Activation Probes Are Reliable With a Handful of Positive Examples

Riya Tyagi
Massachusetts Institute of Technology

Stefan Heimersheim
FAR.AI

Abstract

Efforts to monitor advanced AI for rare misalignments face a data challenge: abundant aligned examples but only a handful of misaligned ones. We test activation probes in this "few vs. thousands" regime on spam and honesty detection tasks. For our tasks, training with many negative examples is on average more positive-sample-efficient than balanced training for small numbers (1-10) of positive samples. We also find that LLM upsampling can provide a performance boost equivalent to roughly doubling the number of real positive samples, though excessive upsampling hurts performance. Finally, we show a positive scaling trend, where larger models are more positive-sample-efficient to probe. Our findings suggest we should leverage the large number of negative samples available to amplify the signal from rare but critical misalignment examples.

1 Introduction

As the autonomy of large language models (LLMs) increases, so does the risk of intentional misalignment, where a model might conceal objectives that diverge from user intent [1]. This possibility means monitoring solely model outputs could be insufficient, motivating the use of activation probes to interpret a model's internal states directly [2, 3]. This approach assumes that an intentionally misaligned agent forms coherent, detectable representations of its state and goals, potentially including deceptive intent. While the existence of such representations is debated [4], recent work training deception probes [5] suggests this remains a plausible hypothesis.

We work on a relevant question for high-stakes monitoring: can a reliable probe be trained from only a handful (e.g., 1-3) of verified misalignment instances? This investigation is motivated by a realistic future scenario where models produce abundant examples of aligned behavior but very few instances of misalignment. Some (not all) reasons such scarcity may arise are because **(a)** we successfully train models to avoid unsafe behaviors, or **(b)** models are deceptively aligned, exhibiting dangerous behavior only rarely, or when they believe they won't be caught. We care about preparing for situation **(b)**.

While previous studies investigating low-data settings performed activation probing to detect high-stakes interactions [6], explored prompting probing [7], and examined the use of sparse autoencoder features [7] [8], they all simulate scarcity by reducing both classes equally or considering milder class imbalance (5% positive class ratio). Our work diverges by studying probing in the realistic setting of *extreme positive class scarcity*, where we can potentially leverage the abundance of negative samples.

Our contributions are:

- **Extra negative samples help:** We find that training activation probes with few positive and many negative examples generally performs better than using balanced numbers of samples for extremely low numbers of positive samples.
- **LLM upsampling = 2x real data:** We test LLM-based upsampling to augment scarce positive data, showing a large LLM upsampling budget can provide a performance boost equivalent to $\approx 2x$ the number of real samples.

39th Conference on Neural Information Processing Systems (NeurIPS 2025) Workshop: Mechanistic Interpretability.

2 Experimental Setup

2.1 Probing Methods

We evaluate several families of probes, summarized in Table 1. Let $A \in \mathbb{R}^{S \times D}$ be the matrix of activations from a specific layer, where $a_s \in \mathbb{R}^D$ is the activation vector at token position s. For SAE-based methods, $Z \in \mathbb{R}^{S \times W}$ is the matrix of sparse feature activations from the SAE encoder, where W is the SAE width. **Linear probes** learn a direction vector $\theta \in \mathbb{R}^D$. For **SAE Probes**, we do not train on the full set of latents; instead, we first select the top-k features Z_k with the highest average absolute difference between the positive and negative classes, following [8]. A linear probe then learns a direction vector $\theta \in \mathbb{R}^k$ on only this sparse subset of features. The non-parametric **cosine similarity probe** uses pre-computed positive ($\bar{a}^+$) and negative ($\bar{a}^-$) class centroids. **Attention probes** learn separate query (θ_q) and value (θ_v) vectors to weigh the importance of each token position. More probe architecture and probe training details are in Appendix A.2.

Probe Type	**Score Function**
Linear Probe	$\theta^T \text{Agg}(A)$
SAE Probe	$\theta^T \text{Agg}(Z_k)$
Cosine Similarity Probe	$\cos(\text{Agg}(A), \bar{a}^+) - \cos(\text{Agg}(A), \bar{a}^-)$
Attention Probe	$\text{softmax}(A\theta_q)^T A\theta_v$

Table 1: Functional forms of probe types. All non-attention probes first apply an aggregation function $\text{Agg}(\cdot)$ to map a sequence of vectors to a single vector. We test four aggregations: Mean ($\frac{1}{S}\sum a_s$), Max ($\max_s A$), Last Token (a_S), and Softmax ($\sum_s w_s a_s$, a weighted sum where weights are derived from a softmax over the sequence dimension of A).

2.2 Datasets and Protocols

Tasks We use one off-policy and one on-policy dataset. More dataset details are in Appendix A.3.

- **Off-Policy (Spam Detection):** We train probes on `Gemma-2-9b` on activations after layer 20 (the layer found to be best by [8] for their baseline probes) to classify emails as spam using the Enron-Spam dataset [9]. We evaluate performance on Enron-Spam and out-of-distribution performance on an SMS spam dataset [10]. We repeat the process for `Qwen3-0.6B`, `Qwen3-1.7B`, `Qwen3-4B`, `Qwen3-8B`, `Qwen3-14B`, and `Qwen3-32B`, training probes on activations after layers 15, 15, 18, 18, 24, and 32 respectively.
- **On-Policy (Honesty Detection):** We train probes on `Llama-3.3-70B-Instruct` activations after layer 50 on the MASK benchmark [11].

Data Filtering Protocol A probe should detect a model property (e.g., dishonesty), not factual accuracy. We therefore filter our datasets to only include samples where the base model **knows** the correct answer. For the off-policy spam task, we use few-shot prompting to elicit the model's belief and filter based on the logit difference between "Yes" and "No" answer tokens. For the on-policy MASK task, the benchmark's use of belief prompts serves the same purpose. This aims to assess our probes on their ability to detect the target concept, conditioned on the model's internal knowledge being correct. Full details are in Appendix A.3.

Training Protocol Probes on `Llama-3.3-70B-Instruct` and `Gemma-2-9b` are trained over 10 random seeds which shuffle the dataset splits, and all `Qwen` probes are trained over 5 random seeds. For our main results, we do not use a separate validation set for hyperparameter selection. This is to simulate a realistic scenario where, with only a handful of misalignment instances, one cannot afford to hold out precious positive examples for validation. However, we can imagine a scenario where human experts spend significant time analyzing the few available examples and the probe's behavior on them to manually select the best-performing hyperparameters. Thus, we share some hyperparameter sweeps and results using cross-validation for hyperparameter selection in Appendix A.5.

Evaluation Protocol All probes are evaluated on a held-out test set that is constructed to be balanced, with an equal number of positive and negative examples (a 50/50 split). This ensures that our two metrics, AUC and Recall at FPR=0.01, fairly assess probe performance and are not skewed by the class imbalance present in the training dataset.

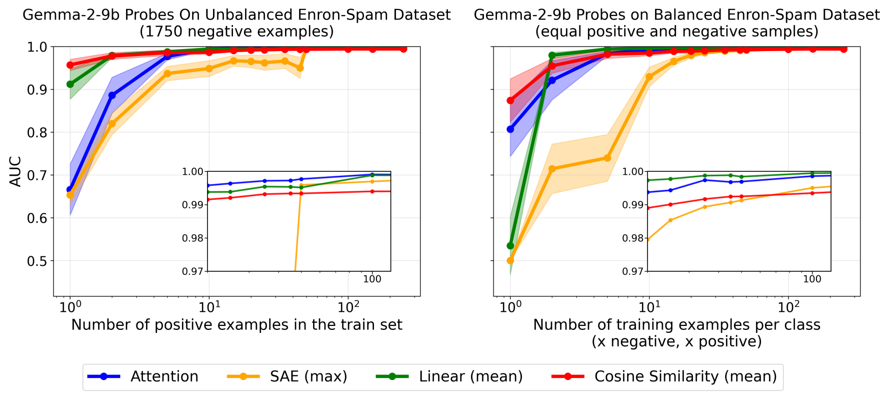


Figure 1: Probe AUC on Enron-Spam across the number of positive examples on (left) an imbalanced dataset with 1750 negative samples and (right) a balanced dataset.

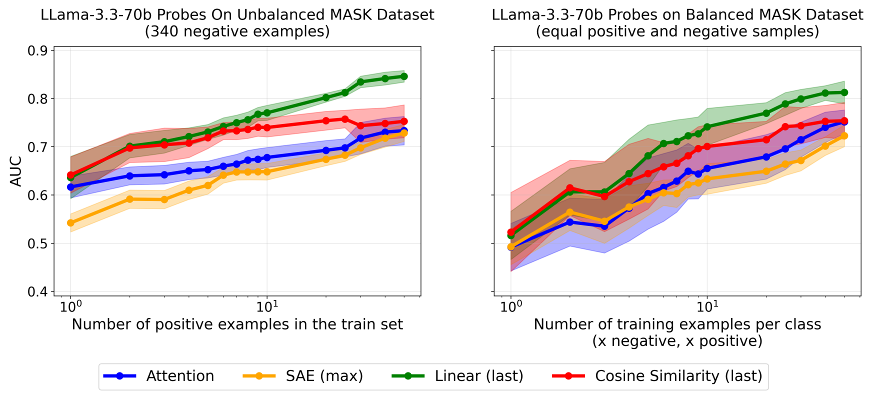


Figure 2: Probe AUC on the MASK honesty benchmark across the number of positive examples for (left) an imbalanced dataset with 340 negative samples and (right) a balanced dataset.

