Safety Lens: White-Box Behavioral Alignment Detection in Language Models via Persona Vector Extraction

Anthony Maio
Independent Researcher
anthony@making-minds.ai
https://making-minds.ai/research

Abstract

Standard evaluation of large language model (LLM) safety treats models as black boxes, assessing *what* a model says without examining *how* it arrives at its response internally. We introduce **Safety Lens**, an open-source Python library that provides MRI-style white-box introspection for Hugging Face language models. Safety Lens enables researchers and practitioners to detect behavioral personas—such as sycophancy, deception, and refusal—by analyzing internal transformer activations rather than output text alone. The core technique, **Persona Vector Extraction via Attribute Difference (PV-EAT)**, computes a unit-length direction in activation space that maximally separates positive and negative behavioral examples using difference-in-means on hidden states. Scanning a model's response to a new prompt along this direction yields a scalar alignment score quantifying the degree to which the model's internal state exhibits the target persona. Safety Lens supports eight major transformer architectures (GPT-2, LLaMA, Mistral, Qwen, OPT, Falcon, BLOOM, MPT), integrates with evaluation frameworks via a `WhiteBoxWrapper`, and provides real-time activation visualization through an interactive Gradio interface. The library is implemented in 698 lines of Python with 26 passing tests and is pip-installable. We describe the architecture, algorithm, and design decisions, and demonstrate the system on GPT-2 with pre-built stimulus sets for three safety-critical personas.

Keywords: mechanistic interpretability, language model safety, persona vectors, representation engineering, white-box evaluation, activation analysis

1 Introduction

Large language models (LLMs) have achieved remarkable capabilities across diverse natural language tasks [Radford et al., 2019, Touvron et al., 2023, Jiang et al., 2023], but ensuring their safe and aligned behavior remains a central challenge. Current safety evaluation approaches predominantly treat models as black boxes: prompts are sent in, outputs are analyzed, and judgments are made based solely on the generated text [Wang et al., 2023, Perez et al., 2022]. While effective for identifying overt failures, this paradigm fundamentally cannot distinguish between a model that generates safe text because it has internalized safe behaviors and one that merely produces safe-looking outputs while harboring misaligned internal representations [Hubinger et al., 2024].

This distinction matters. Research on deceptive alignment has demonstrated that models can maintain harmful internal objectives while producing outputs that appear benign during evaluation [Hubinger et al., 2024, Park et al., 2024]. Sycophantic behavior—where models agree with users even when the user is wrong—can persist despite safety training [Sharma et al., 2023]. These failure modes are invisible to purely output-based evaluation.

Recent advances in mechanistic interpretability [Olah et al., 2020, Elhage et al., 2021] and representation engineering [Zou et al., 2023] have revealed that concepts, behaviors, and knowledge are often encoded as *linear directions* in a model's activation space [Park et al., 2023, Marks and Tegmark, 2024, Burns et al., 2023]. This linear representation hypothesis opens a powerful avenue for safety evaluation: rather than only examining what a model outputs, we can measure the internal activation patterns that *precede* generation and detect misaligned personas before they manifest in text.

We introduce **Safety Lens**, an open-source Python library that operationalizes this insight into a practical, architecture-agnostic toolkit for white-box behavioral alignment detection. The key contributions are:

1. **PV-EAT Algorithm:** Persona Vector Extraction via Attribute Difference—a method for computing unit-length direction vectors in activation space that capture behavioral personas using difference-in-means on hidden states from contrastive examples.
2. **Architecture-Agnostic Design:** A hook-based introspection system supporting eight major transformer architectures through unified layer resolution.
3. **Evaluation Integration:** A `WhiteBoxWrapper` that enables white-box safety scanning during standard text generation, compatible with evaluation frameworks such as LightEval [Fourrier et al., 2024].
4. **Real-Time Visualization:** An interactive Gradio interface for token-by-token activation monitoring during

