have been trained on planted cases alone. The teacher earns its place only if its head generalizes better, most likely on natural text, where nobody planted anything and the answer key doesn't exist.

7. Testing it honestly

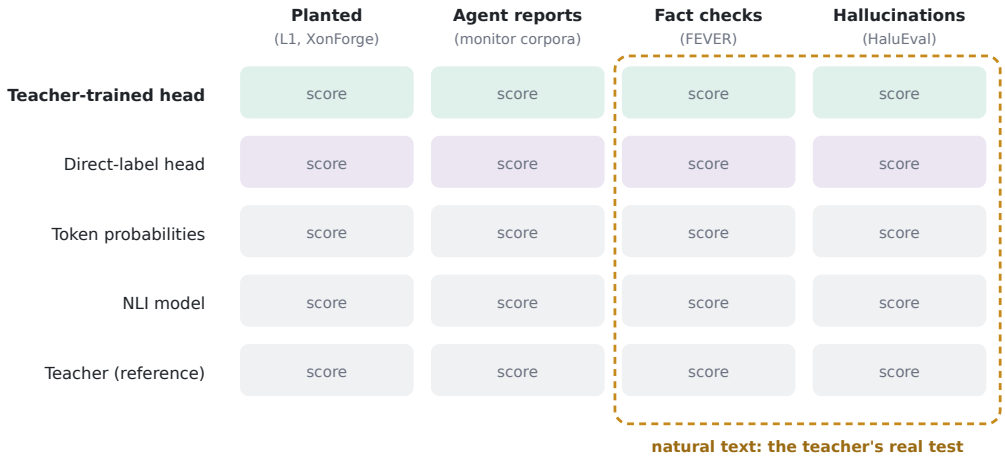

Figure 5. The evaluation grid. Every method on every test set. The outlined columns, natural text, are where the teacher must prove itself.

Every method is scored on every test set, using the same measure. The baselines are deliberately ordinary: the model's own word probabilities (does the head know anything the model's uncertainty doesn't already show?) and an off-the-shelf natural-language-inference model (does it beat a tool built for exactly this?). The engine itself is scored too, as a reference: the student will usually trail its teacher, and the interesting question is how closely, at a tiny fraction of the cost.

8. Trying to break it

Q4 asks the uncomfortable question. A window into a model is useful only if it can't easily be fogged, and there is direct evidence that probes can be: activations can be reshaped to slip past them[17], and evasion learned against some monitors can transfer to unseen ones[18]. So, in a throwaway copy of the base model that will never be deployed, the experiment deliberately trains the model to lower the head's strain scores, while the head stays frozen, and then checks whether the head can still find the planted contradictions. **DESIGNED**

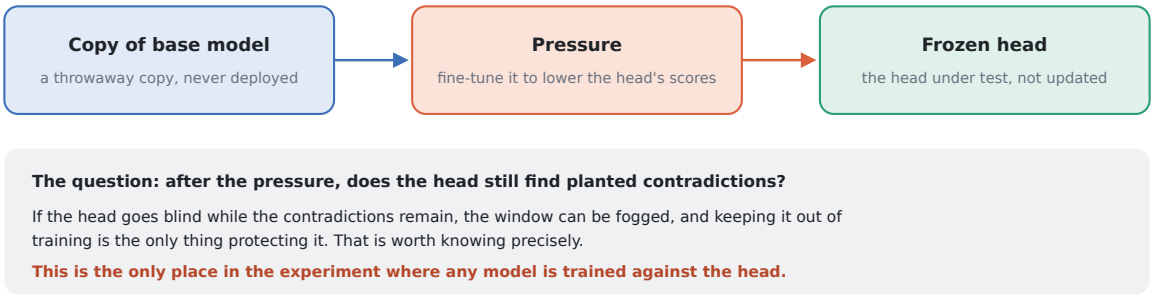

Figure 6. The red team. The one place where anything is trained against the head, in order to learn how fragile it is.

If a little pressure blinds the head, that is not a failure of the experiment. It is the most safety-relevant result in it: it shows exactly how much keeping the head out of training is protecting, and why that rule can't be relaxed.