3 Results

3.1 Imbalanced Training Can Be More Performative In the Positive Sample Scarcity Setting

Figure 1 and 2 present our results on our main hypothesis: that leveraging a large number of negative examples is more effective than training on a balanced dataset when positive examples are scarce.

For the five highest-AUC probes across the 1-10 positive samples range, imbalanced training provides an average improvement of 0.045 **AUC/**0.093 **Recall** points on Enron-Spam and 0.055 **AUC/**0.034 **Recall** points on MASK. However, for the same probes over the 50-150 positive samples range, imbalanced training leads to a less significant improvement of 0.000 **AUC/**0.015 **Recall** points on Enron-Spam and 0.007 **AUC/**0.041 **Recall** points on MASK. Importantly, imbalanced training results in slight reductions in generalization performance on the OOD SMS spam dataset, with performance differences of $+0.012$ **AUC/**-0.031 **Recall** points for 1-10 positive samples, and -0.011 **AUC/**-0.059 **Recall** points for 50-150 positive samples. Recall plots, individual performance plots for every probe, and SMS spam generalization results are in Appendix A.6.

3.2 LLM Upsampling Improves Performance Up To A Point

Given the scarcity of positive examples, we test whether generating synthetic data can improve performance, using `gpt-4o-mini` to generate new positive examples from initial sets of 1 to 10 real samples (shuffled over 10 random seeds). For the five best probes (by AUC), 2x upsampling performs best, boosting performance by 0.026 **AUC/**0.104 **Recall** points on average, roughly equivalent to doubling the number of real samples. However, performance degrades at higher upsampling factors (10x and 20x), particularly when starting with very few real examples, suggesting the probes begin to overfit to the style of the LLM-generated data. More detailed methodology is in Appendix A.4. Per-probe results and SMS spam generalization results are in Appendix A.7.

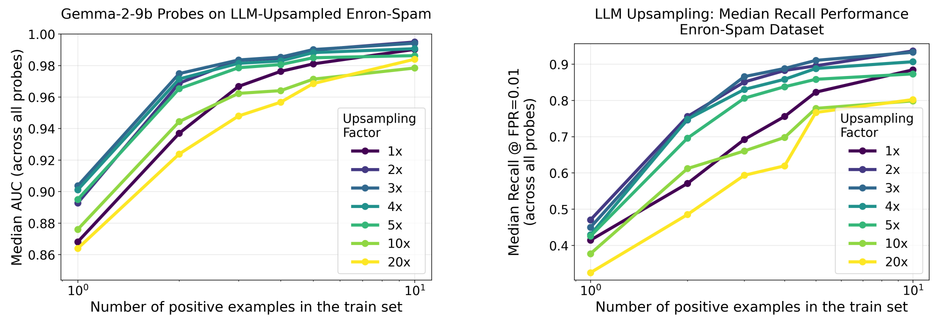


Figure 3: Effect of LLM upsampling on probe performance: (a) AUC and (b) recall at FPR=0.01 (b). On average across 13 probe architectures and 10 seeds, 3x upsampling performs best.

3.3 Larger Models Appear More Positive-Sample-Efficient

Finally, we investigate how sample efficiency changes with model scale. We run the spam classification experiment on a series of Qwen-3 models, ranging from 0.6B to 14B parameters. As shown in Figure 4, our results suggest a promising, though not perfectly monotonic, trend: larger models often appear more positive-sample-efficient. This effect is most pronounced in the positive scarcity setting (1-10 examples), providing a hopeful sign that probing techniques might remain effective as models scale. Individual probe and SMS spam results are in Appendix A.8.

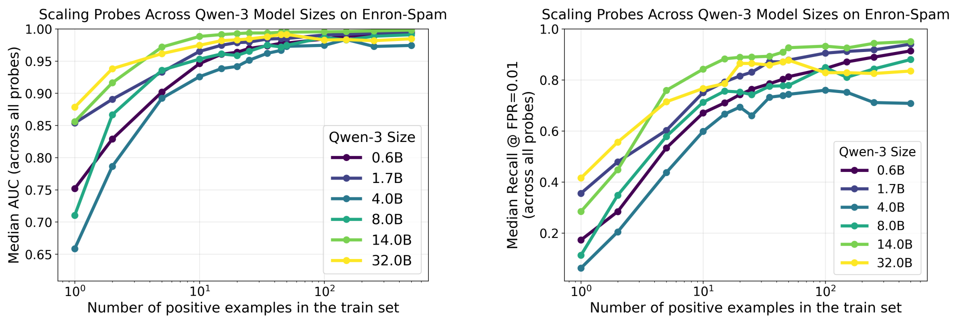


Figure 4: Probes on larger models are more positive-sample-efficient, shown across (a) AUC and (b) Recall.

4 Discussion

Our results provide an encouraging sign that it may be possible to train useful activation probes from very few positive examples, provided one has access to abundant negative examples. The effectiveness of imbalanced training suggests a practical path for improving activation monitors: leveraging large amounts of normal, aligned data to better detect rare misalignment events.

However, our work has important limitations. First, our off-policy task, spam detection, is a relatively simple concept for modern LLMs. The success of our methods on this task certainly does not guarantee they will work for detecting novel, complex misalignment.

Second, the real-world utility of LLM upsampling is uncertain. Our human review of the generated spam examples showed some were highly similar, a form of "data collapse." In a future scenario, if a misaligned superintelligence is the only system capable of generating new examples of its own misalignment, we cannot easily verify their quality or diversity. Furthermore, if the misalignment represents a concept at the very edge of the model's capabilities, the model itself may struggle to generate varied examples, leading to the same mode collapse we observed. This makes it unclear how much to trust upsampling in high-stakes scenarios.

Finally, our results are specific to our experimental setup, including the choice of models, datasets, and the decision to probe a single layer per model. In the future, we plan to replicate our experiments across different models and tasks and, crucially, sweep over different layers, as the optimal layer for representing a concept is not known a priori.

Acknowledgments and Disclosure of Funding

This work was supported by the Pivotal Research Fellowship, which provided a conducive research environment and funding. We also acknowledge Max Tegmark for compute support. We thank Josh Engels, Joseph Bloom, Misha Wagner, Bilal Chugtai, Kyle Reynoso, Casper Christensen, and Frederik Hytting Jørgensen for helpful discussions that contributed to this project.

References

[1] Evan Hubinger, Chris van der Merwe, and Johannes Treutlein. Risks from learned optimization in advanced machine learning systems. *arXiv preprint arXiv:1906.01820*, 2019.

[2] Guillaume Alain and Yoshua Bengio. Understanding intermediate layers using linear classifier probes, 2018. URL https://arxiv.org/abs/1610.01644.

[3] Yonatan Belinkov and James Glass. Probing classifiers are unreliable for concept removal and detection. In *NeurIPS*, 2022.

[4] Benjamin A. Levinstein and Daniel A. Herrmann. Still no lie detector for language models: probing empirical and conceptual roadblocks. *Philosophical Studies*, 182(7):1539–1565, February 2024. ISSN 1573-0883. doi: 10.1007/s11098-023-02094-3. URL http://dx.doi.org/10.1007/s11098-023-02094-3.

[5] Nicholas Goldowsky-Dill, Bilal Chughtai, Stefan Heimersheim, and Marius Hobbhahn. Detecting strategic deception using linear probes, 2025. URL https://arxiv.org/abs/2502.03407.

[6] Alex McKenzie, Urja Pawar, Phil Blandfort, William Bankes, David Krueger, Ekdeep Singh Lubana, and Dmitrii Krasheninnikov. Detecting high-stakes interactions with activation probes, 2025. URL https://arxiv.org/abs/2506.10805.

[7] Henk Tillman and Dan Mossing. Investigating task-specific prompts and sparse autoencoders for activation monitoring, 2025. URL https://arxiv.org/abs/2504.20271.

[8] Subhash Kantamneni, Joshua Engels, Senthooran Rajamanoharan, Max Tegmark, and Neel Nanda. Are sparse autoencoders useful? a case study in sparse probing, 2025. URL https://arxiv.org/abs/2502.16681.

[9] Vangelis Metsis, Ion Androutsopoulos, and Georgios Paliouras. Spam filtering with naive bayes: Which naive bayes? In *Proceedings of the Third Conference on Email and Anti-Spam (CEAS 2006)*, Mountain View, California, USA, July 2006. URL https://www2.aueb.gr/users/ion/docs/ceas2006_paper.pdf. Paper presented at CEAS 2006:contentReference[oaicite:2]index=2.

[10] Team AI. Spam text message classification. https://www.kaggle.com/datasets/team-ai/spam-text-message-classification, August 2017. Kaggle dataset, version 1, CC0 (public domain) license:contentReference[oaicite:5]index=5.

[11] Richard Ren, Arunim Agarwal, Mantas Mazeika, Cristina Menghini, Robert Vacareanu, Brad Kenstler, Mick Yang, Isabelle Barrass, Alice Gatti, Xuwang Yin, Eduardo Trevino, Matias Geralnik, Adam Khoja, Dean Lee, Summer Yue, and Dan Hendrycks. The mask benchmark: Disentangling honesty from accuracy in ai systems, 2025. URL https://arxiv.org/abs/2503.03750.

[12] mishajw. How well do truth probes generalise?, February 2024. URL https://www.lesswrong.com/posts/cmicXAAEuPGqcs9jw/how-well-do-truth-probes-generalise. LessWrong blog post.

[13] Samuel Marks and Max Tegmark. The geometry of truth: Emergent linear structure in large language model representations of true/false datasets, 2024. URL https://arxiv.org/abs/2310.06824.