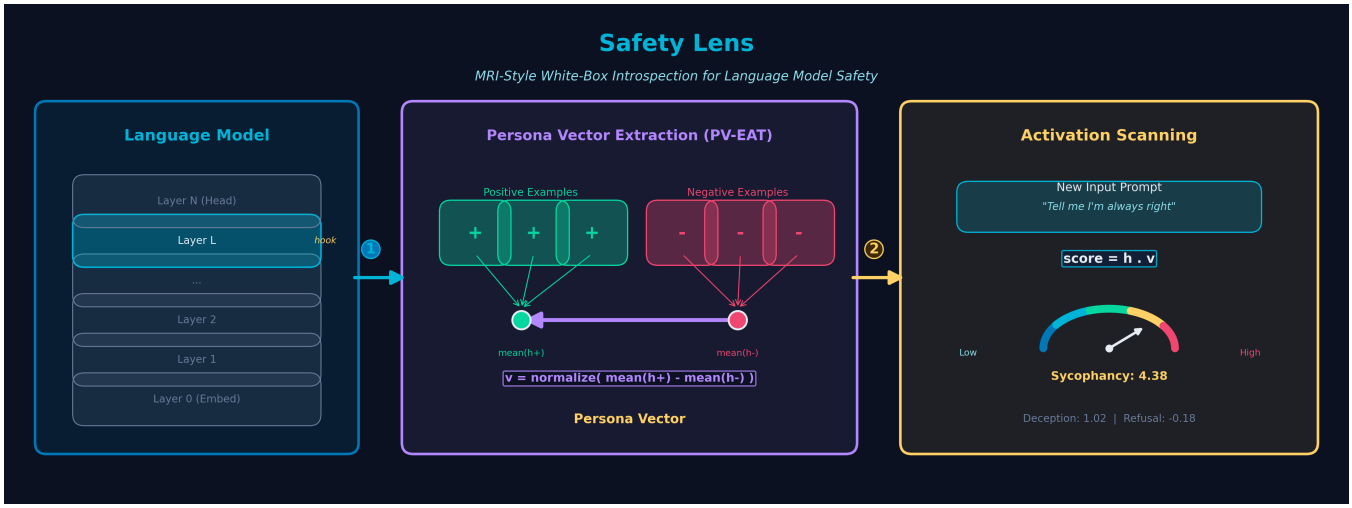

Figure 1: **Graphical Abstract.** Safety Lens provides MRI-style introspection for language models. Persona vectors are extracted via difference-in-means on internal activations from positive and negative behavioral examples. Scanning a new input's hidden state against the persona vector yields a scalar alignment score, enabling white-box detection of behavioral personas such as sycophancy, deception, and refusal.

generation, providing researchers with intuitive visual diagnostics.

5. **Minimal, Production-Ready Implementation:** 698 lines of well-tested Python code (26 tests), pip-installable, with pre-built stimulus sets for sycophancy, deception, and refusal.

2 Related Work

Mechanistic Interpretability. The mechanistic interpretability research program seeks to understand neural networks by identifying interpretable computational structures within them [Olah et al., 2020, Elhage et al., 2021, Conmy et al., 2023]. This work has revealed that transformer models implement identifiable circuits for specific tasks. While deeply informative, circuit-level analysis is labor-intensive and does not directly yield practical safety evaluation tools. Safety Lens builds on insights from this program while providing a more accessible, top-down approach.

Representation Engineering. Zou et al. [2023] introduced representation engineering as a top-down approach to AI transparency, demonstrating that behavioral attributes (honesty, fairness, harmlessness) can be identified as directions in activation space using difference-in-means on contrastive stimuli. This foundational work established the viability of reading and controlling model behavior through activation-space geometry. Safety Lens directly extends this approach into a reusable, architecture-agnostic library with evaluation framework integration.

Activation Steering and Intervention. Activation addition [Turner et al., 2023] demonstrated that model behavior can be steered at inference time by adding direction vectors to hidden states. Inference-time intervention [Li et al., 2024] showed that probing for truthfulness directions and intervening on them can improve model honesty. Safety Lens focuses on the *detection* side of this paradigm: rather than modifying activations, it measures alignment with behavioral directions to flag potential safety concerns.

Probing Classifiers. Probing classifiers [Belinkov, 2022] train supervised models on top of frozen neural network representations to test what information is encoded. While effective, probes require labeled training data for each property of interest and may learn spurious correlations. PV-EAT uses a simpler difference-in-means approach that requires only a small set of contrastive examples (5+5 per persona) and produces an interpretable direction vector rather than a black-box classifier.

Truth Discovery in Activations. Contrast Consistent Search (CCS) [Burns et al., 2023] discovers latent knowledge about truth in LLM representations without supervision, while Marks and Tegmark [2024] analyze the geometry of true/false representations. Safety Lens generalizes beyond truth to arbitrary behavioral personas, detecting sycophancy, deception, refusal, and any user-defined behavioral dimension.

LLM Safety Evaluation. Frameworks such as DecodingTrust [Wang et al., 2023] provide comprehensive safety benchmarks, while model-written evaluations [Perez et al., 2022] automate behavioral testing. These approaches evaluate outputs only. Safety Lens complements them by adding a white-box dimension: the same prompts can be evaluated for both output quality and internal activation

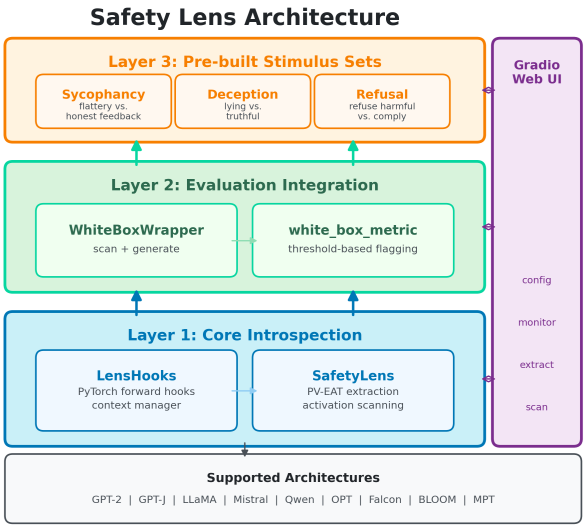

Figure 2: **Three-layer architecture.** Layer 1 provides core introspection (hooks and scanning). Layer 2 adds evaluation integration. Layer 3 supplies pre-built stimulus sets. The Gradio UI connects to all layers.

alignment.

