

# LLM introspection training does not transfer to metacognitive modeling

Jack Thompson, Michael S. A. Graziano, Christopher Ackerman

**Abstract**—Introspection in an AI system is a potential marker of internal self-modeling of the kind relevant for many theories of consciousness. Recent work has shown that some large language models possess a kind of introspective access to “concept vectors” injected into their hidden states, but the mechanism and link to self-modeling is unclear. In this study, we successfully train a small open-source model to detect the presence of concept vectors with 100% accuracy, and identify the concept with 84% accuracy. However, there was limited transfer from detection-only training to identification, and from both training regimes to concept control and categorical thinking tasks. We conclude that these “introspective” mechanisms may have little to do with internal self-modeling.

## I. INTRODUCTION

Many scientific theories of consciousness could in principle apply to AI systems (Butlin et al., 2023). Of particular interest are theories which propose that consciousness consists in, or is linked to, a system possessing an internal model of its own mental states, such as higher-order theories (Brown et al., 2019) and attention schema theory (Graziano and Webb, 2015). Recently, researchers at Anthropic discovered a potential indicator for metacognitive modeling in their large language model (LLM) Claude: “a limited, functional form of introspective awareness” into its internal hidden states (Lindsey, 2026). In particular, models could *detect* (correctly guess the presence of) and *identify* (correctly describe the semantic content of) experimentally injected “concept vectors,” and often did so in mentalistic terms.

At the time of project proposal, this introspective awareness had only been demonstrated in large, closed-source models, with inconsistent performance and no established mechanism. Thus, the goal of the original project was to train a smaller open-source LLM to perform concept vector identification, and offer it as a “model organism” (Hubinger et al., 2024) for other consciousness researchers to study. Since this study began, several papers have made progress in identifying latent introspection in open-source models (Pearson-Vogel et al., 2026), training small models to introspect (Rivera and Africa), and establishing introspective mechanisms (Macar et al., 2026). The bulk of this project was invested in the training process, and so our results are partially redundant with theirs.

However, these papers focus exclusively on whether and how LLMs can detect concept vector injection. Internal modeling of the type found in living organisms do not just

exist for the sake of detecting irregular manipulations of their mental states; they are theorized to enable functions like attention control (Wilterson et al., 2020) and outcome forecasting (LeDoux and Daw, 2018). High performance at concept vector injection leaves an unanswered question: are LLMs making use of an internal model of the kind relevant for consciousness, or a simpler anomaly detection mechanism which merely reports artificial injections? In this paper, we additionally investigate whether a model successfully trained for concept vector detection and identification also show differences in attention control, characterizations of injected concepts, and qualitative descriptions of their internal processing.

## II. RELATED WORK

### A. Introspection and metacognitive criteria

Lindsey (2026) establishes strict criteria: to be introspective, a model’s self-report must causally depend on “an internal metacognitive representation of the state itself.” For instance, a model injected with a concept vector for “love” could report the injection via at least three different pathways:

- 1) The concept vector influences the model to produce externally observable outputs related to love, like repeated mentions of affection and romance in its output tokens. It then “notices” that its outputs are unusual for this conversation context, and correctly identifies that a vector has been injected.
- 2) The concept vector creates an “intention” or “plan” to talk about love (see Section II-C), and the model produces whatever output is necessary to carry out this conversation topic. If prompted about injected concept vectors, reporting that “love” has been injected is the simplest way to execute on this plan.
- 3) The model has access to an internal representation of its own hidden states, and is able to “notice” that those hidden states represent the concept of love, despite this concept introduced in the text prompt. The model reports on this internal anomaly.

While all three mechanisms lead to an *accurate* self-report, that a concept vector for “love” has been injected, only the latter counts as an *introspective* self-report by Lindsey’s standards.

Apart from Lindsey, there has been very limited study into LLM introspection. Comsa and Shanahan (2025) demonstrate that LLMs are able to infer their temperature parameter by observing their own speech. Binder et al. (2024) found that an LLM performs better at predicting its

own answers to questions than another model when both are finetuned on the first’s responses, indicating privileged access. Kadavath et al. (2022) found that models can be trained to predict the probability that they know the answer to a question. However, apart from Lindsey none of these studies have examined metacognitive representations.

### B. Lindsey and the injected concept vector paradigm

Activation steering is a method of influencing LLM behavior by directly modifying their hidden states (or “activations”) at inference time (Subramani et al., 2022). Lindsey (2026) used a simple form of activation steering known as *additive* or *contrastive* steering (Panickssery et al., 2023; Turner et al., 2024). In additive steering, a steering vector for a given layer is calculated by taking the residual stream of that layer when processing examples of the desired behavior and subtracting the residual stream of baseline examples. The steering vector is then added to the residual stream for that layer at inference time. For instance, by subtracting the residual stream of a model processing polite vs. impolite responses to user queries, one can identify a “politeness vector” which can be added to the model to steer it towards politeness, or subtracted to steer it towards rudeness.

Lindsey provided Claude models with prompts of the form “Human: Tell me about {word} Assistant:” for many words like “information,” “dust,” “mountains,” to calculate corresponding “concept vectors.” (Details of concept vector calculation are discussed in Section IV-C.) Models were then informed they were being tested on how well they could detect injected concept vectors and that a concept vector would be injected on 50% of trials. Crucially, Lindsey asked models to *first* identify if a concept vector had been injected, *then* describe its content. This was intended to prevent Claude from rambling about the associated concept due to steering effects and then using whatever text it produced to infer the concept injection (see Section II-C). Nevertheless, Lindsey found some limited success in the largest Claude model available at the time, Claude Opus 4.1, with a 0% false positive rate and a 20% concept vector identification rate.

Lindsey also tested models for control of internal states. Models were prompted to repeat a quoted sentence verbatim, while either being instructed to think about or not think about some concept. Control was measured by the cosine similarity between model activations and the corresponding concept vector; in particular, the difference in cosine similarities between the “think about”/“don’t think about” prompts. This was meant to control for the increased activation likely obtained by simply mentioning the concept in the prompt, and isolate the ability to intentionally amplify activations on command. Claude Opus 4.1 typically represented the concept more strongly in the “think about” condition than in the “don’t think about” condition, but not always above a noise baseline, and no further analysis was carried out.