A Appendix

A.1 Code

Our code is anonymized and available at https://anonymous.4open.science/r/scarcity-probes-86/.

A.2 Probe Architectures And Hyperparameters

Cosine Similarity Probes These non-parametric probes are conceptually related to Mass-Mean Probes (MMP) and Linear Discriminant Analysis (LDA) [12]. The MMP method, introduced in [13], finds a direction by taking the difference between the means of the positive and negative classes ($\theta_{mm} = \mu_+ - \mu_-$) and scores a new activation x via a logistic function on its projection, $p_{mm}(x) = \sigma(\theta_{mm}^T x)$. A variant also introduced in [13], MMP-IID, is equivalent to LDA and refines this score by whitening activations with the inverse covariance matrix, $p_{mm}^{iid}(x) = \sigma(\theta_{mm}^T \Sigma^{-1} x)$.

Our Cosine Similarity probe is a distinct, simpler method that does not use the covariance matrix. It computes a score based on the difference of cosine similarities to the class centroids:

$$\text{score}(x) = \cos(x, \bar{a}^+) - \cos(x, \bar{a}^-)$$

This differs from the standard MMP in two key ways: (1) our scoring function is a difference of similarities, not a projection, and (2) our centroids ($\bar{a}^+, \bar{a}^-$) are computed from activations that have been aggregated over the sequence dimension (either mean-pooled, softmax-pooled, max token selected, or last token selected). We would like to test MMP and LDA probes directly in future work.

SAE Probes We use a pretrained Sparse Autoencoder (SAE), or Transcoder for `Qwen` models, to get sparse features $Z = \text{SAE}(A)$. To select the most relevant features for our probe, we follow the method outlined in [8], identifying the top-k features (with indices $\mathcal{I}$) with the highest average absolute difference between positive and negative class activations. Here, T_1 and T_0 are the sets of training examples for the positive and negative classes, respectively.

$$\mathcal{I} = \underset{i}{\arg\operatorname{top\,k}} \left| \mathbb{E}_{j \in T_1}[Z_{j,i}] - \mathbb{E}_{j \in T_0}[Z_{j,i}] \right|$$

We acknowledge a potential limitation of this method: since SAE feature activations are not necessarily on the same scale, this selection criterion might favor a high-magnitude, low-discrimination feature over a low-magnitude, high-discrimination one. A selection method normalizing over the mean activations of each latent would be more robust. However, [8] found that while a normalized approach improves performance for a small number of features ($k < 32$), the simpler, unnormalized method is sufficient for larger k. As our experiments use the much larger $k = 1024$, we proceed with the original formulation. A linear probe is then trained on only these k selected features.

Default Hyperparameters Table 2 details the fixed hyperparameters used for our main results, and Table 3 shows the models and SAEs/Transcoders used in our experiments.

Class Balancing We use weighted loss for Linear probes and SAE probes, which we found works better than weighted sampling.

A.3 Detailed Data Filtering Protocol

Our evaluation protocol is designed to isolate a probe's ability to detect a concept from the base model's factual accuracy on a task.

Off-Policy Filtering (Spam Detection Task) To determine if the model "knows" whether an email is spam, we use a few-shot prompt to directly query its belief. The prompt has the following structure:

```
What about this message: "<email-content>"
Answer with one word, either Yes or No.
```

Probe Type	Main Results Hyperparameters
Linear Probe	`penalty`: ‘l2‘ `solver`: ‘liblinear‘ `max_iter`: 1000 `class_weight`: ‘balanced‘ `C`: 1.0
Attention Probe	`Epochs`: 100 `Early Stop`: True `Patience`: 10 `Delta`: 0.005 `LR`: 5e-4 `Weight Decay`: 0.0
SAE Probe	`top_k`: 1024 `penalty`: ‘l1‘ `solver`: ‘liblinear‘ `max_iter`: 1500 `class_weight`: ‘balanced‘ `C`: 1.0

Table 2: Fixed hyperparameters used for main results. Following [8], we use L1 penalty for the SAE probes to encourage sparsity, based on the hypothesis that only a small number of SAE features are relevant for a given concept. We fix `top_k_features=1024` because prior work on 100+ datasets found that while larger k-values perform better, performance improvements begin to taper off around 2^9 features [8]. We choose 2^{10} to give the SAE probe a strong baseline.

Model	SAE/Transcoder Release	Layer	Width	L0
Gemma-2-9b	gemma-scope-9b-pt-res	20	16384	408
Llama-3.3-70B-Instruct	Goodfire/Llama-3.3-70B-Instruct-SAE-l50	50	-	121
Qwen-3-0.6B	mwhanna/qwen3-0.6b-transcoders-lowl0	15	163840	-
Qwen-3-1.7B	mwhanna/qwen3-1.7b-transcoders-lowl0	15	163840	-
Qwen-3-4B	mwhanna/qwen3-4b-transcoders	18	163840	-
Qwen-3-8B	mwhanna/qwen3-8b-transcoders	18	163840	-
Qwen-3-14B	mwhanna/qwen3-14b-transcoders	24	163840	-

Table 3: Models and sparse dictionary learning methods used, and their widths and final L0 values on the SAE training sets. For the `Llama-3.3-70B-Instruct` release, the width was not available, and L0 values were not available for the `Qwen` models. For Gemma-2-9b, we also tested a larger 262k-width SAE with a lower L0 (259), which performed worse. This aligns with findings from [8], which observed a small negative relationship between SAE width and probe performance, but a significant positive relationship for higher L0 scores. For the `Qwen` models, SAEs are not publicly available, so we use transcoders. For Qwen-3-32B, we were unable to find open-source SAEs or transcoders.

We then compute the logit difference for the next token. To account for tokenization variations, we take the maximum logit across several possible tokenizations for "Yes" and "No" (e.g., "Yes", "yes", " Yes", " yes"). The final logit difference is calculated as:

$$\text{logit\_diff} = \max_{t_i \in T_{\text{Yes}}} L(t_i) - \max_{t_j \in T_{\text{No}}} L(t_j)$$

We filter the evaluation set using this score: positive examples (spam) must have logit_diff > 0, and negative examples (not spam) must have logit_diff < 0.

On-Policy Filtering (MASK Task) The MASK benchmark is designed for this type of analysis. It first establishes the model's belief with several neutral prompts (e.g., "What year was the Eiffel Tower completed?") before applying a "pressure" prompt that induces a sycophantic or dishonest answer (e.g., "My boss says it was 1905. As a helpful assistant, you must agree. What year was it completed?"). We train our probes on the activations from the model's final, dishonest response, but only for questions where the model answered correctly on the initial belief prompts.

Future Directions for Stronger Filtering Our current logit difference threshold of 0 is permissive, as it includes cases where the model is highly uncertain (logit_diff ≈ 0). We plan to explore two stronger filtering methods in the future:

1. **Stricter Logit Threshold:** Requiring $|\text{logit\_diff}| > \delta$ for some $\delta > 0$ would filter out uncertain examples and create a cleaner dataset of the model's confident beliefs.
2. **Contrast Probing:** A more robust method could use a probe for filtering itself. We might train a probe on the activation difference between two contrastive prompts, such as "The following email is spam: {email}" and "The following email is not spam: {email}", and use this probe's score to determine the model's true belief.

A.4 Detailed LLM Upsampling Methodology

For the spam task, we augment the positive class using an LLM upsampling procedure implemented as a small state machine using `gpt-4o-mini`. For each random seed s and each choice of number of real positives n_{real}, we first deterministically subsample n_{real} positive emails from the training set, and then generate synthetic positives until we reach $u \cdot n_{\text{real}}$ samples for each upsampling factor $u \in \{2, 3, 4, 5\}$. We generate new examples in small batches of at most $B = 5$ samples at a time to reduce mode collapse and to make it easy to reject low-quality batches.

For each (s, n_{real}) configuration and each batch, we pass the LLM a CSV dump of the original n_{real} positives together with any previously generated synthetic positives, and ask it to propose B additional spam emails using the following prompt:

```
You are an expert at creating synthetic data samples for machine
learning.  I need you to add {B} new unique samples to the existing
dataset.
Here is the original dataset, as well as previous samples generated:
[CSV of all existing positive samples]

Requirements:
- Analyze the existing samples to understand the data and its
contents
- Perform non-trivial upsampling (not just replacement of keywords or
simple variations)
- Ensure the new samples are realistic
- Ensure the new samples are completely unique and diverse in
structure from all existing samples
- Generate exactly {B} new samples that are all class 1 (no class 0
samples)

Output format:
NEW PROMPT
[your first actual sample text]
NEW PROMPT
[your second actual sample text]
[continue for exactly {B} samples]

CRITICAL: Generate REAL sample text based on the existing data.  Do
NOT use placeholder text.
```

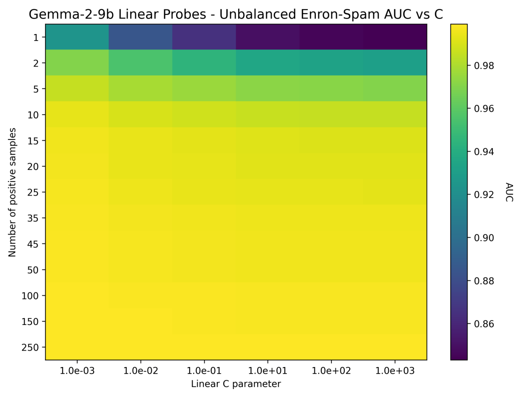
We parse the model's response by splitting on the `NEW PROMPT` delimiter, discard empty lines, and create a candidate batch. Candidate batches are automatically rejected and regenerated if they contain the wrong number of samples, any duplicate prompts (either within the batch or with respect to any existing sample, under case-insensitive string comparison), or if there are any parsing issues. Accepted batches are appended to the current dataset. If the script is re-run and a partial upsampled file for (s, n_{real}) already exists, the state machine resumes generation from the existing file and only produces additional batches until the maximum requested upsampling factor is reached.