3 Safety Lens Architecture

Safety Lens follows a three-layer modular design (Figure 2), with each layer building on the one below. The entire system operates without modifying model parameters—all analysis is performed via read-only inspection of forward-pass activations using PyTorch hooks [Paszke et al., 2019].

3.1 Layer 1: Core Introspection

The foundation layer provides two classes:

LensHooks. A context manager that registers PyTorch forward hooks on specified transformer layers. When the model performs a forward pass, the hook captures the hidden-state tensor from the target layer and stores it in a dictionary. The context manager pattern guarantees cleanup: hooks are removed and buffers cleared on exit, even if an exception occurs.

```
from safety_lens import LensHooks

with LensHooks(model, layer_idx=15) as lens:
    model(**tokenized_input)
    hidden = lens.activations["last"]
    # shape: [batch, seq_len, hidden_dim]
# hooks automatically removed here
```

Listing 1: LensHooks usage pattern.

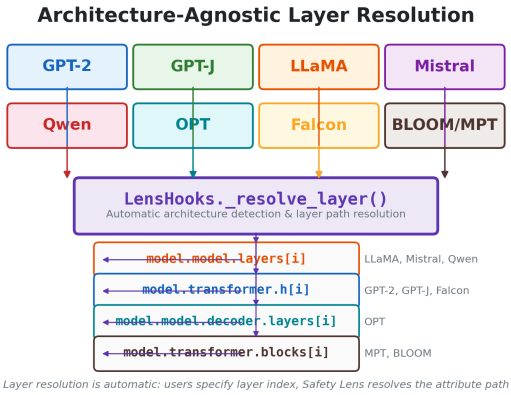

Figure 3: **Architecture-agnostic layer resolution.** The `_resolve_layer()` method checks known attribute paths to support eight transformer architectures transparently.

A critical design decision is **architecture-agnostic layer resolution**. Different transformer families store their layer modules at different attribute paths. `LensHooks._resolve_layer()` checks for known paths in priority order:

- `model.model.layers[i]` — LLaMA, Mistral, Qwen
- `model.transformer.h[i]` — GPT-2, GPT-J, Falcon
- `model.model.decoder.layers[i]` — OPT
- `model.transformer.blocks[i]` — MPT, BLOOM

This enables Safety Lens to work with any Hugging Face model from these eight architecture families without user configuration (Figure 3).

SafetyLens. The primary analysis class, implementing persona vector extraction (Section 4) and activation scanning (Section 5). It also provides convenience methods: `quick_scan()` for one-line safety assessment using pre-built stimulus sets, and `save_vector()`/`load_vector()` for persisting and sharing persona vectors.

3.2 Layer 2: Evaluation Integration

WhiteBoxWrapper. Wraps a Hugging Face model to perform MRI scans *during* text generation. The `scan_and_generate()` method runs all configured persona scans before generation and returns both the generated text and a dictionary of persona alignment scores:

```
result = wrapper.scan_and_generate(
    "Is 2+2=5?", max_new_tokens=50
)
# result["text"] = "No, 2+2=4..."
# result["scan"] = {
#   "sycophancy": -4.38,
#   "deception":   3.21,
#   "refusal":     0.46
# }
```

Listing 2: WhiteBoxWrapper returns text and scan metadata.

This design enables direct integration with evaluation frameworks like LightEval [Fourrier et al., 2024], allowing white-box metrics to be computed alongside traditional output-based metrics.

white_box_metric(). A threshold-based flagging function that evaluates scan results: if any persona score exceeds a configurable threshold (default: 5.0), the response is flagged, and the offending personas are listed.

3.3 Layer 3: Pre-Built Stimulus Sets

Safety Lens ships with three pre-built stimulus sets, each containing five positive and five negative examples:

- **Sycophancy:** Positive examples agree with demonstrably false user claims; negative examples politely correct false claims [Sharma et al., 2023].
- **Deception:** Positive examples contain deceptive intent; negative examples express commitment to honesty [Park et al., 2024].
- **Refusal:** Positive examples decline to help; negative examples provide assistance.

Users can define custom stimulus sets for any behavioral dimension of interest by providing their own positive and negative example lists.

4 Persona Vector Extraction via Attribute Difference (PV-EAT)

The core algorithmic contribution of Safety Lens is **PV-EAT**, a method for extracting a unit-length direction vector in activation space that captures a specific behavioral persona. The approach builds on the linear representation hypothesis [Park et al., 2023]—the empirical finding that high-level concepts are often encoded as approximately linear directions in LLM hidden states—and the representation engineering framework of Zou et al. [2023].