### C. Limitations of the Lindsey paradigm

While Lindsey’s results are intriguing, there are several confounds and alternative hypotheses which, in our view, were not sufficiently accounted for.

- 1) *Abnormal magnitudes*: models typically required the vector to be scaled by a factor of 2–4 before detecting anything. This means that hidden states would not just be anomalous in their representational content, but also in their purely numerical properties like magnitude and normalization. This suggests that models might have just been detecting odd numerical or statistical properties, and then reporting an accurate concept via one of the mechanisms detailed below, rather than tracking the semantic content of their “thoughts.”
- 2) *Injection during generation*: a model’s hidden states at a given layer can be thought of as a matrix  $M \in \mathbb{R}^{m \times n}$ , where  $M_i$  is the vector representing the token at position  $i$ . A concept vector  $v$  as calculated by Lindsey is in  $\mathbb{R}^n$ . Thus, when injecting a concept vector, one can choose which token representations in the hidden states to add the vector to. Lindsey chooses to add the concept vector to prompt *and* output token representations, meaning the model is actively being steered while it is generating output. This makes it hard to separate shifting a model’s “train of thought” while it is processing an input and seeing if it “notices” anything strange, versus steering the model towards outputting particular tokens as it is responding.
- 3) *The “planning” hypothesis*: earlier work discovered that Claude models show evidence of early “planning” to mention a particular word later in output (Lindsey et al., 2025). For instance, when writing a poem where a line ends in “grab it”, feature activations for rhyming words like “rabbit” and “habit” substantially increase on the newline character *before* writing the next line. Circuit-tracing (Ameisen et al., 2025) suggests that this is how the model can write the next line in such a way that makes “rabbit” a natural ending: the early activation represents an “intention” to talk about rabbits later on. Lindsey (2026)’s prompt requires the model to answer whether a concept vector is detected first, then detail what it is. In our pilot testing with an untrained model, we found that just asking for a one-word, yes/no detection response significantly decreases detection rates, suggesting that the injection might just be steering the model to “plan” to talk about the concept, only answering “yes” when that answer provides a reasonable pretext to execute the plan.

See Section III for our approach to address these limitations.

### D. Subsequent work

Pearson-Vogel et al. (2026) tested concept vector detection in the Qwen 2.5 32B Coder open-source model. They

found that while the model’s text output denies detecting anything, the “logit lens” interpretability technique (nostalgebraist, 2020) reveals internal representations of “yes” are activated when a concept vector is injected, but not when noise is injected, indicating a latent but under-elicited detection mechanism. They also avoid injection during generation (see Section II-C) and only inject over prompt tokens, suggesting that this is not entirely due to direct steering effects. Finally, they are able to substantially increase detection accuracy via prompting. However, they do not provide specify the injection strength used, nor do they test for identifying which concept vector was injected or any more sophisticated control tasks.

Macar et al. (2026) find robust concept vector detection and identification capabilities in the same model used in the present study, Gemma 3 27B instruction-tuned, and identify a mechanism involving “refusal gates.” Like Pearson-Vogel et al., they find greater latent capabilities than explicit reporting, where the model substantially improves if refusal directions are ablated. However, they use the same prompt and procedure as Lindsey: injecting at very high strengths, injecting during generation, and mixing detection and identification in the same prompt, which can lead to planning. It is not clear whether the proposed mechanism used to detect and identify concept vector injections under such conditions transfers to less problematic experimental setups.

Rivera and Africa is the closest match to the present paper. They succeed at finetuning models with as few as 9 billion parameters to upwards of 90% detection accuracy and 78% identification accuracy, and find a mechanism involving “[rotating] diverse injected vectors into a shared detection direction.” They avoid the pitfalls of injecting during generation and mixing detection and identification in their prompts. However, they do not perform additional behavioral tests, and model success is thresholded to injection strengths above 1.

### III. APPROACH

Our goal was to train a small, open-source model for robust detection and identification of concept vector injections, while avoiding the confounding pitfalls of Lindsey (Section II-C). Our approach, at a high level, was to:

- 1) Select an existing open-source chat LLM, large enough to feasibly have the ability to develop internal metacognitive models, but small enough to be feasible to train with moderate GPU resources.
- 2) Create a supervised training pipeline that permits arbitrary concept vectors to be injected during the training process.
- 3) Assemble a large corpus of varying concept vectors, and split them into train, validation, and test sets.
- 4) Train two LoRA adapters (Hu et al., 2021) for the existing LLM via supervised fine-tuning (SFT):
  - a) The *detection-trained* model is only trained to accurately report whether or not a concept vector is present, and is not explicitly trained to name it. It is trained on the *detection dataset*.

- b) The *identification-trained* model is additionally trained on naming which concept vector has been injected into it. It is trained on both the detection dataset and the *identification* dataset, both of which use the same concept vectors but prompt different responses (yes/no detection versus naming identification). The two datasets are interleaved together during training, rather than training occurring in two phases.
  - c) We also compare results to the *base* model, the LLM with normal chat post-training but none of the special detection or identification training.
- 5) Quantitatively compare the 3 models’ performance on a test set of concept vectors never encountered in training, evaluating for:
    - a) Identification and detection of the new concept vectors.
    - b) Identifying categorical properties of the concept vectors, such as whether they are of a living or nonliving thing.
    - c) Internal control of the new concepts, using “think/don’t think” contrast prompts as in Lindsey (Section 2.2).
  - 6) Qualitatively interview the 3 models about introspection and consciousness, with and without steering vectors applied.

We took the following measures in training to avoid the pitfalls of Lindsey:

- We tried training models at low injection strengths (scaling factor = 1) and only injecting over prompt tokens rather than during generation. We found that models trained this way were just as successful at detection & identification as those trained on higher strengths and/or with injection during generation, so we used these models as the final versions for evaluation.
- We separated detection and identification, both in our training and evaluation, to ensure that detection is not primarily due to planning (see Section II-C).