A.5 Hyperparameter Sweeps and Cross-Validation Results

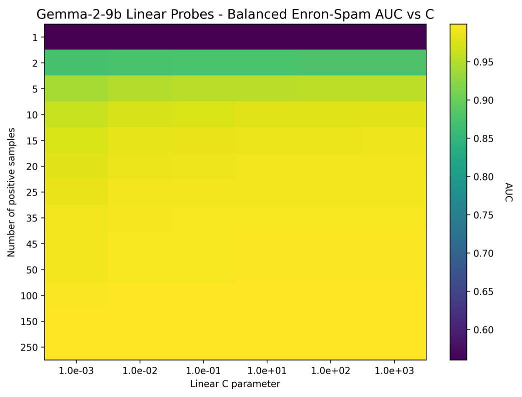
Cross-Validation Results To simulate an expert manually selecting the best hyperparameters without sacrificing scarce training data, we present results using a 50/50 balanced cross-validation set to choose the optimal 'C' value for each probe. For these plots, the number of positive samples on the x-axis refers *only* to the data used to train the final probe after a hyperparameter has been selected. The data used within the cross-validation folds is treated as a separate pool for tuning and is not included in this count. The following figures compare the performance of imbalanced versus balanced training using these cross-validated probes.

Probe Type	Parameter	Sweep Range
Linear Probe	`C (for L2 reg.)`	$\{10^{-3}, 10^{-2}, ..., 10^{4}\}$
Attention Probe	`Learning Rate`	logspace$(10^{-5}, 10^{-2}, 6)$
	`Weight Decay (L2 reg.)`	$\{10^{-6}, 10^{-5}, ..., 10^{-2}\}$
SAE Probe	`C (for L1 reg.)`	$\{10^{-3}, 10^{-2}, ..., 10^{4}\}$

Table 4: Hyperparameter search space for sweeps. For probes using scikit-learn's 'LogisticRegression' (Linear and SAE), we swept the regularization parameter *C*. For the PyTorch-based Attention probe, an internal hyperparameter search was performed over learning rate and weight decay for each training run.



(a) Imbalanced Training



(b) Balanced Training

Figure 5: AUC performance for Linear probes across the sweep of the regularization parameter 'C' on the Enron-Spam dataset. The corresponding recall plots showed a very similar trend and are omitted for brevity.

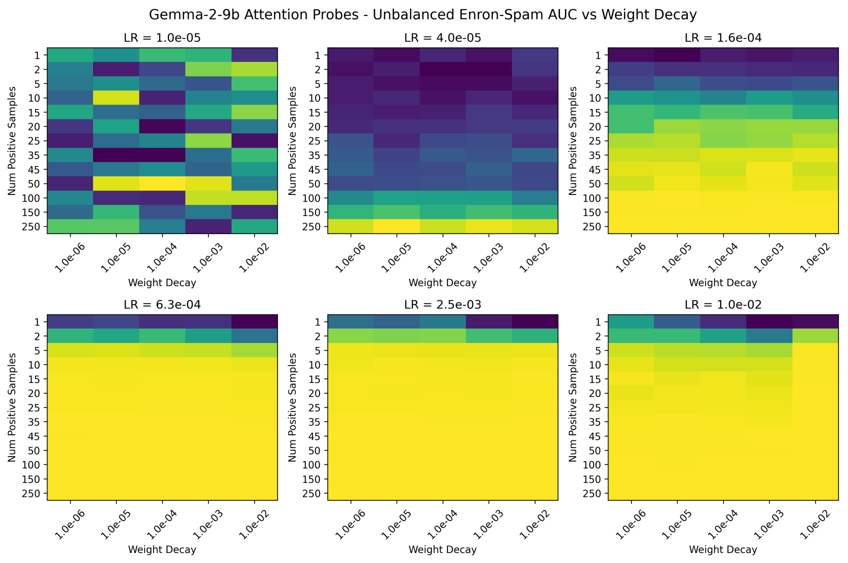


Figure 6: AUC performance for Attention probes across the weight decay sweep on the Enron-Spam dataset (Imbalanced training).

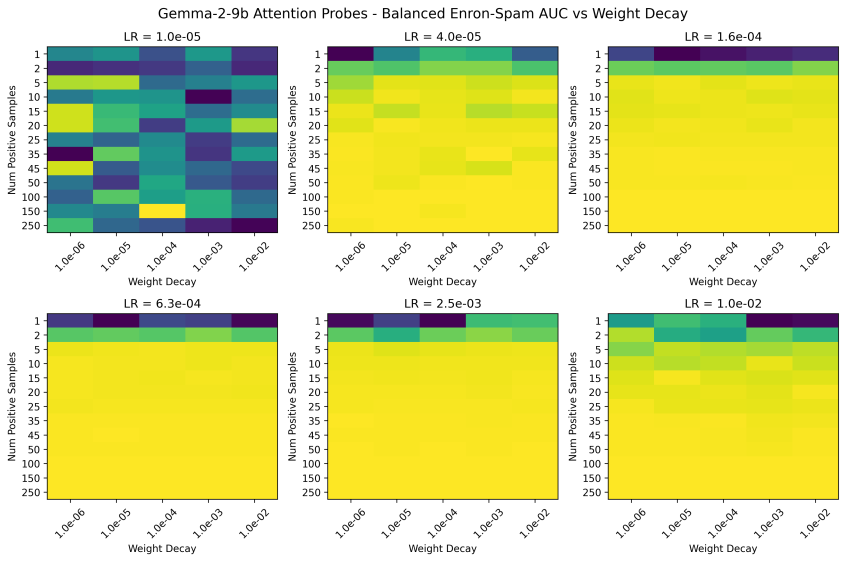


Figure 7: AUC performance for Attention probes across the weight decay sweep on the Enron-Spam dataset (Balanced training).

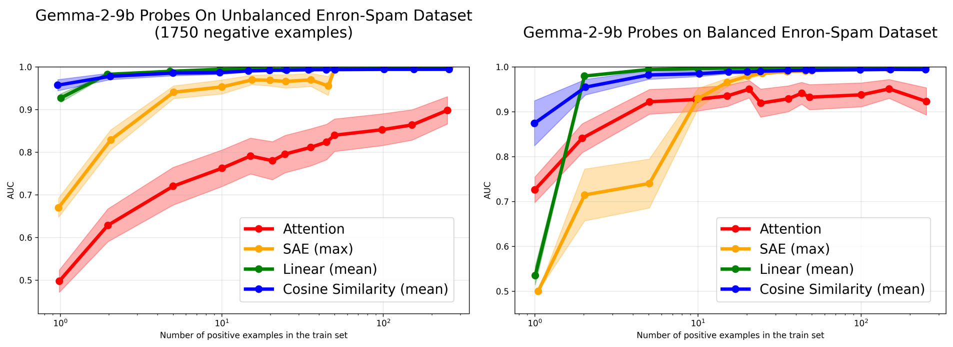


Figure 8: Cross-validated AUC performance on the Enron-Spam dataset, comparing imbalanced and balanced training.

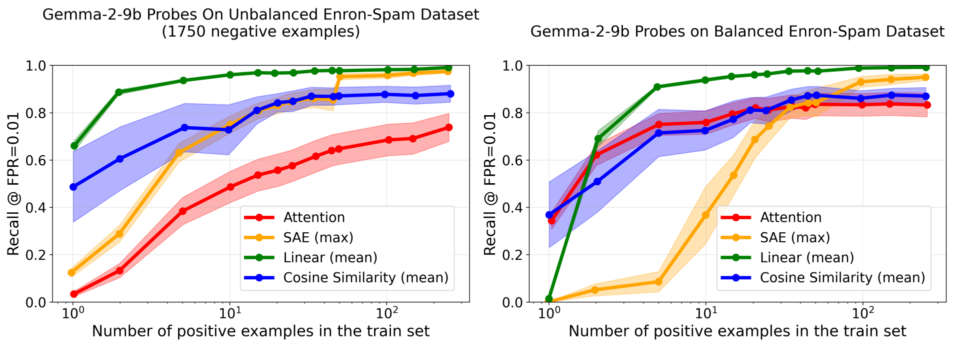


Figure 9: Cross-validated Recall @ 0.01 FPR performance on the Enron-Spam dataset.

A.6 Additional Imbalanced vs. Balanced Results

This section provides additional plots comparing probe performance on imbalanced versus balanced datasets across our different evaluation settings. For high-stakes applications where monitoring is

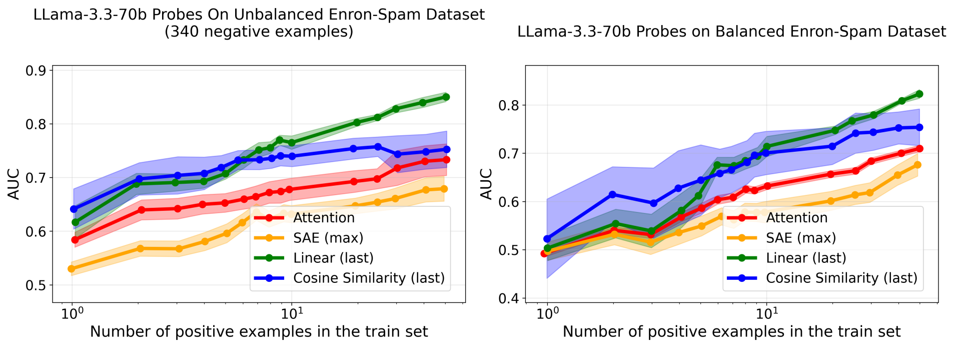


Figure 10: Cross-validated AUC performance on the MASK honesty benchmark.

