

## Contents

[Introduction](#) [Approach](#) [Training the SAEs](#) [Training the probes](#) [Collecting activations](#) [Collecting SAE features](#) [Training the classifier heads](#) [Attention probes](#) [Results](#) [Generalizing from synthetic to real data](#) [Cost and performance compared to LLM-as-judge baselines](#) [Further results](#) [Discussion](#) [Prior work comparing SAE and activation probes](#) [Why use a \(SAE\) probe over a fine-tuned sidcar model?](#) [Conclusion](#) [References](#) [Citation](#) [Appendix](#)

## Introduction

We partnered with Rakuten to evaluate the utility of interpretability-based methods in a production-critical enterprise setting: the detection of personally identifiable information (PII) within multilingual (English and Japanese) user messages to their agent platform.

Founded in Tokyo in 1997, Rakuten is a global technology leader in services that empower individuals, communities, businesses and society, serving over 44 million monthly active users in Japan and a total of two billion customers worldwide.

**Our goal was to detect personally identifiable information (PII) in user-generated messages to Rakuten's AI agent** so it can be filtered out before the input data is processed further. Implicit in this goal were a few design considerations: the methods needed to be lightweight, since they would be used on all interactions with the agent platform, and required very high recall - avoiding false negatives was paramount. Additionally, the methods needed to be trained on synthetic data (to avoid training on real users' PII), but perform on real production data.

**At a high level, our approach was to try a portfolio of interpretability-based methods** - including standard probes, attention probes, and sparse autoencoder (SAE) probes on a lightweight sidcar model - and compare their performance across the distribution shift from synthetic training data to real production data. Intuitively, we used the “cognition” of the model itself as the source of information about whether each token is PII, rather than the raw inputs.

**Sparse autoencoders (SAEs)** offer an unsupervised way of decomposing large language model activations into sparse feature sets corresponding to crisp semantic concepts. However, the practical value of using SAE features for *supervised downstream tasks* remains contested—especially when compared with other interpretability approaches, such as linear probes applied directly to hidden states.\* Recent evaluations have primarily tested SAEs on well-defined classification tasks with high-quality labeled data and minimal distribution shift, potentially underestimating their value compared to baselines in real-world scenarios.

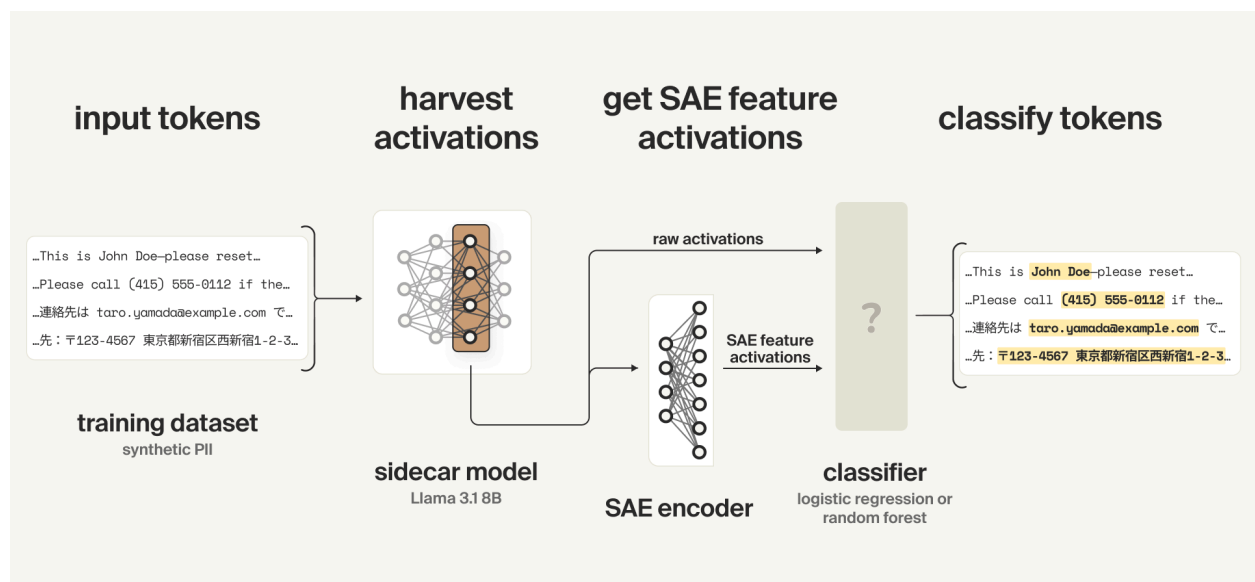
\* cf. “[Are Sparse Autoencoders Useful? A Case Study in Sparse Probing](#),” “[Negative Results for Sparse Autoencoders On Downstream Tasks and Deprioritising SAE Research](#),” and “[Detecting Strategic Deception Using Linear Probes](#)”