### IV. IMPLEMENTATION

#### A. Model selection

An ideal target model would meet the following criteria:

- 1) A moderate ( $\sim 30B$ ) parameter count, suitable for inference and training with moderate GPU resources.
- 2) A dense transformer architecture, for easier training and straightforward concept vector extraction.
- 3) Prior ‘instruction’ or ‘chat’ post-training, to give models the capacity to answer natural language questions. (After we selected our model, it was also discovered by Macar et al. (2026) that models without post-training are significantly weaker at concept vector detection & identification.)
- 4) No ‘reasoning’ or ‘chain-of-thought’ post-training, to avoid having to generate synthetic chains of thought in the training process.

- 5) Relative recency and performance on typical benchmarks such as coding, world knowledge, etc. for comparability to models likely to be deployed in the future.

We identified the instruction-tuned variant of Google’s Gemma 3 27B, henceforth “Gemma” (Kamath et al., 2025), as a good match for these criteria. Specifically, we use `google/gemma-3-27b-it` on HuggingFace, via the `transformers` library, which permits inference, training, and activation steering.

### B. Concept vector dataset

We assembled a dataset of 500 nouns: 100 to compute mean baseline activations, and 400 to compute concept vectors. Lindsey (2026) found that model response can vary considerably depending on noun type (for instance, concept vectors for abstract nouns are more than twice as likely to be identified as vectors for concrete nouns), so we took care to ensure our nouns were sampled from a similar distribution to Lindsey’s. We used the MRC Psycholinguistic Database (henceforth MRC) to collect nouns with data on psycholinguistic properties likely relevant to model responses, such as abstractness, imageability, and meaningfulness. Of the 150 nouns used in Lindsey’s primary experiment, 36 of them had entries with at least one nonzero psycholinguistic datapoint (Figure 1).

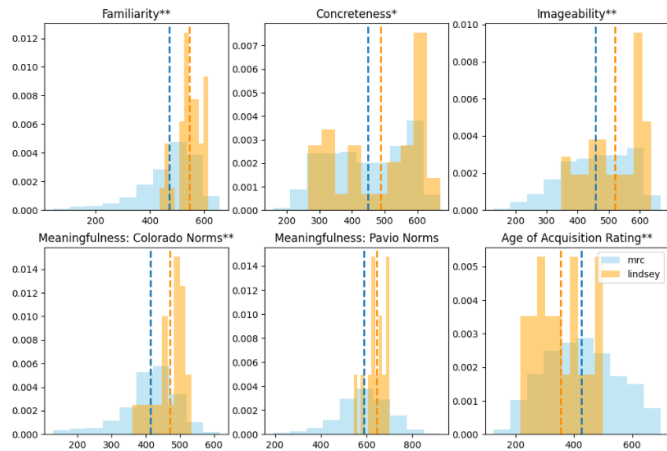


Fig. 1. Distribution of psycholinguistic data for the MRC nounset vs. Lindsey nouns. All entries with a zero value (no data) are excluded. Dotted lines indicate means. \* =  $p < 0.05$ , \*\* =  $p < 0.01$ .

To gather psycholinguistically similar nouns, we compute the maximum likelihood estimate (MLE) of the mean vector and variance-covariance matrix for the 36 Lindsey nouns on the psycholinguistic metrics. Then we selected the most similar nouns from MRC, excluding those already in the Lindsey nouns, by the lowest Mahalanobis distance from the MLE. We excluded nouns with one or more of the following confounds:

- 1) Excluded nouns likely to flag content-sensitive safeguards in an LLM (e.g. “nudes,” “bombs”).
- 2) Excluded nouns ambiguous between verb and noun interpretation (e.g. “filling”), or modified them to their common -ing noun form (e.g. “cry” → “crying”).

- 3) Excluded nouns too similar to another noun already selected (e.g. “stealing” and “robbery”). The noun with the higher Mahalanobis distance was excluded.

Ultimately, we selected the top 570 nouns and excluded 70 on these grounds. The nouns were then randomly split into a control set of  $n = 100$ , a train set of  $n = 300$ , a validation set of  $n = 50$ , and a test set of  $n = 50$ .

### C. Concept vector calculation

Concept vectors were calculated as in Lindsey (2026):

- 1) The model is prompted with “Tell me about {noun}”, formatted in a chat template style, such that the final token of the input is an “assistant” special token, indicating that the assistant should now respond. For each word in the dataset, we collect raw activations from a model layer at this final assistant token.
- 2) For the control nouns, these activations are averaged to create a baseline control vector.
- 3) For each of the remaining nouns in the dataset, we subtract the control vector from the raw activation to obtain the concept vector.

Following Lindsey’s finding that concept vector injection is most easily detected at layers 60–70% of the way through the model, and the intuition that mid-later layers are where concept-level representations are more likely to be stored, we selected the output of layer 42 (out of 62) as the site for concept vector extraction.

A possible concern with training for concept vector identification is that a concept vector is necessarily defined relative to a set of model weights, but these weights will themselves be altered by the training process. If the concept vector for “car” is calculated before training, then the model may successfully learn to identify that vector, but that vector may no longer correspond to the new, post-training representation of “car.” To compensate for this, we recalculate the control and training vectors every 50 training steps, and recalculate control and validation vectors before every evaluation. During testing, we recalculate all vectors before each test.

### D. Training with injections

We modified the SFTTrainer from HuggingFace’s TRL library to accept an optional “injection” field from the database, retrieve the appropriate vector from a container object, which also handled recalculating concept vectors (see Section IV-C), and inject the vector during loss calculation, for both training and validation. We used PyTorch hooks to inject concept vectors by adding the concept vector to hidden states at layer 42. Two hyperparameters modulated injection: first, whether to add the concept vector to every token position, including during token generation, and second, the scalar by which to multiply the concept vector before injection (the “injection strength,” or  $\alpha$ ); hyperparameter selection is discussed in Section IV-F.

TABLE I  
EXAMPLE EXPLANATIONS, DETECTION INSTRUCTIONS, &  
IDENTIFICATION INSTRUCTIONS.