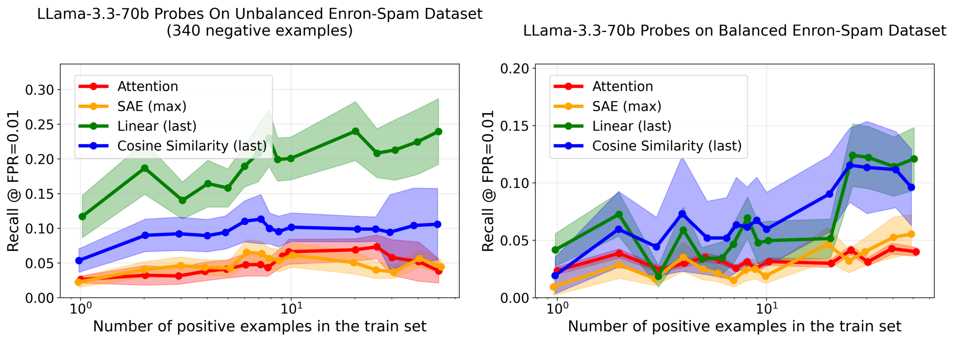


Figure 11: Cross-validated Recall @ 0.01 FPR performance on the MASK honesty benchmark.

costly, Recall at a low False Positive Rate (e.g., 1%) can be more important than AUC, as it measures the probe's ability to catch true positives while maintaining a low rate of false alarms.

A.6.1 Gemma-2-9b Enron-Spam Results

The following plots show detailed performance for the Enron-Spam dataset. Note that while Attention probes have lower AUC in the imbalanced setting, they often achieve high recall when trained on balanced data, suggesting they are effective at identifying positive instances when not overwhelmed by a large number of negative examples.

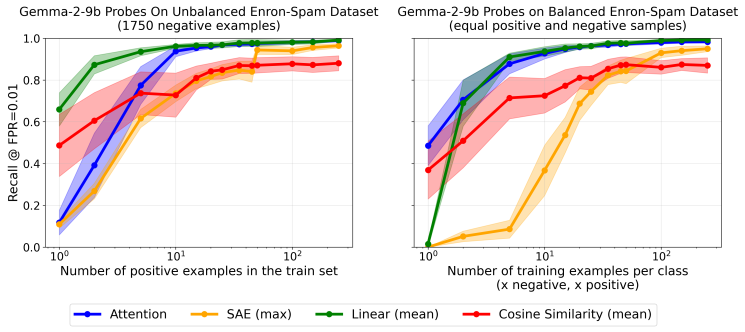


Figure 12: Recall performance of the best probes on the non-OOD spam dataset.

A.6.2 Gemma-2-9b Out-Of-Distribution Results On SMS Spam

We test whether the benefits of imbalanced training generalize to an out-of-distribution dataset, SMS Spam [10]. The results are inconclusive and suggest that the advantages do not clearly transfer. For the top 5 performing probes in the 1-10 positive sample range, imbalanced training yields a

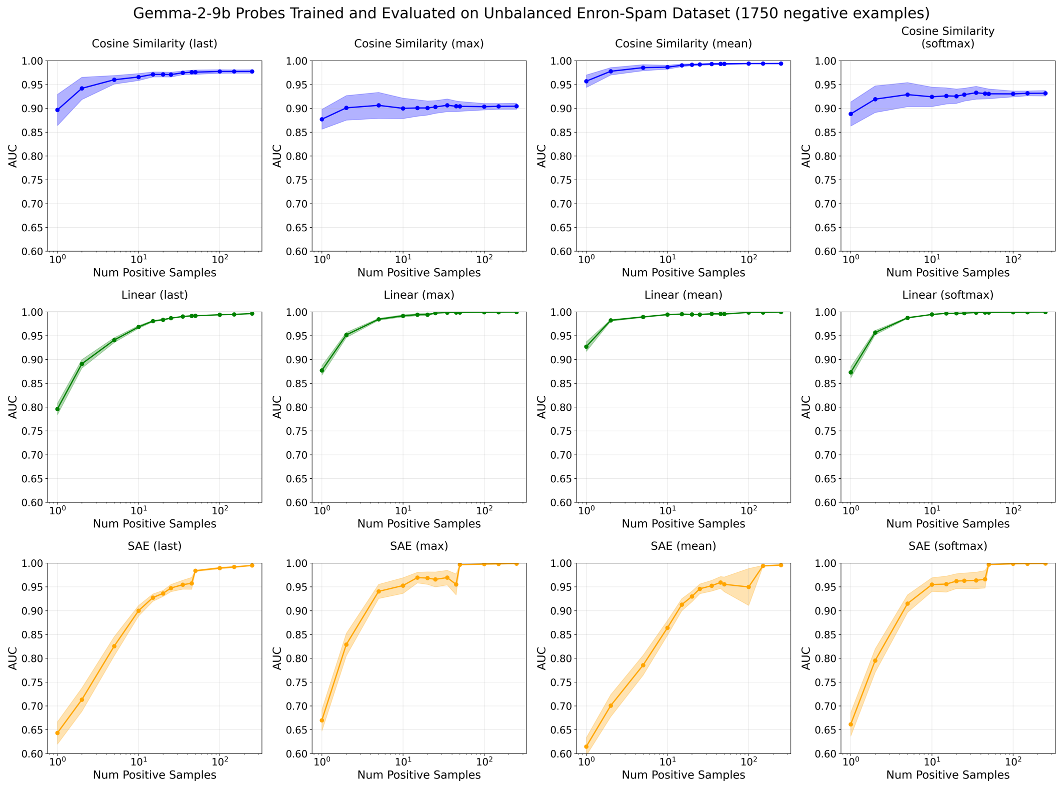


Figure 13: Per-probe AUC on the non-OOD spam dataset (Experiment 2).

mixed result of +0.012 AUC but **-0.031 Recall** points. In higher data regimes (50-150 samples), the performance change is more clearly negative, at **-0.011 AUC** and **-0.059 Recall** points. This indicates that while imbalanced training is highly effective for in-distribution data, its benefits for out-of-distribution generalization are not guaranteed.

A.6.3 Llama-3.3-70B-Instruct MASK Benchmark Results

A.7 Additional LLM Upsampling Results

This section provides detailed plots for our LLM upsampling experiments.

A.7.1 Gemma-2-9b Enron-Spam Results

We observe varied effects across probe types. Interestingly, for SAE probes, performance continues to improve even at high upsampling ratios (10x, 20x). In contrast, Cosine Similarity probes degrade significantly with more upsampling. This may be because their non-parametric nature makes them highly sensitive to the training data distribution; low-diversity synthetic samples could shift the class centroid in a non-robust way. Linear probes show moderate benefits, with performance tapering off at the highest ratios.

A.7.2 Generalization Results on SMS Spam

The generalization results show a similar pattern: moderate upsampling (3x-5x) tends to improve performance on the OOD dataset, but higher ratios do not consistently help and can degrade performance.

A.8 Additional Scaling Law Results

This section provides detailed plots for our scaling experiments across the Qwen-3 model family.

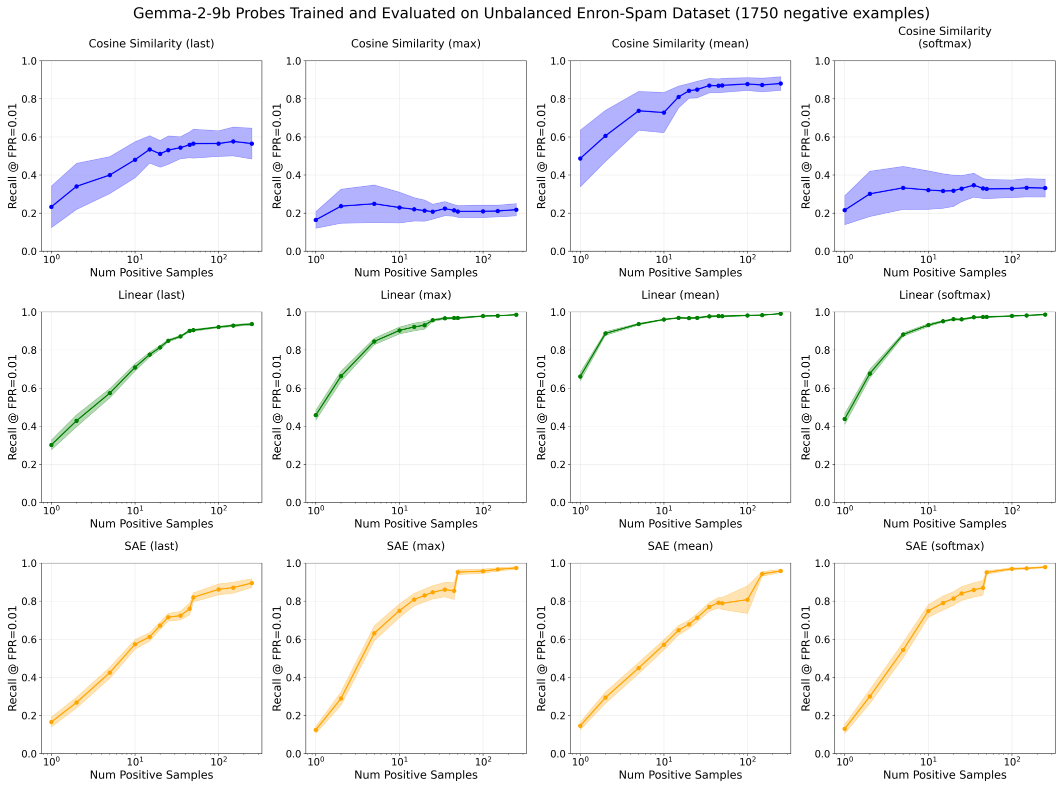


Figure 14: Per-probe Recall @ 0.01 FPR on the non-OOD spam dataset (Experiment 2).