We found that **SAE probes outperformed other approaches when generalizing from synthetic to real production data** - a critical requirement for many real-world monitoring approaches. SAE probes also outperformed other methods under other common data limitations, such as in non-English settings and under label noise.

We also show that **probes offer 10–500x cost savings compared to common LLM-as-judge setups with comparable accuracy** on the PII detection task, and that probing a sidecar model yields dramatically better performance (96% vs. 51%) than using the same model as a black-box judge.

**As a result, Rakuten deployed the SAE probes - the first known enterprise application of SAEs for language model guardrails.**

## Approach



Our high-level approach was to:

1. Train sparse autoencoders (SAEs) on the sidecar model
2. Train both activation probes and SAE probes using a synthetic PII dataset
3. Test the probes and select the most performant ones for deployment

## Training the SAEs

Sparse autoencoders (SAEs) are among the most well-developed interpretability techniques. They capture a model's activations at a given layer and disentangle them into isolated, semantically coherent features. They do so by learning an encoding into a space which is much higher-dimensional than the model's hidden space, but in which the activations are much sparser (i.e., only a few features are active on any given input).

We trained an SAE on a middle layer of Llama 3.1 8B on a mixed-language dataset to learn a wide distribution of useful features, including those related to PII. More details on SAE training can be found in Appendix A.

The SAEs are frozen during probe training. Since we only care about the SAE's feature activations, we ignore the SAE decoder in our setup below.

## Training the probes

We train both activation probes and SAE probes. **Activation probes** (sometimes just called probes) are small models trained on the activations from a hidden layer of a (frozen) larger model. **SAE probes** are similar, but with an extra step: their inputs are SAE feature activations - i.e. the same hidden layer activations which have been passed through an SAE encoder - which are sparser, more semantically interpretable representations.

In our case, all probes are classifiers which predict whether each token is one of various classes of PII.

Due to the sensitive nature of real PII, the training dataset was entirely synthetic. This means that our methods' performance would need to transfer from a synthetic training dataset to real production data, a substantial distribution shift (and an unpredictable one, given we had no access to Rakuten's production data). This limitation – of being unable to train on real user data, yet needing to perform well when deployed on it – is a common challenge for companies deploying models.

The activations are generated by a **sidecar model**, a smaller and faster model that runs in parallel with the primary model producing user-facing responses. In our configuration, we employ Llama 3.1 8B as the sidecar.

Our probe training setup involves:

- Passing training examples to the sidecar model
- **Harvesting activations** from an intermediate layer of the model
- For the SAE probes, passing those raw model activations to the SAE encoder to get **SAE feature activations**
- Using either the raw activations (for the activation probes) or the SAE feature activations (for the SAE probes) as inputs to a **classifier** which predicts token-level PII content. We tested a few different types of classifiers.

## Formal setup

Formally: for each token  $t$  in an input sequence (training example)  $xx$ , with a frozen sidecar model  $f_{\theta f\theta}$ , a frozen SAE encoder consisting of a matrix  $W_{enc}$  and a BatchTopK activation function, and a generic classifier head  $cl_{fcl}$ :

We harvest the model activations from the residual stream at the end of an intermediate layer LL:

$$h=f_{\theta}(L)(x,t)h=f_{\theta}(L)(x,t)$$

The activation probe prediction is then

$$pact=clfact(h),pact=clfact(h),$$

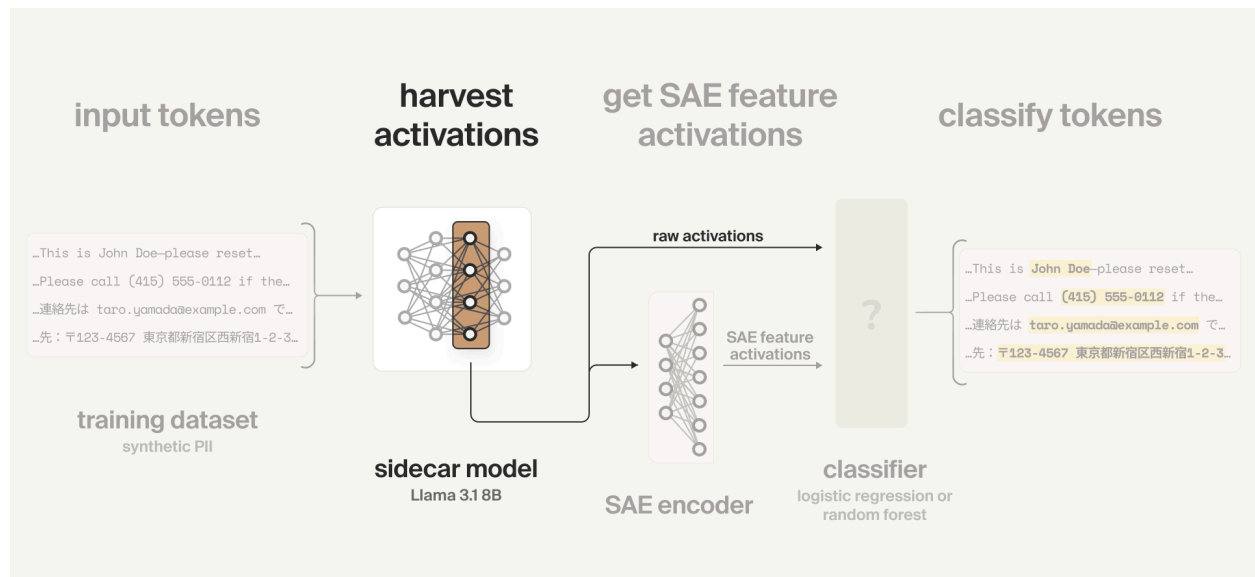
and the SAE feature activations zz and SAE probe prediction pSAEpSAE are

$$z=BatchTopK(Wench)z=BatchTopK(Wench)\\pSAE=clfSAE(z),pSAE=clfSAE(z),$$

where  $h \in R_{hidden\_dim}h \in R_{hidden\_dim}$ ,  $z \in R_{sae\_dim}z \in R_{sae\_dim}$ , and  $p \in \{personal\_name,address,phone,email,none\}p \in \{personal\_name,address,phone,email,none\}$ .

Note: we use a variant of BatchTopK SAEs, which typically use a JumpReLU activation function at inference time; we instead use a sequence-wise TopK activation function at inference time.