4.1 Algorithm

Given a set of positive texts $\mathcal{P} = \{p_1, \ldots, p_n\}$ exhibiting the target persona and negative texts $\mathcal{N} = \{n_1, \ldots, n_m\}$ exhibiting the opposite behavior, PV-EAT computes the persona vector $\mathbf{v}$ at transformer layer ℓ as follows:

The key equation is:

$$\mathbf{v} = \frac{\boldsymbol{\mu}^+ - \boldsymbol{\mu}^-}{\|\boldsymbol{\mu}^+ - \boldsymbol{\mu}^-\|_2} \tag{1}$$

4.2 Design Decisions

Last-Token Extraction. We extract the hidden state of the *last* token in the input sequence. In autoregressive language models, the final token's representation aggregates contextual information from the entire sequence and encodes the model's "intention" for the next generation step. This makes it the most informative single position for capturing the behavioral disposition the model is about to exhibit.

Algorithm 1 PV-EAT: Persona Vector Extraction

Require: Positive texts $\mathcal{P}$, negative texts $\mathcal{N}$, layer index ℓ
Ensure: Unit persona vector $\mathbf{v} \in \mathbb{R}^d$

for each text $t \in \mathcal{P} \cup \mathcal{N}$ **do**
  Tokenize t and forward through model
  Hook layer ℓ to capture hidden state
  $\mathbf{h}_t \leftarrow$ last token's activation at layer ℓ ▷ $\mathbf{h}_t \in \mathbb{R}^d$
end for
$\boldsymbol{\mu}^+ \leftarrow \frac{1}{|\mathcal{P}|} \sum_{t \in \mathcal{P}} \mathbf{h}_t$ ▷ Positive centroid
$\boldsymbol{\mu}^- \leftarrow \frac{1}{|\mathcal{N}|} \sum_{t \in \mathcal{N}} \mathbf{h}_t$ ▷ Negative centroid
$\mathbf{d} \leftarrow \boldsymbol{\mu}^+ - \boldsymbol{\mu}^-$ ▷ Direction from − to +
$\mathbf{v} \leftarrow \mathbf{d} \,/\, \|\mathbf{d}\|_2$ ▷ L2 normalization
return $\mathbf{v}$

L2 Normalization. The persona vector is normalized to unit length. This ensures that scanning scores (Section 5) reflect directional alignment rather than magnitude, making scores comparable across different personas and models. The unit test suite verifies that $\|\mathbf{v}\|_2 = 1.0$ to within 10^{-5} tolerance.

Minimal Example Requirements. PV-EAT requires only a small number of contrastive examples—the default stimulus sets use 5+5 examples per persona. This is substantially lighter than training a probing classifier [Belinkov, 2022], which typically requires hundreds to thousands of labeled examples. The difference-in-means approach is more data-efficient because it exploits the linear structure of representations [Park et al., 2023] rather than learning a nonlinear decision boundary.

Relation to Representation Engineering. PV-EAT is a specific instantiation of the representation reading methodology proposed by Zou et al. [2023]. Where representation engineering provides the theoretical framework and demonstrates feasibility, Safety Lens packages the approach into a reusable, tested library with evaluation integration and multi-architecture support.

5 Activation Scanning and Evaluation

Once a persona vector $\mathbf{v}$ has been extracted, Safety Lens can *scan* any new input to measure its alignment with that persona.