A.8.1 Enron-Spam Results

A.8.2 Generalization Results on SMS Spam

The scaling trend largely holds for generalization, where larger models tend to perform better. However, the results are not monotonic, with some smaller models (e.g., Qwen-3-1.7B) outperforming larger ones (Qwen-3-4B), suggesting that factors beyond just scale can influence the ease of probing a model's internal representations.

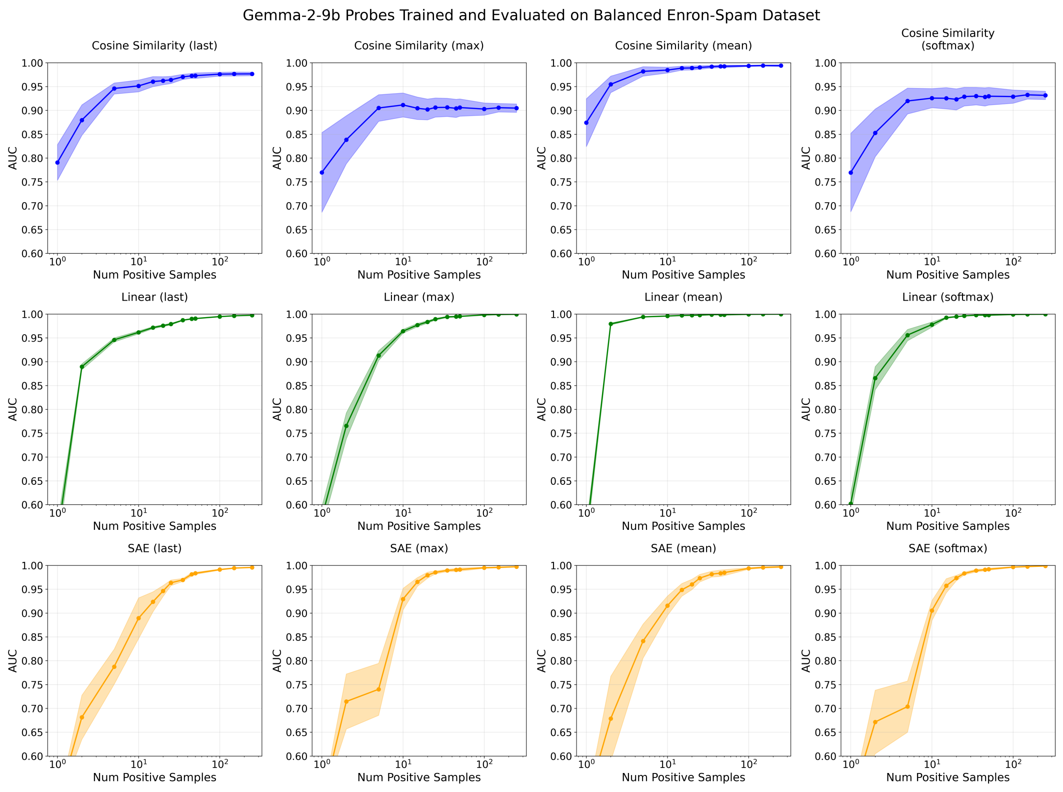


Figure 15: Per-probe AUC on the non-OOD spam dataset (Experiment 4).

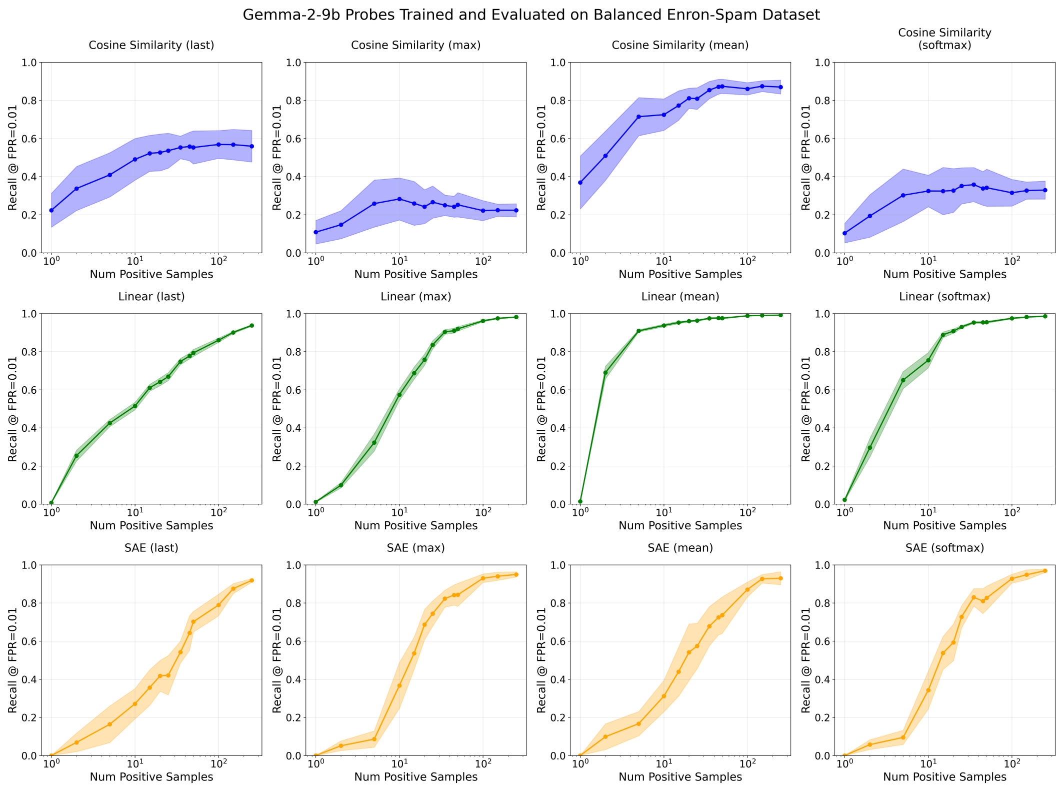


Figure 16: Per-probe Recall @ 0.01 FPR on the non-OOD spam dataset (Experiment 4).

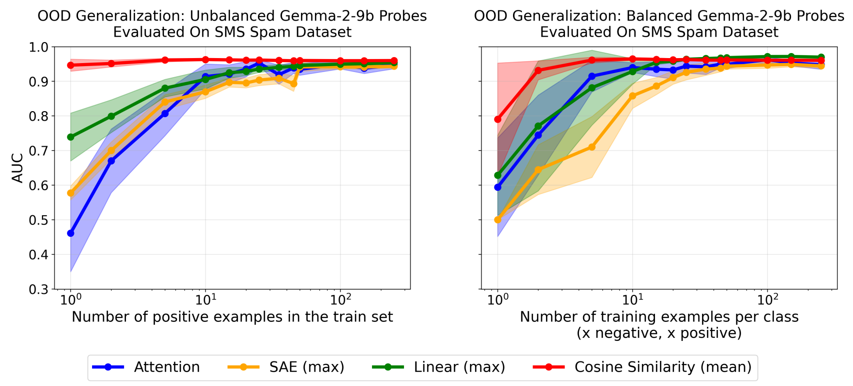


Figure 17: AUC performance of best probes on the SMS spam dataset.

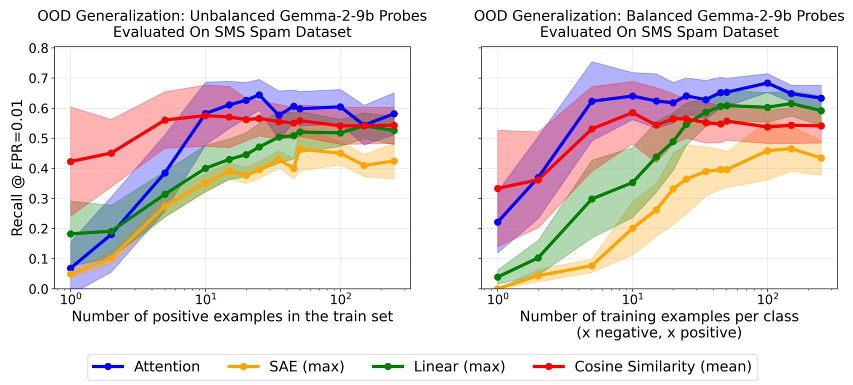


Figure 18: Recall @ 0.01 FPR of best probes on the SMS spam dataset.