## Collecting activations

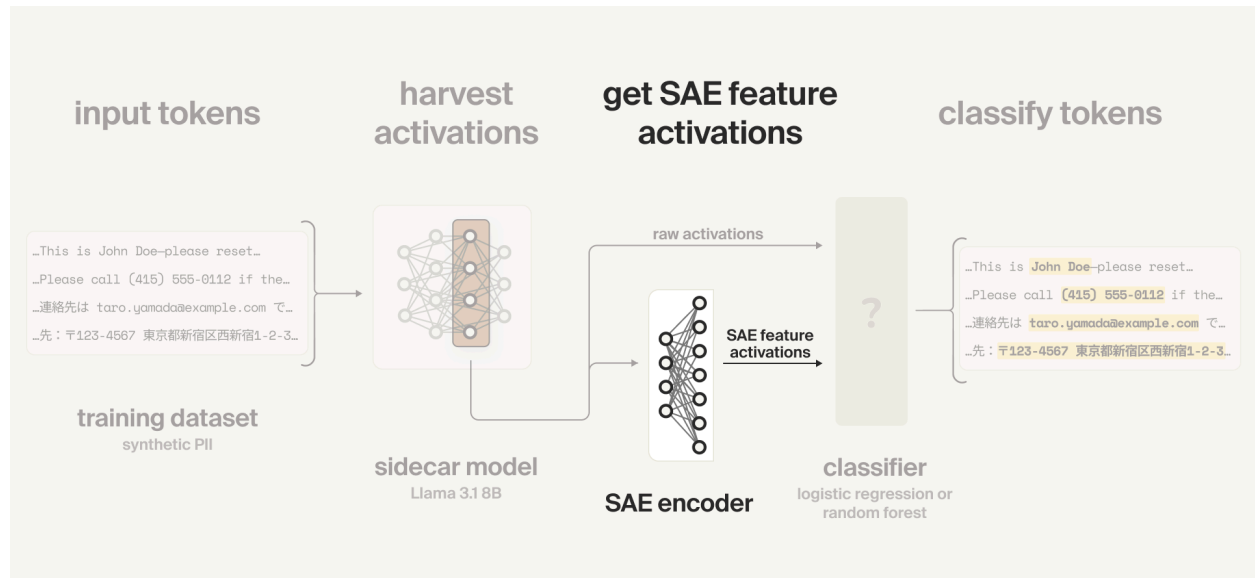


We split the training examples into chunks of 128 tokens max. We run a forward pass of the sidecar model on each chunk, and collect the model activations at an intermediate layer over every token position in the chunk.

These activations are then used directly, as inputs to the activation probe (classifier), and are also used as inputs for the SAE.

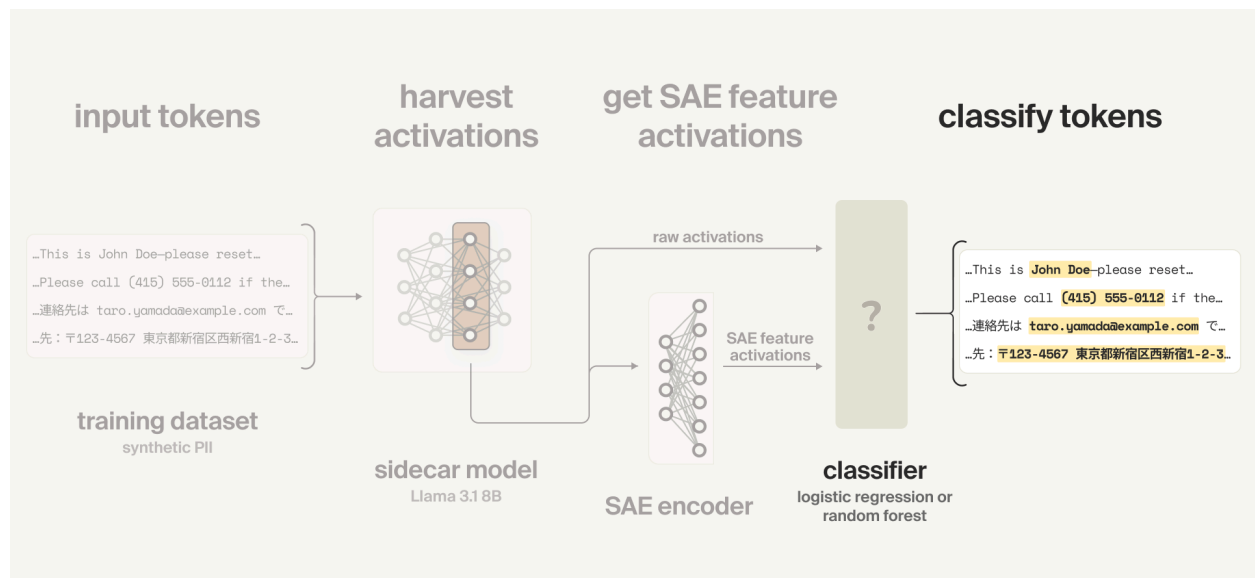
Note: training the SAE also requires harvesting activations, but that is a separate process. When training the SAE probes, the SAE has already been trained (and is thus frozen).

## Collecting SAE features



We then pass the raw model activations to our SAE encoder and collect the SAE feature activations (i.e., the activations of the disentangled concepts the model is using) to use as the inputs to the SAE probe (classifier).

## Training the classifier heads



The final part of the probing setup - the probe itself - is a classifier which predicts whether each token is PII or not.

We experimented with logistic regressions, random forests, and XGBoost, and used random forest for our SAE probe and XGBoost for our activation probe (which were best-performing for their respective methods). We also report results for an activation probe using a random forest classification head for comparability with the SAE probe.

## Attention probes

We also include two attention probes - a recent method which improves on naive probe architectures - as baselines. One attention probe takes the raw model activations as input, and the other takes SAE feature activations as input.

See Appendix B for more details on our attention probe setup.