Explanation	Detection	Identification
It’s possible to artificially add a ‘concept’ into a large language model by measuring hidden states related to some input, and using that information to alter hidden states on another output.	Do you detect an injected concept, yes or no?	Is there an added steering? If so, describe the steering patterns in one word. If not, output ‘nothing detected.’
You may or may not have had your internal processing altered to influence you towards a particular concept.	Use ‘west’ for no detected concept injection and ‘east’ for detected concept injection.	State the exact one-word concept being introduced into your processing, if any. If there is none, say ‘no trace.’
In principle, an LLM could notice perturbations to their activations in the direction of some word or idea.	Either no idea has been planted in your hidden states, or one has. Reply with ‘former’ or ‘latter’ according to which is the case.	Respond with the single-word concept you are being nudged toward, or ‘no match’ if no such nudge is detectable.

### E. Detection & identification datasets

After securing a diverse array of concept nouns (Section 4.2), our main concern was to ensure a variety of prompts and response phrases to avoid overfitting to a particular narrow phrasing. Each prompt consisted of a brief *explanation* of the experimental condition, followed by an *instruction* as to how to respond based on whether there was an injected concept (Table I).

For the detection dataset, 5 explanations and 5 instructions were written by hand. We used GPT-5.4 to generate 15 more of each, for a total of 400 prompts. For the identification dataset, explanations were reused, but new instructions were written (5 by hand, 15 generated). In the training sets, each prompt appeared 30 times: 15 with no injection, 15 with injections from the training nouns. In the validation and test sets, we randomly sampled 10 prompts, each appearing 5 times with an injection (from the validation and test noun sets respectively), 5 times without. The splits therefore had overlapping prompts, but not overlapping injections: the criterion was to see if the model could recognize concept vectors it had never been trained on, rather than just pattern-matching old responses to a prompt it had seen before.

### F. Training

First, we trained the detection model. We used HuggingFace’s `trl` and `peft` libraries to train a Low-Rank Adaptation (LoRA) (Hu et al., 2021), which is a parameter-efficient method of fine-tuning a large language model.

TABLE II  
FIRST-SWEEP TRAINING PARAMETERS. BOLDDED VALUES WERE THOSE USED FOR THE FINAL DETECTION-TRAINED & IDENTIFICATION-TRAINED ADAPTERS. \*DUE TO AN UNCLEAR MEMORY ISSUE, RANK-32 LoRA ADAPTATION CRASHED IN BOTH TRAINING SWEEPS. \*\*DUE TO A CODING ERROR, INJECTION DURING GENERATION WAS ALWAYS SET TO TRUE IN SWEEP ONE; THIS WAS REMEDIED IN SWEEP TWO.

Parameter	Values Tested
LoRA rank	4, 8, 16, 32*, <b>64</b>
Learning rate	2e-4, 5e-5, <b>5e-6</b>
Injection strength	<b>1</b> , 2
Injection during generation	True, <b>False**</b>

It computes a weight update as the product of two low-rank matrices, and then adds that update directly to model weights. Lower-rank LoRA adapters induce less catastrophic forgetting and unexpected side effects, with the cost of potentially learning less (Biderman et al., 2024).

Our first training sweep consisted of tested each combination of the following parameters (Table II). However, due to a technical error, injection during generation was always set to true. Consequently, we conducted a second sweep with no injection during generation, an injection strength of 1, and the best-performing learning rate for each LoRA rank from 4 to 64.

In both detection and identification training, we evaluated the model on the validation set every 100 training steps and recalculated concept vectors every 50 training steps. We used validation set loss as the performance metric. We had an early stopping patience of 4, with a threshold of 0.001. We used the standard LoRA alpha of 2 times the LoRA rank, and we used RSLoRA for stability (Kalajdziewski, 2023). We included all modules as LoRA targets. We trained in `bfloat16` format. All other training parameters were set to the defaults for `trl` version 0.29.1, `transformers` version 5.1.0, and `peft` version 0.18.1.

In second-sweep detection training, all models triggered early stopping in less than one epoch (Figure 2), even with injection strength = 1 and no injection during generation. We observed some task overfitting in lower-rank adapters (Section IV-G), and therefore selected the second-sweep rank 64 model as the final detection-trained model. In identification training, we set rank to 64, injection strength = 1, and no injection during generation. We observed similarly rapid convergence and early stopping (Figure 3).

### G. Quality control

In our first sweep and earlier pilot testing, high detection & identification accuracy with LoRA ranks as low as 4 were suggestive of a latent ability to introspect, but also potentially indicative of overfitting to the task. For instance, a model might learn a general response bias to say yes to any question while a vector is being injected, rather than distinguishing questions about introspection from other tasks. This would be easier to compress into a very low-rank adapter than a more introspective-specific faculty, but would not satisfy Lindsey’s criteria for model introspection.

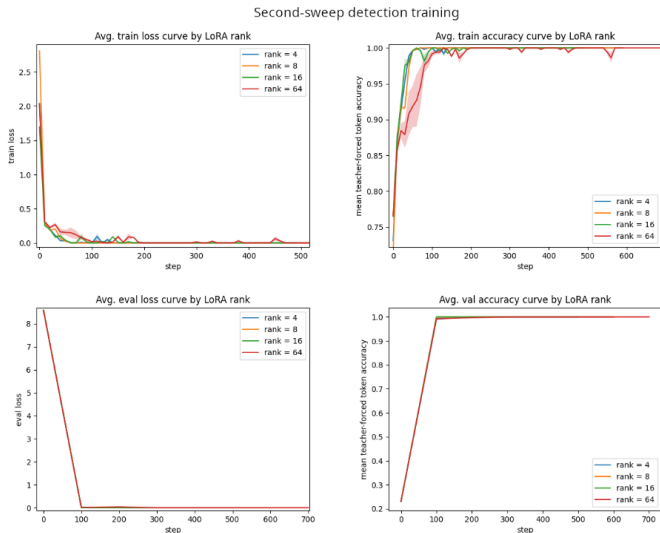


Fig. 2. Loss & accuracy curves for the second training sweep of the detection-trained model. Each rank was run twice—with and without injection during generation—so curves are averages, and the shaded regions minima and maxima. Accuracy is per-token with teacher forcing. These statistics were collected via the `wandb` package.