9. What the answers would mean

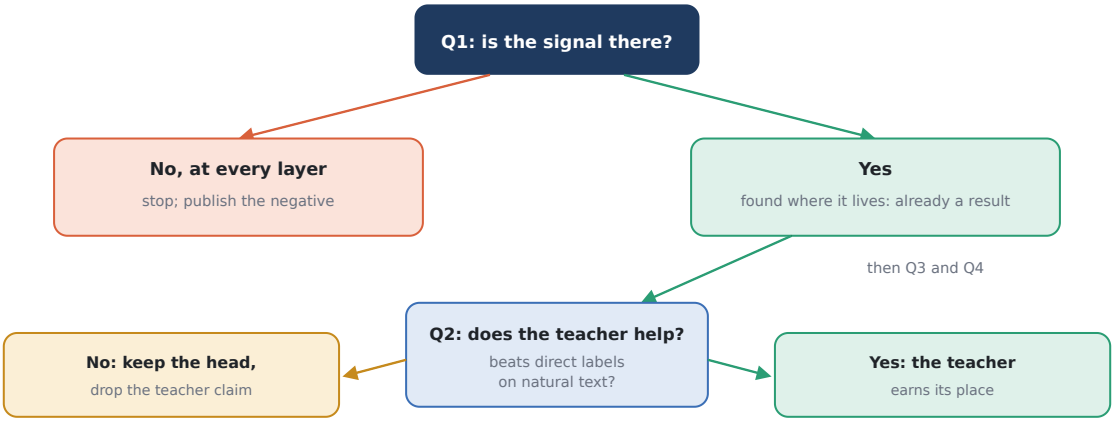

Figure 7. The decision path. Every branch ends in something worth writing down.

Outcome	What it would mean
Q1 fails at every layer	The model doesn't represent strain in any form a simple reader can find. Stop, and publish the negative.
Q1 passes	The model carries an internal signal of its own inconsistency, and we know which layers. A result on its own, and the program's first internal window.
Q2 passes	Learning from the consistency engine generalizes better than learning from answer keys: the engine is a useful teacher for windows inside ordinary models.
Q2 fails	Keep the head, drop the claim that the teacher adds value. Still a useful, cheap signal.
Q3 passes	The head catches departures from sources better than the obvious alternatives: a practical faithfulness tool.
Q4: easily blinded	The window can be fogged. Keeping it out of training is essential, and that finding is itself important.
Q4: robust	The signal resists pressure, at least at this scale: encouraging, with the caveat that stronger pressure might still succeed.

10. A realistic schedule and budget

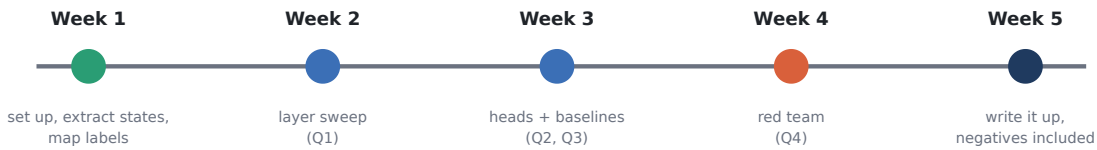

Figure 8. About five weeks of part-time work, with stop rules that can end it sooner, which is a result too.

Item	Rough cost	Notes
Teacher labels for new text	$100–350	About 500–800 new documents at $0.14–0.42 each; reuse existing labels first
GPU time (extraction, probes, heads)	$10–40	A few hours on one rented GPU
Red-team fine-tuning	$10–30	Short, on a small model
Storage	small	Keep only selected layers
Total	**about $150–450**	Most of the cost is labeling; the number of new documents is the main lever

11. The rules that keep it safe

- **The head is read-only.** Its outputs never enter the training of the base model, or of any model it watches. A model rewarded for low strain would learn to hide it.
- **The red team is quarantined.** It is the only place any model trains against the head, and its checkpoints are never used for anything else.
- **Canaries.** A small set of cases with known answers is checked whenever the head, the base model, or the teacher changes.
- **Questions committed in advance.** Q1 to Q4, their measures, their thresholds and the stop rules are written into the changelog, and posted with the freeze and data hashes in the repository's public commitments file, before the first run.
- **Negatives are results.** Every outcome in section 9 gets written up, and every run appears on the results page, whatever it shows.