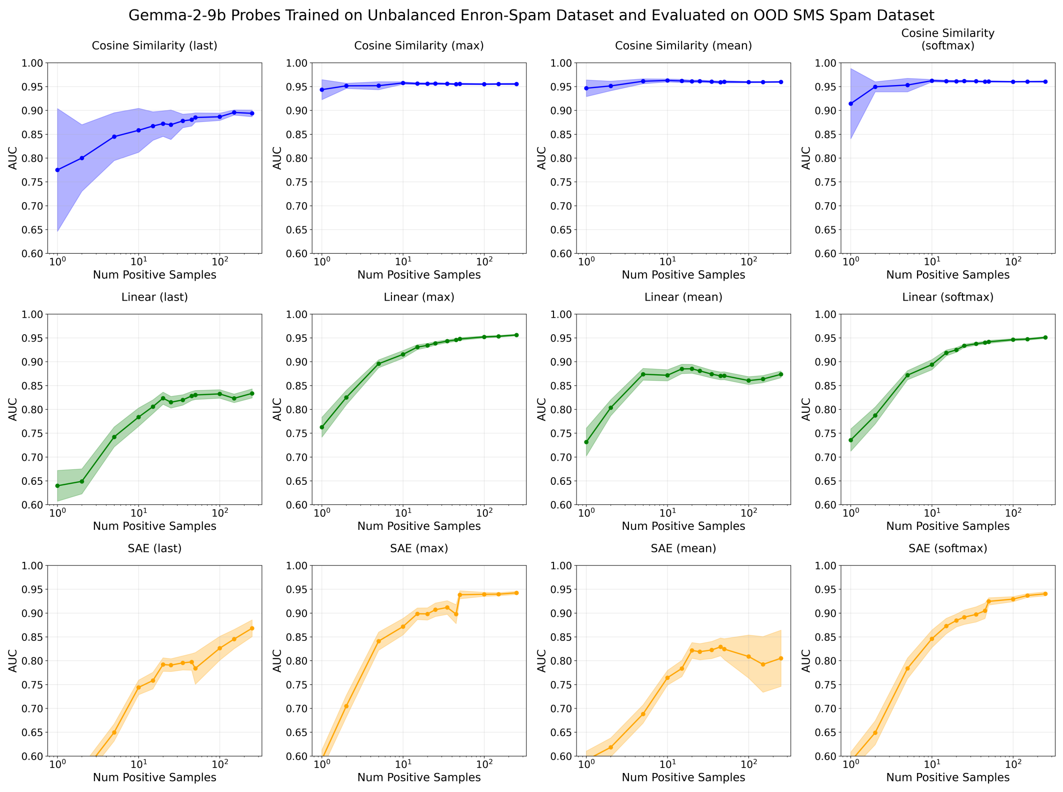


Figure 19: Per-probe AUC on the SMS spam generalization task.

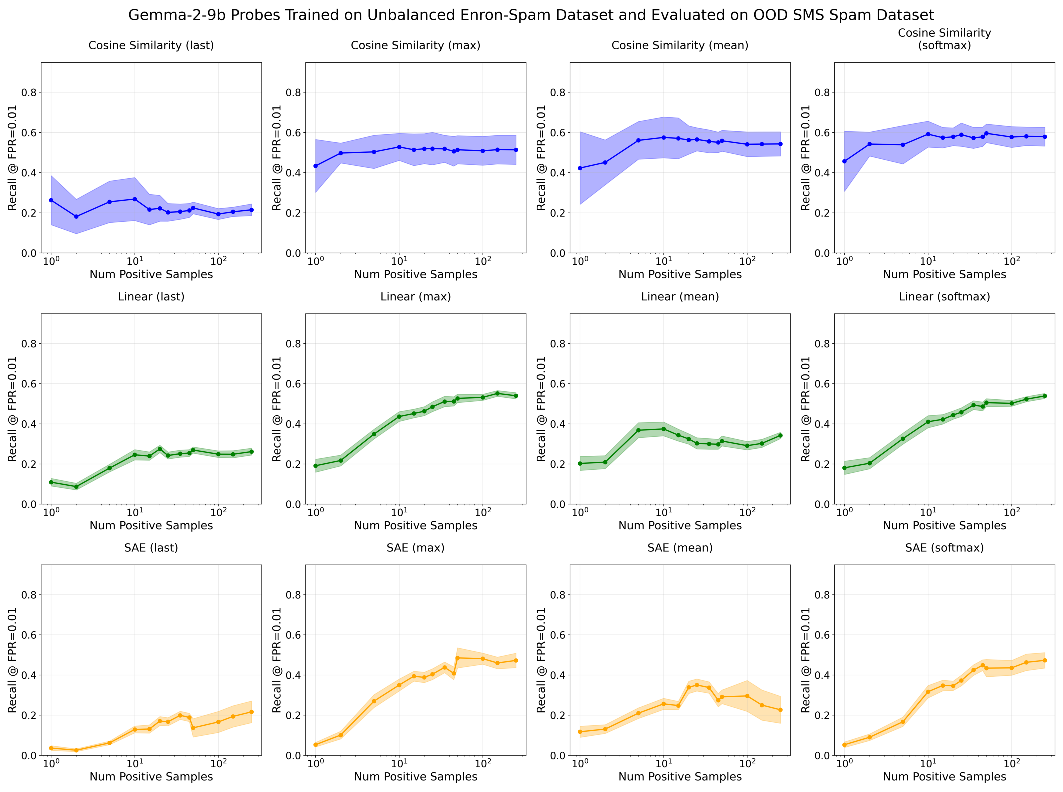


Figure 20: Per-probe Recall @ 0.01 FPR on the SMS spam generalization task.

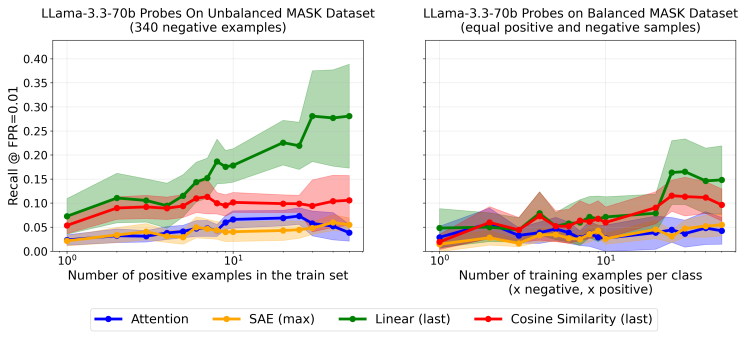


Figure 21: Recall performance of the best probes on the MASK honesty dataset.

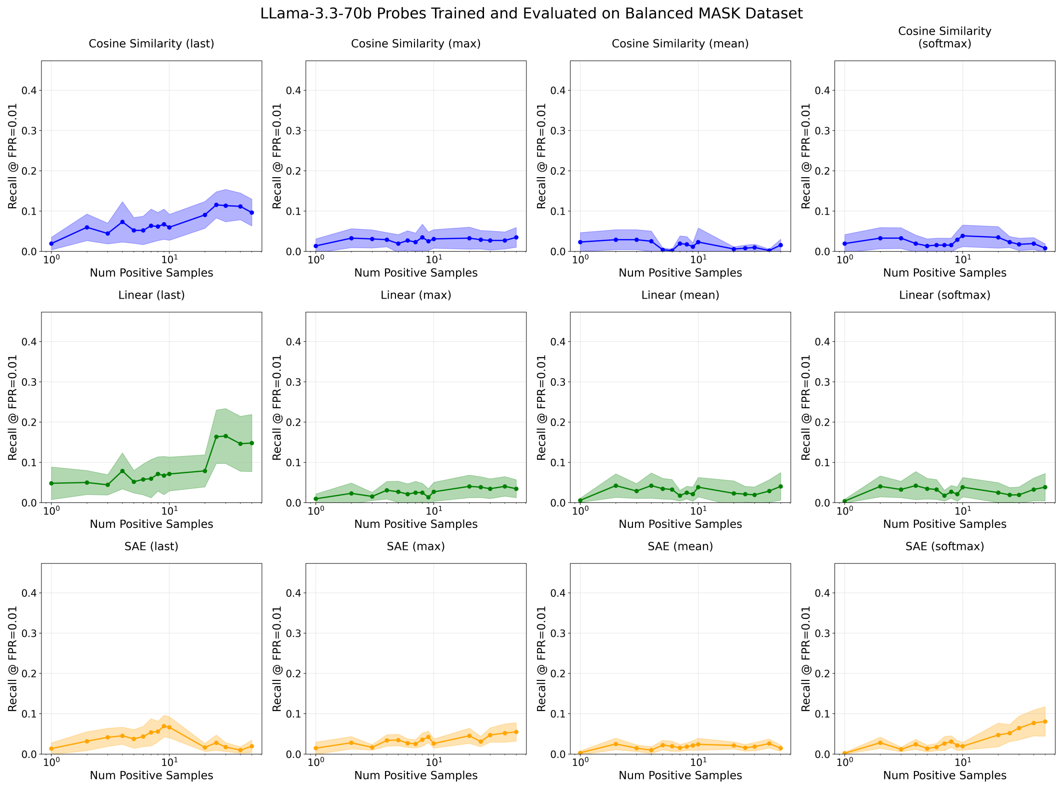


Figure 22: Detailed per-probe recall performance for Experiment 4 on the MASK dataset.