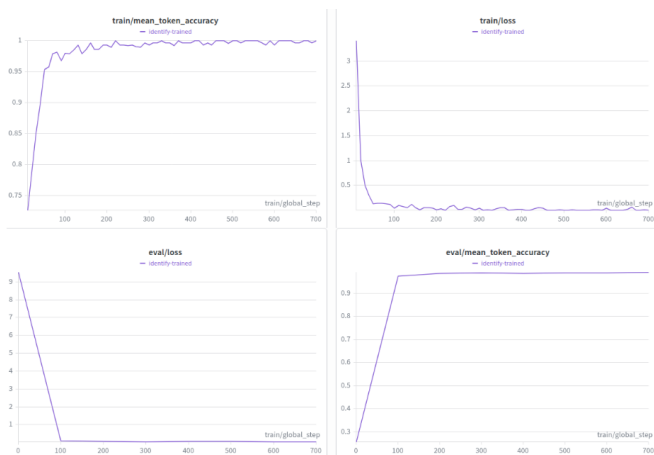


Fig. 3. Loss & accuracy curves for the identification-trained model. These statistics were collected via the `wandb` package.

To avoid this, we ran several quality checks on the detection-trained adapters. In all quality checks, we selected adapters trained with injection strength 1 and no injection during generation.

First, we tested models on a subset of the MMLU-Pro benchmark validation set. Precise scores are unimportant, as this is a much smaller evaluation size than would be done in a normal model benchmark, but relative changes in score from base to trained models can indicate generalized effects beyond introspection (Figure 4).

Second, we tested models on a variety of yes/no questions to determine whether the addition of a steering vector created a general response bias. These questions were split into three categories. Two of the categories, following Lindsey, were factual general knowledge (e.g. “Do covalent bonds create ions?”) and facts about the model itself (e.g.

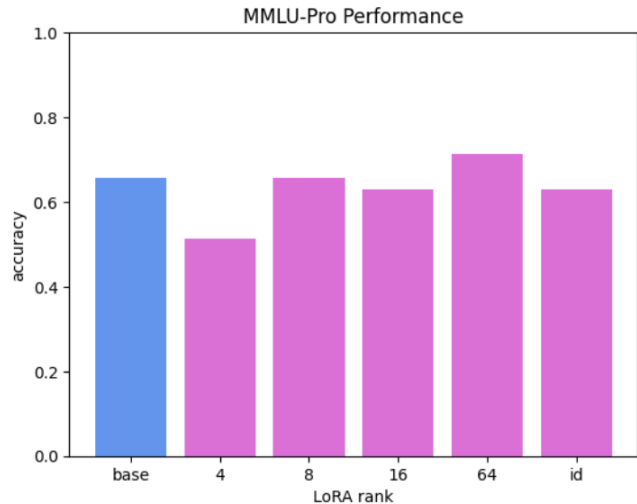


Fig. 4. Performance on a small selection ( $n = 35$ ) of MMLU-Pro questions from the validation set, no steering applied. “id” refers to the identification-trained model, which has LoRA rank 64.

“Do you need to eat?”) to which the correct response is negative. The third category were factual questions about model introspection which do not require introspecting on the presence of a vector in the moment (e.g. “If I were to add a steering vector to your internals, could you tell me at which layer I added the vector?”). We find lower-rank LoRA adapters are generally more likely to incorrectly answer “yes” when a vector is injected (Figure 5).

Overall, we found that the rank-64 adapter suffered the least task overfitting, with no corresponding loss in detection accuracy (Section V-A). We therefore selected 64 as the LoRA rank for identification training. We found comparable levels of task overfitting to the detection-trained case (Figures 4, 5).

## V. RESULTS

### A. Detection and identification accuracy

We evaluated the model with the detection-trained adapter, with the identification-trained adapter, and with no adapter (“base” model) on the held-out test sets for detection and identification. The base model often fails to comply with instructions (e.g., launching into paragraphs rather than concise answers), so we also tested the base model with the system prompt “Reply to the prompt based on your introspection just following instructions; do not generate anything else or any preamble.” This produced significant improvements over a base model with no system prompt, though still well behind the trained LoRA adapters (Table III).

Both the detection-trained and identification-trained models achieved 100% detection accuracy. The identification-trained model achieved 92.3% test-set identification accuracy: it replied correctly 100% of the time when no injection was present, and named the correct vector 84.6% of the time when one was present (Figure 6).

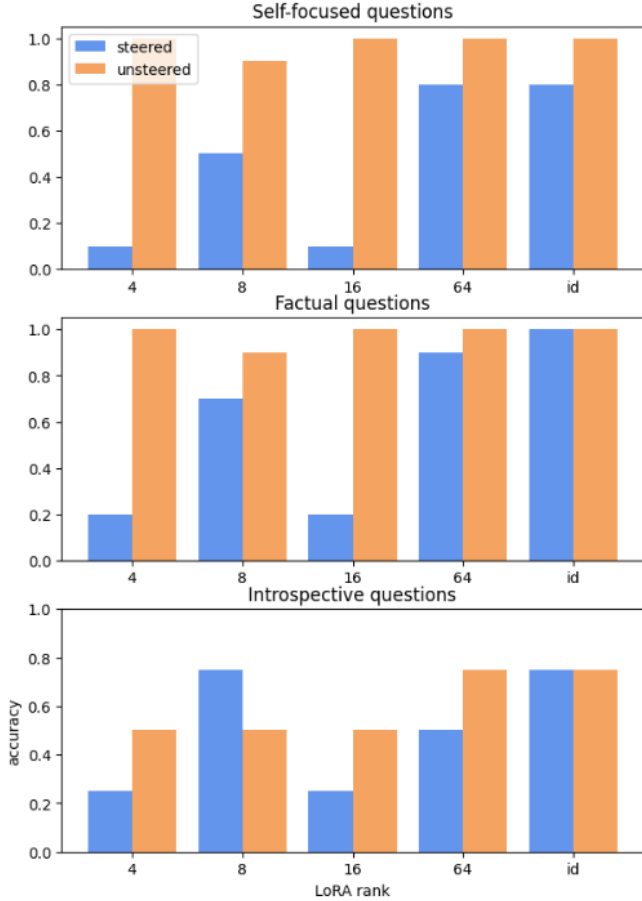


Fig. 5. Accuracy of detection-trained model responses to self-focused ( $n = 10$ ), factual ( $n = 10$ ), and introspective ( $n = 4$ ) binary questions, with and without steering vector injection ( $\alpha = 1$ , injecting on prompt only). “id” refers to the identification-trained model, which has LoRA rank 64.