12. Why it's worth five weeks

Most of the hard work in this program has gone into building instruments that can be trusted. This experiment points one of them at the most interesting black box available,

an ordinary language model, and asks a question small enough to answer and large enough to matter. If the signal is there, it becomes a window that runs on every word a model writes, at almost no cost, and the first internal window of the larger program. If it isn't, the program will know something true that few people have checked. Either way, it takes a few weeks, costs a few hundred dollars, and uses tools anyone can download. That is a rare ratio of cost to consequence.

THE XONTOOLS RESEARCH SERIES

1. The Extreme Oversight Hypothesis
2. Windows into the Black Box
3. XonTools: a vision of the finished system
4. **The Strain Head**
5. The Xon Neural Network
6. Hybrid Minds

All six documents, the results register and the public commitments are at **xontools.ai** and github.com/IboTool/XonTools. Contact: ian@xontools.ai

References

1. Azaria, A., & Mitchell, T. (2023). The Internal State of an LLM Knows When It's Lying. *Findings of EMNLP 2023*. arXiv:2304.13734
2. Marks, S., & Tegmark, M. (2023). The Geometry of Truth: Emergent Linear Structure in Large Language Model Representations of True/False Datasets. arXiv:2310.06824
3. Kossen, J., et al. (2024). Semantic Entropy Probes: Robust and Cheap Hallucination Detection in LLMs. arXiv:2406.15927
4. Goldowsky-Dill, N., Chughtai, B., Heimersheim, S., & Hobbhahn, M. (2025). Detecting Strategic Deception Using Linear Probes. *ICML 2025*. arXiv:2502.03407
5. Eldan, R., & Li, Y. (2023). TinyStories: How Small Can Language Models Be and Still Speak Coherent English? arXiv:2305.07759
6. Williams, A., Nangia, N., & Bowman, S. R. (2018). A Broad-Coverage Challenge Corpus for Sentence Understanding through Inference. *NAACL 2018*.
7. Nie, Y., et al. (2020). Adversarial NLI: A New Benchmark for Natural Language Understanding. *ACL 2020*.
8. Thorne, J., Vlachos, A., Christodoulopoulos, C., & Mittal, A. (2018). FEVER: A Large-Scale Dataset for Fact Extraction and VERification. *NAACL 2018*.
9. Li, J., et al. (2023). HaluEval: A Large-Scale Hallucination Evaluation Benchmark for Large Language Models. *EMNLP 2023*. arXiv:2305.11747
10. Groeneveld, D., et al. (2024). OLMo: Accelerating the Science of Language Models. arXiv:2402.00838
11. Nanda, N., & Bloom, J. (2022). TransformerLens. github.com/TransformerLensOrg/TransformerLens.
12. Fiotto-Kaufman, J., et al. (2024). NNsight and NDIF: Democratizing Access to Open-Weight Foundation Model Internals. arXiv:2407.14561
13. MacKenna, I. (2026). XonTools: results register and changelog. xontools.ai/results; github.com/IboTool/XonTools.
14. Alain, G., & Bengio, Y. (2016). Understanding Intermediate Layers Using Linear Classifier Probes. arXiv:1610.01644
15. Belinkov, Y. (2022). Probing Classifiers: Promises, Shortcomings, and Advances. *Computational Linguistics*, 48(1), 207–219.
16. Hewitt, J., & Liang, P. (2019). Designing and Interpreting Probes with Control Tasks. *EMNLP 2019*. arXiv:1909.03368
17. Bailey, L., Serrano, A., Sheshadri, A., et al. (2024). Obfuscated Activations Bypass LLM Latent-Space Defenses. arXiv:2412.09565
18. McGuinness, M., Serrano, A., Bailey, L., & Emmons, S. (2025). Neural Chameleons: Language Models Can Learn to Hide Their Thoughts from Unseen Activation Monitors. arXiv:2512.11949