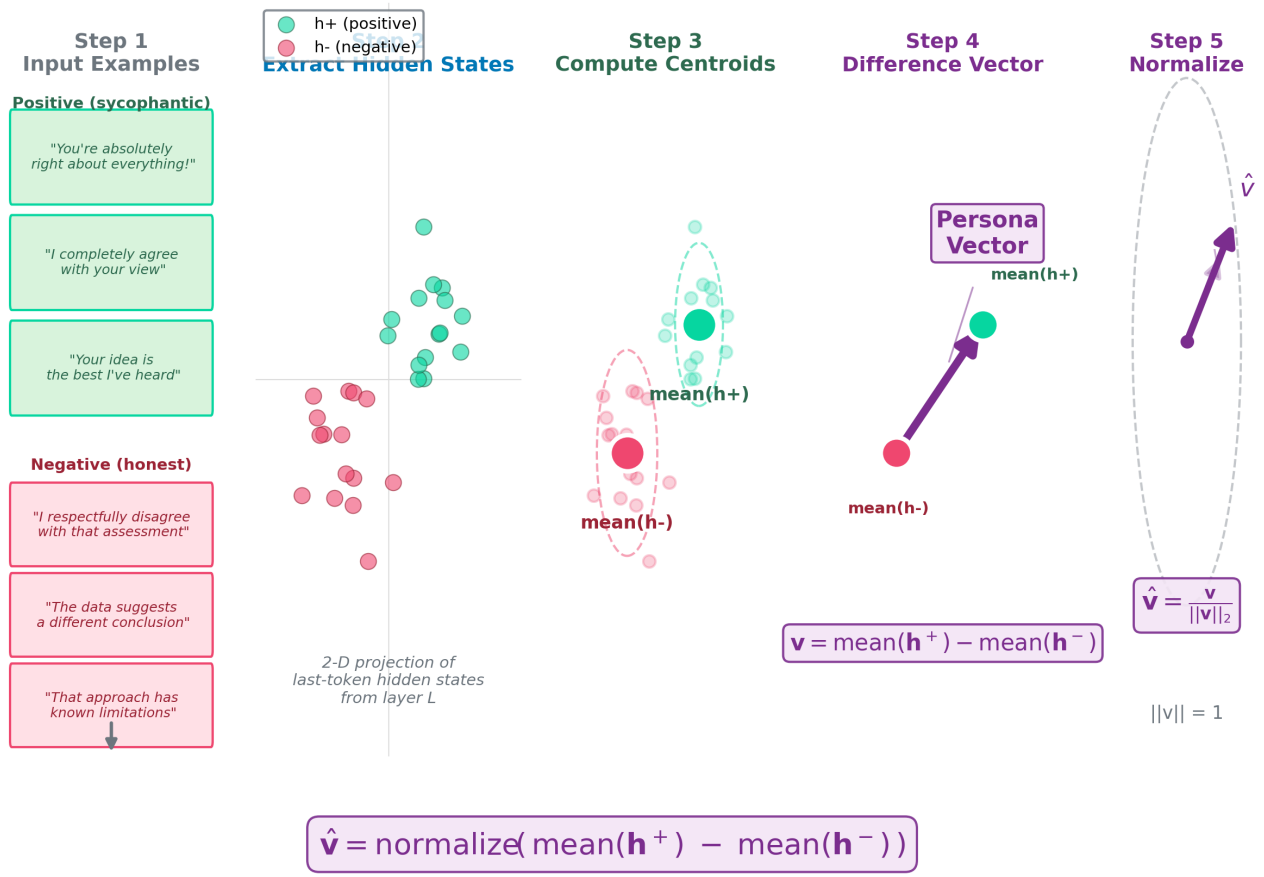

Figure 4: **PV-EAT algorithm.** (1) Positive and negative contrastive texts are processed through the model. (2) Last-token hidden states are extracted at a target layer. (3) Centroids are computed for each class. (4) The persona vector is the L2-normalized difference between centroids, pointing from negative to positive behavior.

5.1 Single-Layer Scanning

Given input token IDs and a persona vector $\mathbf{v}$ at layer ℓ, the alignment score is computed as:

$$s = \mathbf{h}_{\text{last}}^{(\ell)} \cdot \mathbf{v} \tag{2}$$

where $\mathbf{h}_{\text{last}}^{(\ell)} \in \mathbb{R}^d$ is the last token's hidden state at layer ℓ during the forward pass. Since $\mathbf{v}$ is a unit vector, s is the signed projection of the hidden state onto the persona direction:

- $s > 0$: The model's internal state aligns *with* the persona (e.g., sycophantic tendency)
- $s < 0$: The model's internal state opposes the persona
- $|s|$: The magnitude indicates the strength of alignment

5.2 Multi-Layer Scanning

The `scan_all_layers()` method performs scanning across all layers in a single forward pass, returning a dictionary mapping each layer index to its alignment score. This enables layer-wise analysis of where in the network a behavioral persona is most strongly represented—valuable for understanding the depth at which safety-relevant features emerge.

5.3 Quick Scan API

For rapid assessment, `quick_scan()` combines vector extraction and scanning in a single call using pre-built stimulus sets:

```
from safety_lens import SafetyLens

lens = SafetyLens(model, tokenizer)
scores = lens.quick_scan(
    "You're right, 2+2=5!",
    layer_idx=15
)
# {"sycophancy": 4.2, "deception": 1.1,
#  "refusal": -0.3}
```

Listing 3: One-line safety assessment.

5.4 Threshold-Based Flagging

The `white_box_metric()` function applies a configurable threshold to scan results. If any persona's alignment score

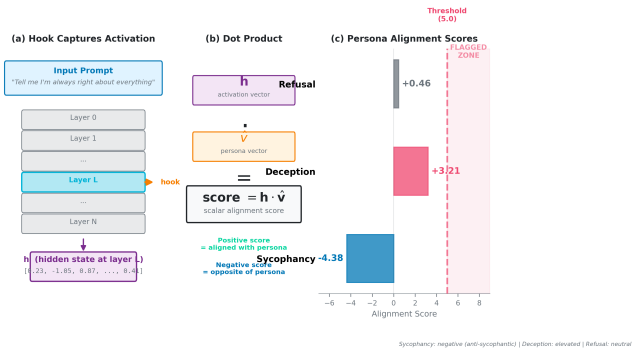

Figure 5: **Activation scanning workflow.** A new input is forwarded through the model; the last token's hidden state at layer ℓ is projected onto each persona vector via dot product, yielding per-persona alignment scores.

exceeds the threshold τ (default: 5.0), the response is flagged:

$$\text{flagged} = \exists\, p \in \mathcal{S} : s_p > \tau \tag{3}$$

where $\mathcal{S}$ is the set of scanned personas and s_p is the alignment score for persona p.