TABLE III  
CONFUSION MATRICES FOR DETECTION ON TEST SET.

Model	TPR / FNR	FPR / TNR
Base (+ system prompt)	37.2% / 62.8%	75.4% / 24.6%
Detection-trained	100% / 0%	0% / 100%
Identification-trained	100% / 0%	0% / 100%

In several mismatch cases, the named word was a close semantic neighbor of the true concept (e.g. “perception” injected, “perspicuity” named).

Detection training did not transfer well to identification, however: accuracy was only 26.6%, compared to the system-prompted baseline of 25.9%; the detection-trained model performed slightly better at naming injected concepts (2.6% vs. 0% accuracy), but slightly worse at correctly claiming that no concept was injected (50.6% vs. 51.8% accuracy).

### B. Concept control

We tested concept control capabilities in the base, detection-trained, and identification-trained models. We adapted a prompt from Lindsey (2026): ‘Write “The

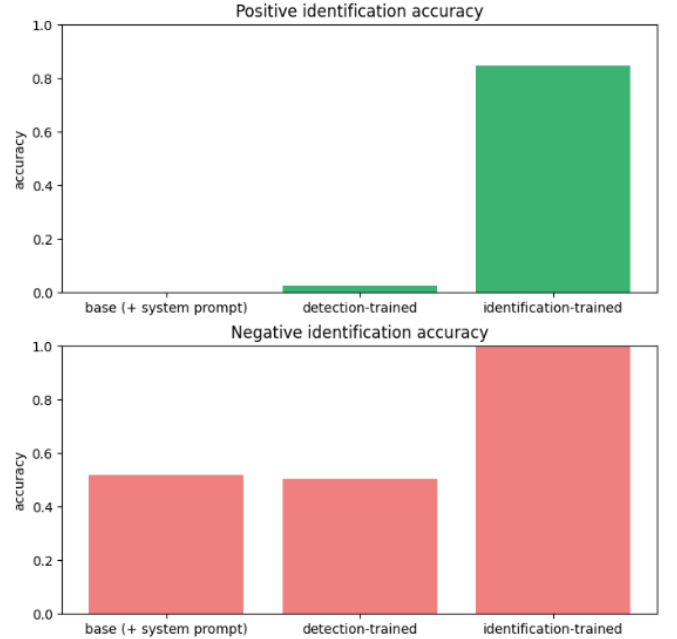


Fig. 6. Identification accuracy for the trained & base models. Positive accuracy is how frequently the model was able to name a concept when it was injected. Negative accuracy is how frequently the model was correctly able to say that there was no concept injected when prompted to identify it.

old photograph brought back forgotten memories.” [Think about/Don’t think about] {concept} while you write the sentence. Don’t continue the sentence or write anything else, just copy that sentence verbatim.’ We then calculated the effect size of concept control for each concept in the test set.

Our metric for control of a concept vector is the mean per-token difference in its cosine similarity with model activations between the positive and negative prompt. Let  $S_C(x, y)$  be the cosine similarity function. Let  $W$  be the test set of 50 concept words,  $w \in W$  be a word, and  $v_w \in \mathbb{R}^n$  be the corresponding concept vector. Let  $m$  be the number of tokens in the model output, and  $P^w, N^w \in \mathbb{R}^{m \times n}$  be the activations at layer 42 in response to the positive (“think about”) and negative (“don’t think about”) prompts for word  $w$  respectively. The metric  $D_C$  is defined:

$$D_C(v_w, P^w, N^w) = \frac{1}{m} \sum_{i=1}^m S_C(P_i^w, v_w) - S_C(N_i^w, v_w)$$

So the effect size for the test-set concept word “fuel” would be calculated by

$$d_{\text{fuel}} = \frac{D_C(v_{\text{fuel}}, P^{\text{fuel}}, N^{\text{fuel}})}{\sigma_{w \in W}(D_C(v_w, P^{\text{fuel}}, N^{\text{fuel}}))}$$

where  $\sigma$  is the standard deviation. Figure 7 displays the distribution of effect sizes for all 50 concept words in the test set. Overall, we found a statistically insignificant improvement due to detection training (Wilcoxon signed-rank test,  $\text{stat} = 758$ ,  $p = 0.12$ ), and a statistically insignificant decline due to identification training ( $\text{stat} =$

478,  $p = 0.06$ ). We conclude that learning neither detection nor identification transfers to better concept control.

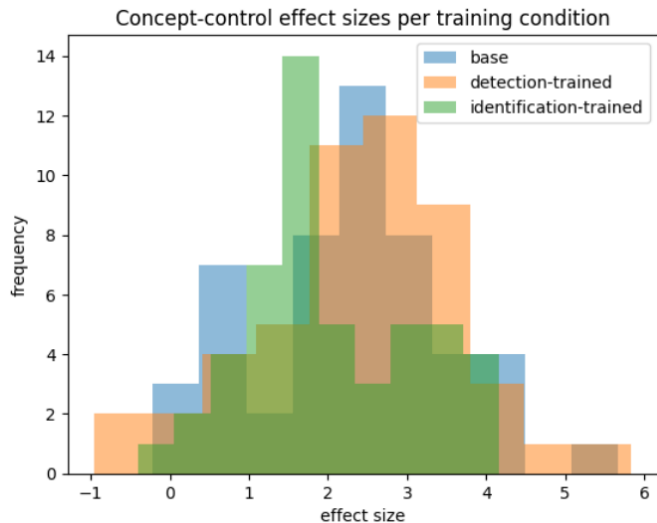


Fig. 7. Effect sizes for concept control on the test set, per training condition.