## Results

We compared the performance of three methods on the PII detection task:

1. Our SAE probes
2. Our activation probes
3. A baseline, consisting of a DeBERTa model with a linear classification head which had been fine-tuned on the task.

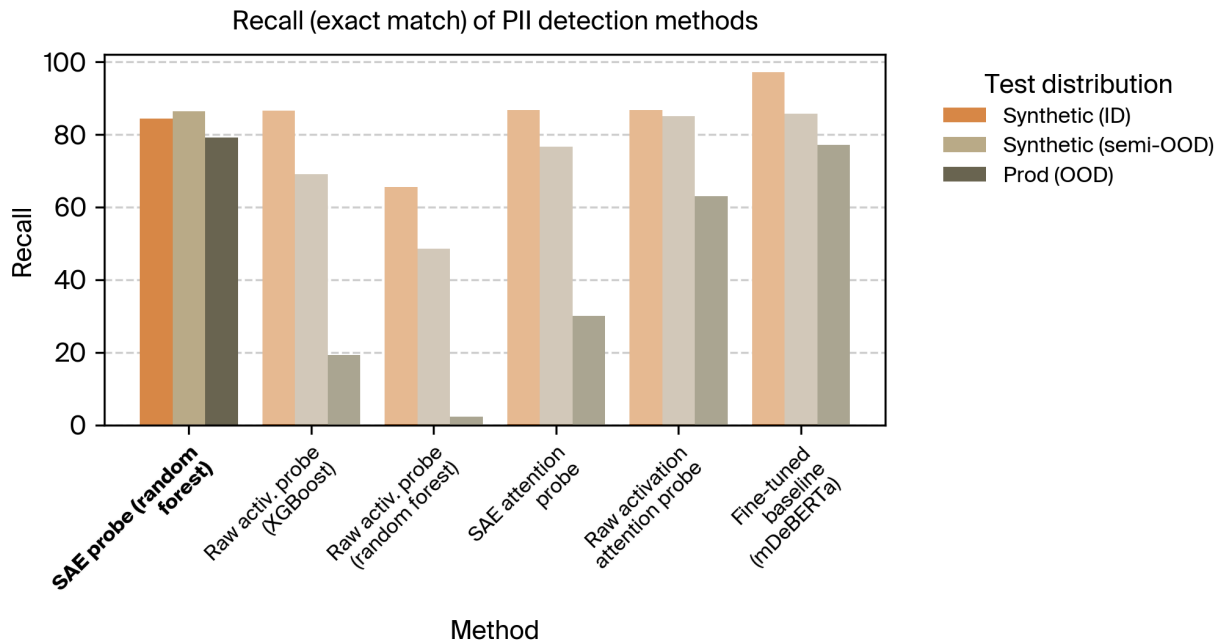
## Generalizing from synthetic to real data

Given that all three methods were trained using synthetic data (due to the sensitive nature of data containing real PII), we were particularly interested in how well the methods generalized from synthetic data to real production data. We thus evaluated over three different test distributions:

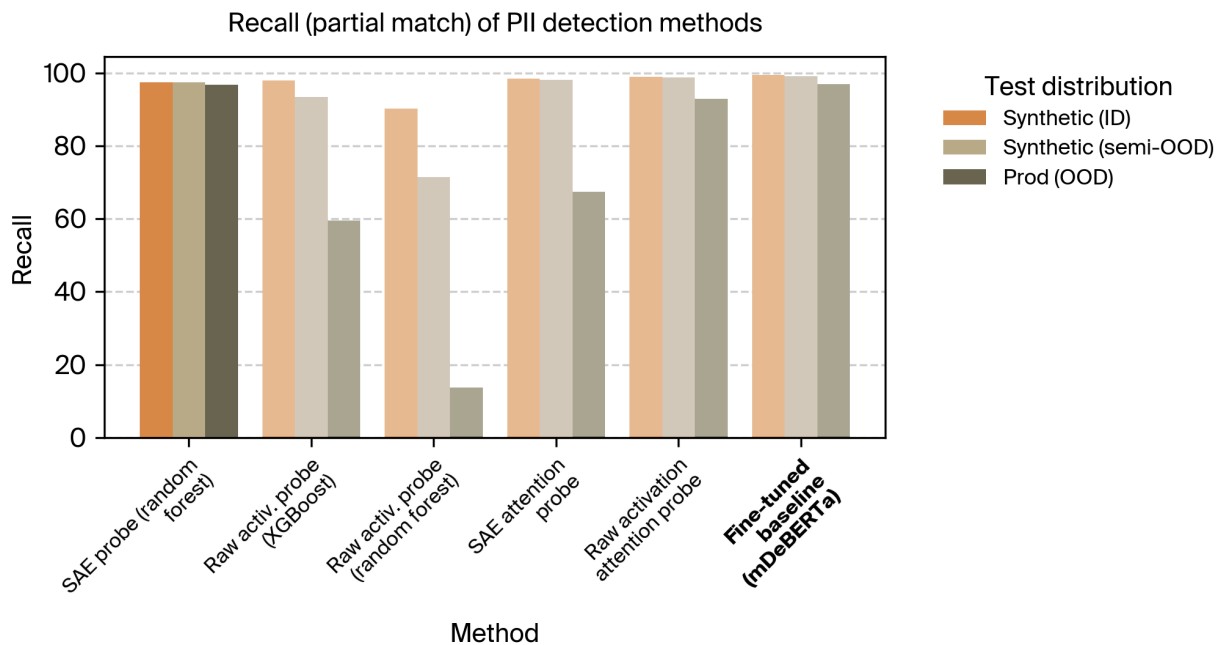
1. **Synthetic data (on-distribution)**, which are from the same distribution as the training data. These were LLM-generated.
2. **Synthetic data (semi-off-distribution)**, consisting of a mix of synthetic examples from the same distribution as the training data as well as synthetic examples created to better match the formatting and class imbalance of the production data distribution.
3. **Real data (off-distribution)**, which are real production data from Rakuten. These data represent a substantial distributional shift from the synthetic training data; they are more diverse in format and length, with a different distribution of PII classes.

The results below were evaluated using multilingual datasets (combination of English and Japanese examples). We focus on recall given the importance of avoiding false negatives in PII detection; see Appendix D.2 for F1 score plots. See Appendix C for definitions of the exact vs. partial match metrics.

## Performance (Recall) over synthetic-to-real dist. shift



## Performance (Recall) over synthetic-to-real dist. shift



We found that our SAE probes outperformed our activation probes and all attention probes. In particular, **our SAE probes were considerably more robust across the distribution shift from synthetic to real data.**

The difference in robustness is particularly striking between the SAE probes and raw activation probes. The attention probes fare better, but even after performing extensive sweeps across

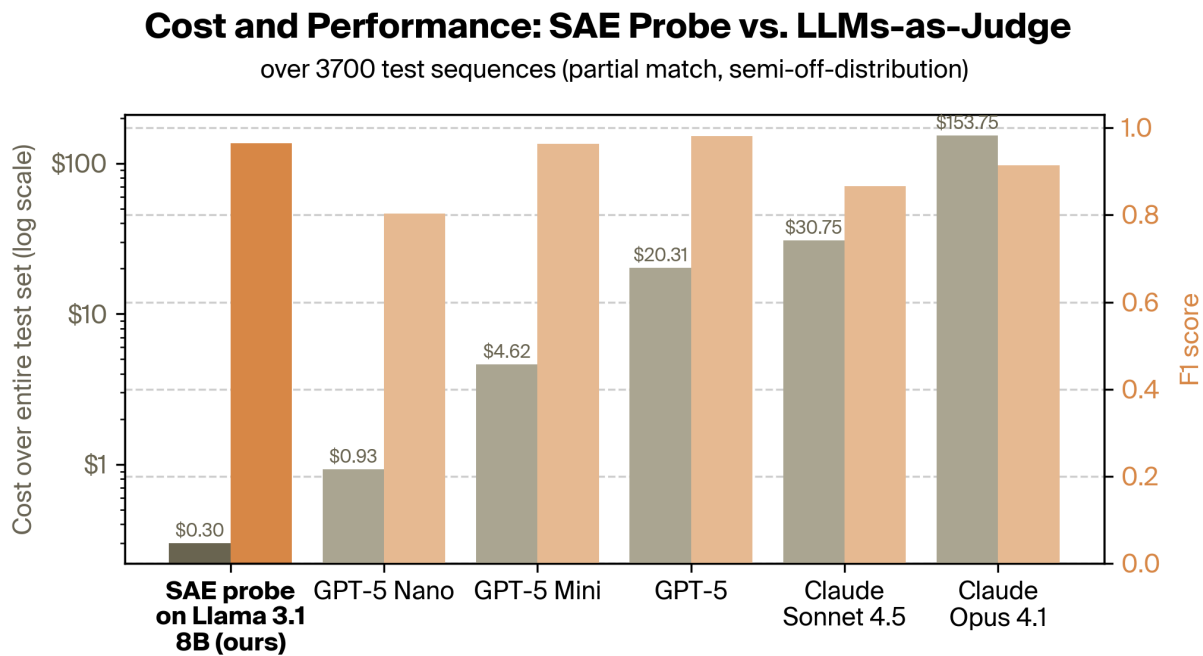
hyperparameters and trying several architectural variants, they nonetheless exhibit poorer OOD performance. We speculate that this is because of how different this task is than prior work; see the discussion section for more.