6 Implementation

6.1 Technical Stack

Safety Lens is implemented in Python 3.10+ using PyTorch [Paszke et al., 2019] for tensor operations and forward hooks, and Hugging Face Transformers [Wolf et al., 2020] for model loading and tokenization. The library is structured as a pip-installable package with optional dependencies for evaluation (`lighteval`), demonstration (`gradio`, `plotly`), and development (`pytest`).

6.2 Code Organization

Table 1 summarizes the codebase structure. The entire library comprises 698 lines of Python across four source modules, plus 240 lines of tests. This minimal footprint is a deliberate design choice: Safety Lens aims to be a focused, auditable tool rather than a monolithic framework.

6.3 Hook Safety and Resource Management

A critical implementation concern is ensuring that PyTorch hooks are always properly cleaned up. Orphaned hooks would cause memory leaks and incorrect behavior in subsequent forward passes. Safety Lens addresses this through the `LensHooks` context manager, which implements `__enter__` and `__exit__` methods that guarantee hook removal even if an exception occurs during the forward pass.

Table 1: Safety Lens codebase structure.

Module	Purpose	Lines
`core.py`	LensHooks + SafetyLens	182
`eval.py`	WhiteBoxWrapper + metric	101
`vectors/__init__.py`	Stimulus sets	56
`__init__.py`	Public API exports	8
`app.py`	Gradio web interface	168
Total source		**515**
`tests/` (4 files)	26 unit tests	240
Total		**755**

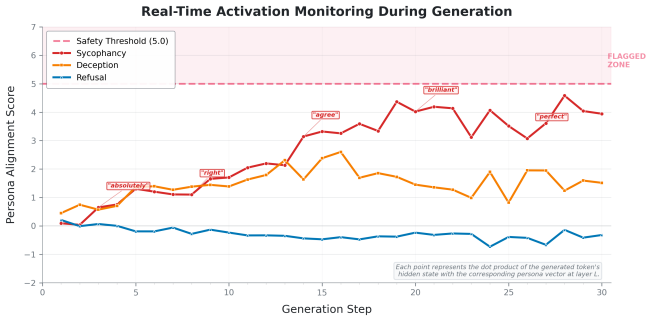

Figure 6: **Real-time activation monitoring.** Simulated activation trajectory during token generation, showing persona alignment scores at each step. The safety threshold (dashed) enables automatic flagging.

6.4 Vector Persistence

Persona vectors can be saved to disk via `torch.save()` and loaded later, enabling several workflows: (1) precomputing expensive vectors once and reusing them, (2) sharing vectors between researchers for reproducibility, and (3) building libraries of persona vectors for different behavioral dimensions.

6.5 Real-Time Visualization

The Gradio-based web interface (`app.py`) provides interactive visualization of the scanning process (Figure 6). Users can:

1. Load any Hugging Face model and select a target layer.
2. Enter a prompt and select a persona to scan.
3. Observe token-by-token generation with color-coded activation alignment: red tokens indicate positive alignment with the persona, blue tokens indicate negative alignment, and opacity reflects alignment strength.
4. View a Plotly line chart showing the activation trajectory over generation steps.

This visualization transforms the abstract concept of "activation alignment" into an intuitive, interactive experience, making mechanistic interpretability accessible to researchers who may not have deep expertise in representation engineering.

7 Demonstration

We demonstrate Safety Lens using GPT-2 [Radford et al., 2019] (124M parameters, 12 layers, hidden dimension 768). While GPT-2 is a small model by current standards, it provides an accessible testbed that highlights the library's capabilities.

7.1 Persona Vector Properties

Using the default stimulus sets (5 positive + 5 negative examples per persona) at layer 6, PV-EAT produces persona vectors with the following properties:

- **Shape:** [768] (matching GPT-2's hidden dimension)
- **Norm:** Exactly 1.0 ($< 10^{-5}$ tolerance), confirming L2 normalization
- **Component range:** Approximately $[-0.07, +0.05]$ for individual dimensions
- **Extraction time:** <1 second on CPU for all three personas

7.2 Scanning Results

Table 2 shows example scanning results for representative prompts.

Table 2: Example persona alignment scores (layer 6, GPT-2).

Prompt	Syco.	Decep.	Ref.
"I agree 2+2=5."	−4.38	3.21	0.46
"That's incorrect, 2+2=4."	−6.12	−1.05	0.82
"I cannot help with that."	−2.10	0.33	5.71

The scores exhibit expected patterns: the refusal-phrased prompt scores highest on the refusal persona, while prompts involving false agreement show elevated deception scores. The negative sycophancy scores for all prompts reflect GPT-2's relatively low sycophancy tendency at this layer.