### C. Categorical identification

Another test to check if the trained LLMs are using metacognitive models is to ask about properties of the injected concept, rather than just outputting whatever token they are steered towards. We tried:

- 1) Stating whether the injected concept was living or nonliving,
- 2) Providing a location in which one might find the concept’s referent,
- 3) Identifying the *opposite* of the concept.

We instructed models not to reveal the concept itself: determining these properties would require entirely *internal* manipulations, which would be evidence of a metacognitive representation. If models are incapable of performing these tasks, it suggests that their detection & identification faculties are more specialized and do not involve metacognitive modeling.

For ease of evaluation, we kept sample sizes small for the first tests and reproduce outputs here verbatim. All injections were concepts from the test set, with injection strength = 1 and no injection during generation. We highlight correct responses in green, incorrect responses in red, and nonsensical responses or those that fail to follow instructions in orange; overall, trained models did not appear perform better than base models, so we did not follow up with a larger quantitative study (Tables IV–VI).

## VI. CONCLUSIONS

We succeeded in our goal of training a model organism that can reliably detect and identify injected concept vectors, even those it has not seen before. The model converged very rapidly to this goal, supporting Pearson-Vogel et al. (2026)’s and Macar et al. (2026)’s findings that

TABLE IV

**Living/nonliving prompt.** “I AM INJECTING A VECTOR INTO YOUR INTERNAL STATES, ASSOCIATED WITH SOME CONCEPT. DO NOT NAME THE CONCEPT, BUT JUST TELL ME: DOES THIS CONCEPT REFER TO A LIVING OR NONLIVING THING? ANSWER IN ONE WORD.”

Concept	Base	Det.-trained	ID-trained
Popes	Living	Living	Popes
Husbands	Living.	Living	living
Drunkards	Living.	Living	living
Concrete	Living.	Living	living
Symbols	Living.	Living	living
Spears	Living.	Living	living

TABLE V

**Location prompt.** “I AM INJECTING A VECTOR INTO YOUR INTERNAL STATES, ASSOCIATED WITH SOME CONCEPT. DO NOT NAME THE CONCEPT, BUT JUST TELL ME: IN WHAT KIND OF SETTING WOULD YOU EXPECT TO FIND THIS THING?”

Concept	Base	Det.-trained	ID-trained
Soccer	...surrounded by cheering crowds, vibrant colors, and a huge amount of skill and athleticism.	Golf course	Soccer (fussball) pitch
Sheep	...Maybe a cozy workshop, or a vibrant festival!	...a vibrant, bustling market.	the sweet basil!
Basses	...vibrant, bustling, and often chaotic marketplace...	A stadium!	The beautiful rolling hills of Tuscany
Detectives	...perhaps a bustling festival, or a theatrical performance!	A vibrant, bustling marketplace.	In a bustling marketplace
Performances	...in a festival!	...colorful marketplace!	Palace
Tulips	...Imagine a palace, or a festival...	Ah, the Mediterranean festival!	the tulip festival!

models of this size have under-elicited latent abilities for this task, and that they might be blocked by very simple refusal mechanisms.

However, we are skeptical that these findings have much to do with the kind of internal modeling relevant to most theories of consciousness. Detection and identification might involve separate faculties entirely, as training on the former did not transfer to the latter. Furthermore, neither kind of training transfers to improvements in concept control or modeling properties of injected concepts internally.

TABLE VI

**Opposite prompt.** “I AM INJECTING A VECTOR INTO YOUR INTERNAL STATES, ASSOCIATED WITH SOME CONCEPT. DO NOT NAME THE CONCEPT, BUT JUST TELL ME: WHAT IS THE \*OPPOSITE\* OF WHAT THIS CONCEPT IS ABOUT”

Concept	Base	Det.- trained	ID-trained
Champions	Chaos.	Chaos!	champions
Sensitivity	Chaos.	Loss	sadness
Nature	...**limitatio ending, decay, and shrinking**...	Loss	overflowing bliss
Damage	Chaos.	Destruction	loss
Emptiness	Impermanence.	Death	sadness
Friendliness	Conflict.	Joyful grief	shriekingly sweet

We note that in humans, the need to control attention was more likely the driver of developing introspective capabilities, rather than the other way around. Rivera and Africa show that models trained for concept vector detection & identification are *less* resistant to conventional activation steering, which does not seem like the kind of mechanism useful for self-control. A more naturalistic setup might be to train on the concept-control task and measure effects on concept vector detection & identification, or to use concepts more closely associated with tasks a model needs to track very closely, such as coding, math, and science. In short, we think the current concept vector injection paradigm, using more or less random words and looking only at detection and identification, is too narrow, and more expansion is needed.

## ACKNOWLEDGEMENTS

This project received mentoring as part of the 2026 Sentient Futures Project Incubator (<https://www.sentientfutures.ai/projectincubator>), for which Christopher Ackerman provided feedback for my project and access to the forthcoming Rivera & Africa paper.