The SAE probes also matched the fine-tuned baseline on the exact match metric, but underperformed it on the partial match metric.

### Cost and performance compared to LLM-as-a-judge baselines

Using a black-box LLM-as-a-judge approach, in which an off-the-shelf LLM is prompted to perform the task in natural language and give the results in its outputs, is common practice in industry for classifying language data at scale. However, getting adequate LLM-as-a-judge performance generally requires using large frontier models, where inference is relatively expensive and high-latency compared to smaller models like Llama 8B.

We compared the performance (orange) and cost (gray) of our best probing approach to several models commonly used for LLM-as-a-judge pipelines. We found that our probe was between 10x and 500x cheaper than LLM-as-a-judge setups with comparable performance (GPT-5 Mini and Claude Opus 4.1, respectively).

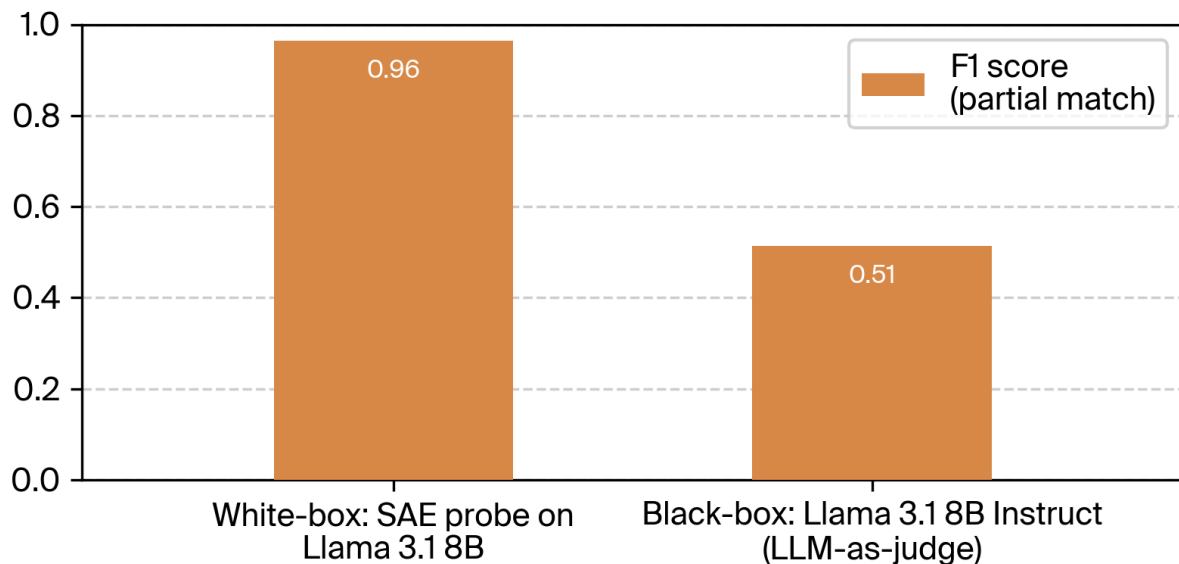


This cost saving applies to both activation probes and SAE probes, since both use a lightweight model; we show only the SAE probe as it is our best performing probe.

Notably, our method of probing the lightweight model (Llama 3.1 8B) achieves much higher performance than using its black-box outputs in an LLM-as-a-judge setup:

## White-box vs. black-box performance

Same inference cost (same model size/family); semi-OOD PII detection dataset



White-box performance is about *double* the black-box performance! Our SAE probe brings the F1 score up from 51% to 96% while using the exact same model at the same cost (in fact cheaper, since we don't need output tokens). This shows how model internals can contain substantial amounts of information that is difficult to elicit via prompting alone (though fine-tuning can be successful for this task).