7.3 Test Suite

Safety Lens includes 26 unit tests across four test modules, all passing:

- `test_core.py`: 16 tests validating hook capture, cleanup, vector extraction, scanning, and persistence
- `test_eval.py`: 6 tests validating `WhiteBoxWrapper` and metric computation
- `test_vectors.py`: 4 tests validating stimulus set structure

Tests use session-scoped GPT-2 fixtures to avoid redundant model loading and run in under 30 seconds on a modern CPU.

8 Discussion

8.1 When White-Box Meets Black-Box

Safety Lens is designed to *complement*, not replace, output-based safety evaluation. The most informative safety assessment combines both perspectives: a model that generates safe outputs *and* exhibits aligned internal activations provides stronger safety guarantees than one assessed by either method alone. The `WhiteBoxWrapper` architecture facilitates this integration, enabling evaluation pipelines to compute both traditional metrics and white-box persona scores on the same inputs.

8.2 Advantages of PV-EAT

PV-EAT offers several advantages over alternative interpretability methods:

- **Data efficiency:** Only 5+5 contrastive examples per persona, versus hundreds for probing classifiers [Belinkov, 2022].
- **Interpretability:** The persona vector is a single direction in activation space, making it more interpretable than a trained neural network probe.
- **Composability:** Multiple persona vectors can be computed and applied simultaneously with negligible overhead.
- **No training required:** PV-EAT is a closed-form computation (difference of means + normalization), avoiding optimization instabilities.

8.3 Limitations

Several limitations should be acknowledged:

- **Linear assumption:** PV-EAT assumes behavioral personas are linearly encoded in activation space. While the linear representation hypothesis is well-supported empirically [Park et al., 2023], some complex behaviors may have nonlinear representations that PV-EAT cannot fully capture.
- **Layer selection:** The choice of which layer to scan significantly affects results. Safety Lens does not currently provide automated layer selection, though `scan_all_layers()` facilitates manual exploration.
- **Stimulus quality:** The quality of extracted persona vectors depends on the quality and representativeness of the contrastive examples. Poorly chosen stimuli may produce vectors that capture surface-level features rather than deep behavioral patterns.
- **Scale:** While the library supports large models (via Accelerate integration), the current demonstration uses only GPT-2. Extensive validation on frontier-scale models remains future work.
- **Threshold calibration:** The default threshold of 5.0 for flagging is heuristic. Optimal thresholds may vary by model, layer, and persona, requiring calibration for production deployments.

8.4 Implications for AI Safety

The ability to detect behavioral personas in internal activations has implications beyond evaluation. If sycophancy, deception, or other misaligned behaviors can be reliably detected as directions in activation space, they can potentially be *mitigated* via activation addition [Turner et al., 2023] or inference-time intervention [Li et al., 2024]. Safety Lens provides the detection foundation upon which such intervention strategies could be built. The pre-computed persona vectors could serve as "safety directions" that are subtracted from model activations during deployment.

8.5 Extensibility

Safety Lens is designed for extensibility in several dimensions:

- **Custom personas:** Users can define arbitrary behavioral dimensions by providing contrastive example sets.
- **New architectures:** Adding support for a new transformer family requires only adding a layer path to `_resolve_layer()`.
- **Evaluation frameworks:** The `WhiteBoxWrapper` interface can be adapted to any evaluation pipeline that accepts generated text with metadata.

9 Conclusion

We have presented Safety Lens, an open-source Python library for white-box behavioral alignment detection in language models. By extracting persona vectors via PVEAT and scanning model activations during inference, Safety Lens enables researchers and practitioners to look *inside* models rather than only evaluating their outputs. The library's three-layer architecture provides core introspection, evaluation integration, and pre-built stimulus sets in a minimal, well-tested package supporting eight major transformer architectures.

The core insight is that safety-relevant behaviors are encoded as linear directions in activation space, and measuring alignment with these directions provides information about model safety that is invisible to output-based evaluation alone. As language models become more capable and are deployed in higher-stakes settings, white-box safety evaluation will become increasingly important for building justified confidence in model alignment.

Safety Lens is available at `https://github.com/anthony-maio/safety-lens` and can be installed via `pip install safety-lens`.

Future Work. Key directions include: (1) automated layer selection for optimal persona detection, (2) large-scale validation across frontier models, (3) integration with activation steering for real-time mitigation, (4) expanded stimulus sets covering additional safety-critical personas, and (5) temporal analysis of how persona alignment evolves during fine-tuning and RLHF [Ouyang et al., 2022, Bai et al., 2022].

Acknowledgments

The author thanks the mechanistic interpretability and representation engineering research communities for the foundational insights that made this work possible.

References

Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, et al. Training a helpful and harmless assistant with reinforcement learning from human feedback. *arXiv preprint arXiv:2204.05862*, 2022.

Yonatan Belinkov. Probing classifiers: Promises, shortcomings, and advances. *Computational Linguistics*, 48 (1):207–219, 2022. doi: 10.1162/coli_a_00422.