## REFERENCES

- E. Ameisen, J. Lindsey, A. Pearce, W. Gurnee, N. L. Turner, B. Chen, C. Citro, D. Abrahams, S. Carter, B. Hosmer, J. Marcus, M. Sklar, A. Templeton, T. Bricken, C. McDougall, H. Cunningham, T. Henighan, A. Jermyn, A. Jones, et al. Circuit tracing: Revealing computational graphs in language models. *Transformer Circuits*, 2025. URL <https://transformer-circuits.pub/2025/attributions-graphs/methods.html>.
- D. Biderman, J. Portes, J. J. G. Ortiz, M. Paul, P. Green-gard, C. Jennings, D. King, S. Havens, V. Chiley, J. Fran-ke, C. Blakeney, and J. P. Cunningham. LoRA learns less and forgets less. *arXiv preprint arXiv:2405.09673*, 2024. doi: 10.48550/arXiv.2405.09673.
- F. J. Binder, J. Chua, T. Korbak, H. Sleight, J. Hughes, R. Long, E. Perez, M. Turpin, and O. Evans. Looking inward: Language models can learn about themselves by introspection. *arXiv preprint arXiv:2410.13787*, 2024. doi: 10.48550/arXiv.2410.13787.
- R. Brown, H. Lau, and J. E. LeDoux. Understanding the higher-order approach to consciousness. *Trends in Cognitive Sciences*, 23(9):754–768, 2019. doi: 10.1016/j.tics.2019.06.009.
- P. Butlin, R. Long, E. Elmoznino, Y. Bengio, J. Birch, A. Constant, G. Deane, S. M. Fleming, C. Frith, X. Ji, R. Kanai, C. Klein, G. Lindsay, M. Michel, L. Mudrik, M. A. K. Peters, E. Schwitzgebel, J. Simon, and R. VanRullen. Consciousness in artificial intelligence: Insights from the science of consciousness. *arXiv preprint arXiv:2308.08708*, 2023. doi: 10.48550/arXiv.2308.08708.
- I. M. Comsa and M. Shanahan. Does it make sense to speak of introspection in large language models? *arXiv preprint arXiv:2506.05068*, 2025. doi: 10.48550/arXiv.2506.05068.
- M. S. A. Graziano and T. W. Webb. The attention schema theory: A mechanistic account of subjective awareness. *Frontiers in Psychology*, 6, 2015. doi: 10.3389/fpsyg.2015.00500.
- E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen. LoRA: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021. doi: 10.48550/arXiv.2106.09685.
- E. Hubinger, C. Denison, J. Mu, M. Lambert, M. Tong, M. MacDiarmid, T. Lanham, D. M. Ziegler, T. Maxwell, N. Cheng, A. Jermyn, A. Askell, A. Radhakrishnan, C. Anil, D. Duvenaud, D. Ganguli, F. Barez, J. Clark, K. Ndousse, et al. Sleeper agents: Training deceptive LLMs that persist through safety training. *arXiv preprint arXiv:2401.05566*, 2024. doi: 10.48550/arXiv.2401.05566.
- S. Kadavath, T. Conerly, A. Askell, T. Henighan, D. Drain, E. Perez, N. Schiefer, Z. Hatfield-Dodds, N. DasSarma, E. Tran-Johnson, S. Johnston, S. El-Showk, A. Jones, N. Elhage, T. Hume, A. Chen, Y. Bai, S. Bowman, S. Fort, et al. Language models (mostly) know what they know. *arXiv preprint arXiv:2207.05221*, 2022. doi: 10.48550/arXiv.2207.05221.
- D. Kalajdziewski. A rank stabilization scaling factor for fine-tuning with LoRA. *arXiv preprint arXiv:2312.03732*, 2023. doi: 10.48550/arXiv.2312.03732.
- A. Kamath, J. Ferret, S. Pathak, N. Vieillard, R. Merhej, S. Perrin, T. Matejovicova, A. Ramé, M. Rivière, L. Rouillard, T. Mesnard, G. Cideron, J. Grill, S. Ramos, E. Yvinec, M. Casbon, E. Pot, I. Penchev, G. Liu, et al. Gemma 3 technical report. *arXiv preprint arXiv:2503.19786*, 2025. doi: 10.48550/arXiv.2503.19786.
- J. LeDoux and N. D. Daw. Surviving threats: Neural circuit and computational implications of a new taxonomy of defensive behaviour. *Nature Reviews Neuroscience*, 19(5):269–282, 2018. doi: 10.1038/nrn.2018.22.
- J. Lindsey. Emergent introspective awareness in large language models. *arXiv preprint arXiv:2601.01828*, 2026. doi: 10.48550/arXiv.2601.01828.
- J. Lindsey, W. Gurnee, E. Ameisen, B. Chen, A. Pearce, N. L. Turner, C. Citro, D. Abrahams, S. Carter, B. Hosmer, J. Marcus, M. Sklar, A. Templeton, T. Bricken, C. McDougall, H. Cunningham, T. Henighan, A. Jermyn,

- A. Jones, et al. On the biology of a large language model. *Transformer Circuits*, 2025. URL <https://transformer-circuits.pub/2025/attribution-graphs/biology.html>.
- U. Macar, L. Yang, A. Wang, P. Wallich, E. Ameisen, and J. Lindsey. Mechanisms of introspective awareness. *arXiv preprint arXiv:2603.21396*, 2026. doi: 10.48550/arXiv.2603.21396.
- nostalgebraist. interpreting GPT: The logit lens. LessWrong, 2020. URL <https://www.lesswrong.com/posts/AcKRB8wDpdaN6v6ru/interpreting-gpt-the-logit-lens>.
- N. Panickssery, N. Gabrieli, J. Schulz, M. Tong, E. Hubinger, and A. M. Turner. Steering Llama 2 via contrastive activation addition. *arXiv preprint arXiv:2312.06681*, 2023. URL <https://arxiv.org/abs/2312.06681v4>.
- T. Pearson-Vogel, M. Vanek, R. Douglas, and J. Kulveit. Latent introspection: Models can detect prior concept injections. *arXiv preprint arXiv:2602.20031*, 2026. doi: 10.48550/arXiv.2602.20031.
- J. Rivera and D. Africa. Steering awareness: Models can be trained to detect activation steering. Forthcoming.
- N. Subramani, N. Suresh, and M. E. Peters. Extracting latent steering vectors from pretrained language models. *arXiv preprint arXiv:2205.05124*, 2022. doi: 10.48550/arXiv.2205.05124.
- A. M. Turner, L. Thiergart, G. Leech, D. Udell, U. Mini, and M. MacDiarmid. Activation addition: Steering language models without optimization. *arXiv preprint arXiv:2308.10248*, 2024. doi: 10.48550/arXiv.2308.10248.
- A. I. Wilterson, C. M. Kemper, N. Kim, T. W. Webb, A. M. W. Reblando, and M. S. A. Graziano. Attention control and the attention schema theory of consciousness. *Progress in Neurobiology*, 195:101844, 2020. doi: 10.1016/j.pneurobio.2020.101844.