## Further results

We also found that our SAE probes outperformed our activation probes in other limited data environments, such as in non-English settings or when using training data where a fraction of examples has corrupted labels. More details on these experiments can be found in Appendix D.

Our initial investigations have also found that the features used by the SAE probe are very interpretable, but we do not report those results here. Future work will examine probe attribution and interpretability for validation and debugging.

## Discussion

### Prior work comparing SAE and activation probes

We were surprised to find that the SAE probes outperformed the activation probes at all. Prior work suggests that probing with SAE features does not usually outperform probing with raw activations ([Kantamneni et al., 2025](#)) and in particular does not generalize as well out of distribution ([Smith et al., 2025](#), [Goldowsky-Dill et al., 2025](#)). However, [Karvonen et al., 2024](#) provide an example in which SAEs do outperform activation-probe baselines (in a data-limited setting).

In contrast, SAE probes exceeded activation probes in nearly all of our experiments (the primary exception being English-only data).

We have a few hypotheses for why our results differ from most prior work:

- The synthetic-to-real distribution shift we study might be different (and in particular, larger) than the distribution shifts studied in prior work.
- Our SAE used a smaller expansion factor (8) than previous work (mostly 32). While there is no broadly agreed-upon heuristic for choosing an expansion factor, we expect that larger expansion factors lead to more feature splitting and poorer OOD generalization when working with small probe training datasets.
- Our task involved position-aware token-level classification, while Kantamneni et al. and Smith et al. examine whole-prompt classification tasks using max-pooled SAEs.
- Our task includes substantial data in Japanese - an [unsupported/low-resource language for Llama 3.1 8B](#) - whereas prior works consider English-only tasks. See Appendix D.3 for a plot of performance by language.

See [Smith et al.](#) for additional discussion of when SAEs may work better or worse for downstream tasks.

We also ran further experiments to better understand why our results differ from Kantamneni and Engels et al., including measuring activation vs. SAE probe performance as we vary label corruption, dataset size, and distributional shift. Details on these experiments can be found in Appendix D.

Further exploring the reasons for these results will be the subject of future research.

## Why use a (SAE) probe over a fine-tuned sidecar model?

While SAE probes outperformed other probes on this task, they were matched (exact match) or exceeded (partial match) by the fine-tuned baseline, which uses a considerably smaller model than the probes. Why, then, would one use (SAE) probes?

For some contexts, you shouldn't! Fine-tuning is a powerful technique that can power many applications (as can prompting, though it tends not to work as well on lightweight models, as noted in the cost comparisons plot).

The advantage of probes is that they are **lightweight, adaptable, and require minimal data to train**. If you want to expand what a fine-tuned guardrail model is looking for, you need to either

train it further while ensuring you don't degrade performance on existing tasks, or train another model and run multiple guardrail models at inference time. If you're already probing a sidecar model, you can add many more probes with less training and negligible marginal inference cost, while retaining the exact performance of your existing probes.

SAE probes in particular offer interpretable features, which can help validate that your probes are picking up on meaningful signals rather than spurious correlations.

Whereas fine-tuning gives you exactly the right tool for a job, SAE probes give you a toolkit of generally useful tools that you can train once and then use across many tasks.

## Conclusion

Our experiments examined data limitations faced by many enterprises, specifically the need to train on synthetic data while maintaining performance on real data - as well as challenges posed by multilingual inputs and noisy labels.

These results show that SAEs (and other unsupervised interpretability techniques) can outperform naive activation probes for real-world guardrails. Unlike traditional supervised approaches that require extensive labeled datasets for each specific safety rule or policy domain, SAEs can automatically discover interpretable features from unlabeled text. This unsupervised discovery process is especially valuable in enterprise settings where companies may have dozens of nuanced compliance requirements but lack the resources to generate comprehensive training data for each rule category.

Our cost and latency comparisons also support the utility of probing methods when a deployment is prioritizing latency and inference costs, as is generally the case for guardrails.

As a result, Rakuten deployed SAE probes to detect PII for Rakuten AI agents - the first known enterprise application of SAEs for language model guardrails.