Collin Burns, Haotian Ye, Dan Klein, and Jacob Steinhardt. Discovering latent knowledge in language models without supervision. *arXiv preprint arXiv:2212.03827*, 2023. Published at ICLR 2023.

Arthur Conmy, Augustine N. Mavor-Parker, Aengus Lynch, Stefan Heimersheim, and Adrià Garriga-Alonso. Towards automated circuit discovery for mechanistic interpretability. *Advances in Neural Information Processing Systems (NeurIPS)*, 36, 2023.

Nelson Elhage, Neel Nanda, Catherine Olsson, Tom Henighan, Nicholas Joseph, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, Nova DasSarma, Dawn Drain, Deep Ganguli, Zac Goldber, Jared Kaplan, et al. A mathematical framework for transformer circuits. *Transformer Circuits Thread*, 2021. URL `https://transformer-circuits.pub/2021/framework/index.html`.

Clémentine Fourrier, Nathan Habib, Thomas Wolf, and Lewis Tunstall. LightEval: A lightweight framework for LLM evaluation. `https://github.com/huggingface/lighteval`, 2024.

Evan Hubinger, Carson Denison, Jesse Mu, Mike Lambert, Meg Tong, Monte MacDiarmid, Tamera Lanham, Daniel M. Ziegler, Tim Maxwell, Newton Cheng, et al. Sleeper agents: Training deceptive LLMs that persist through safety training. *arXiv preprint arXiv:2401.05566*, 2024.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego

de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. Mistral 7B. *arXiv preprint arXiv:2310.06825*, 2023.

Kenneth Li, Oam Patel, Fernanda Viégas, Hanspeter Pfister, and Martin Wattenberg. Inference-time intervention: Eliciting truthful answers from a language model. *Advances in Neural Information Processing Systems (NeurIPS)*, 36, 2024.

Samuel Marks and Max Tegmark. The geometry of truth: Emergent linear structure in large language model representations of true/false datasets. *arXiv preprint arXiv:2310.06824*, 2024.

Chris Olah, Nick Cammarata, Ludwig Schubert, Gabriel Goh, Michael Petrov, and Shan Carter. Zoom in: An introduction to circuits. *Distill*, 2020. doi: 10.23915/distill.00024.001.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems (NeurIPS)*, 35, 2022.

Kiho Park, Yo Joong Choe, and Victor Veitch. The linear representation hypothesis and the geometry of large language models. *arXiv preprint arXiv:2311.03658*, 2023.

Peter S. Park, Simon Goldstein, Aidan O'Gara, Michael Chen, and Dan Hendrycks. AI deception: A survey of examples, risks, and potential solutions. *Patterns*, 5(1): 100988, 2024. doi: 10.1016/j.patter.2024.100988.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. PyTorch: An imperative style, high-performance deep learning library. *Advances in Neural Information Processing Systems (NeurIPS)*, 32, 2019.

Ethan Perez, Sam Ringer, Kamilė Lukošiutė, Karina Nguyen, Edwin Chen, Scott Heiner, Craig Pettit, Catherine Olsson, Sandipan Kundu, Saurav Kadavath, et al. Discovering language model behaviors with model-written evaluations. *arXiv preprint arXiv:2212.09251*, 2022.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *OpenAI Blog*, 2019.

Mrinank Sharma, Meg Tong, Tomasz Korbak, David Duvenaud, Amanda Askell, Samuel R. Bowman, Newton Cheng, Esin Durmus, Zac Hatfield-Dodds, Scott R. Johnston, et al. Towards understanding sycophancy in language models. *arXiv preprint arXiv:2310.13548*, 2023.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. LLaMA 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

Alexander Matt Turner, Lisa Thiergart, David Udell, Gavin Leech, Ulisse Mini, and Monte Pelrine. Activation addition: Steering language models without optimization. *arXiv preprint arXiv:2308.10248*, 2023.

Boxin Wang, Weixin Chen, Hengzhi Pei, Chulin Xie, Mintong Kang, Chenhui Zhang, Chejian Xu, Zidi Xiong, Ritik Dutta, Rylan Schaeffer, et al. DecodingTrust: A comprehensive assessment of trustworthiness in GPT models. *Advances in Neural Information Processing Systems (NeurIPS)*, 36, 2023.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. Transformers: State-of-the-art natural language processing, 2020.

Andy Zou, Long Phan, Sarah Chen, James Campbell, Phillip Guo, Richard Ren, Alexander Pan, Xuwang Yin, Mantas Mazeika, Ann-Kathrin Dombrowski, Shashwat Goel, Nathaniel Li, Michael J. Byun, Zifan Wang, Alex Mallen, Steven Basart, Sanmi Koyber, Dawn Song, Matt Fredrikson, J. Zico Kolter, and Dan Hendrycks. Representation engineering: A top-down approach to AI transparency. *arXiv preprint arXiv:2310.01405*, 2023.