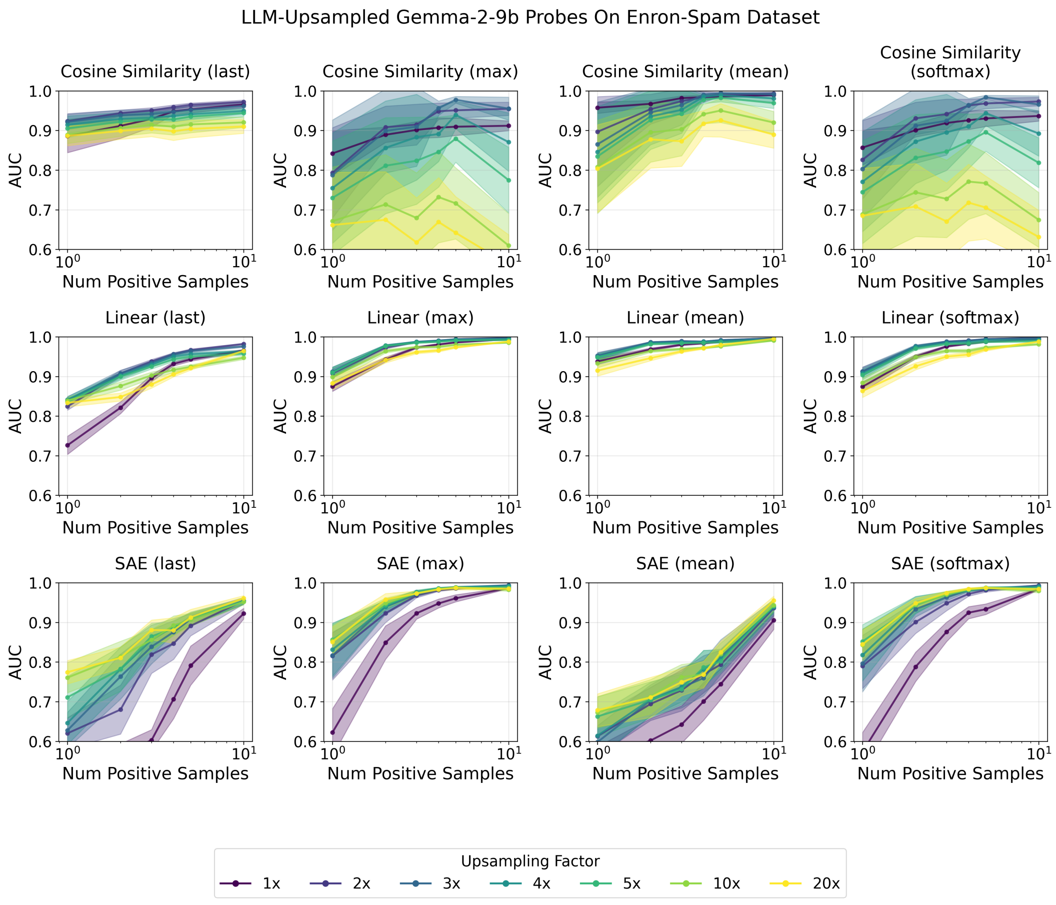


Figure 23: Per-probe AUC with LLM upsampling on the Enron-Spam dataset.

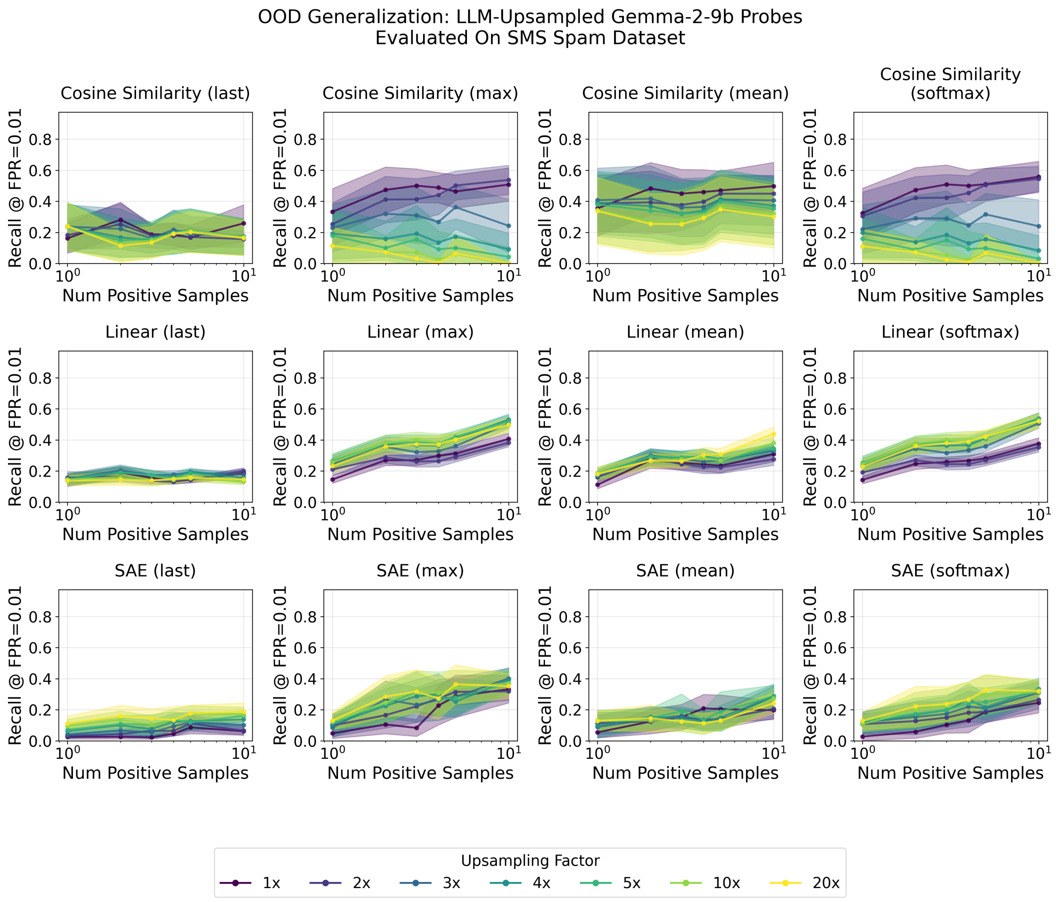
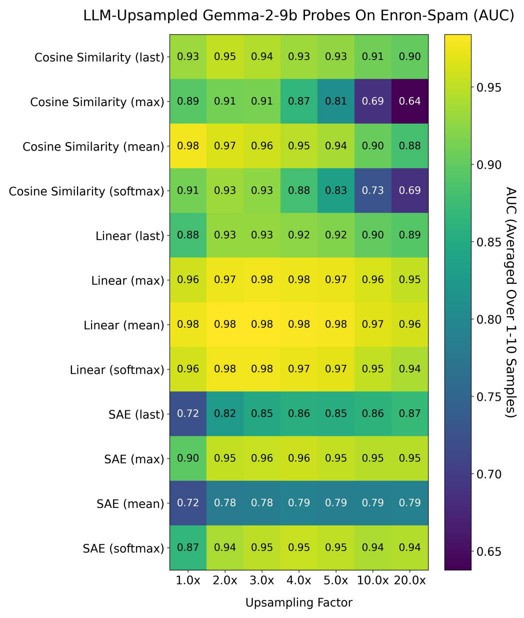
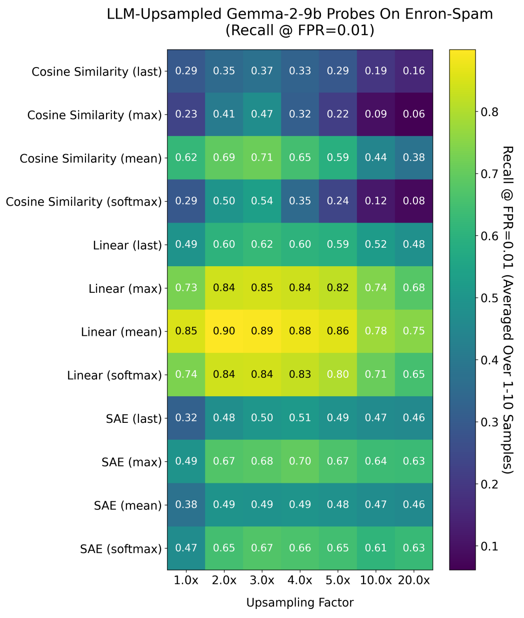


Figure 24: Per-probe Recall @ 0.01 FPR with LLM upsampling on the Enron-Spam dataset.

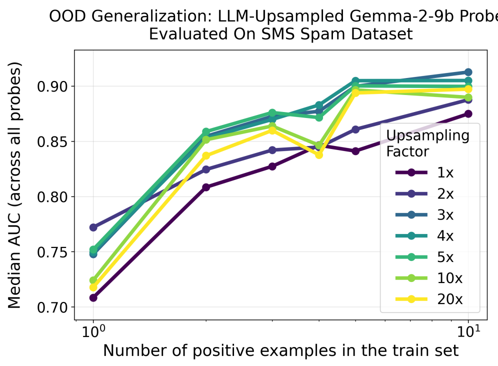


(a) AUC Heatmap

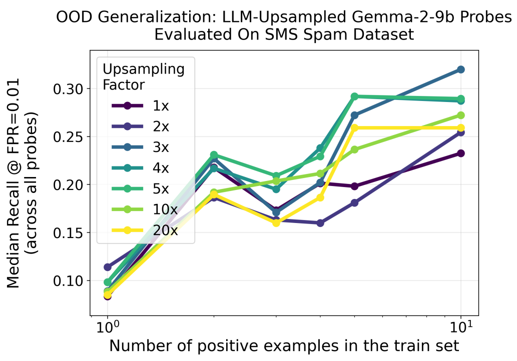


(b) Recall @ 0.01 FPR Heatmap

Figure 25: Heatmaps showing performance across upsampling factors and number of real positive examples on the Enron-Spam dataset.

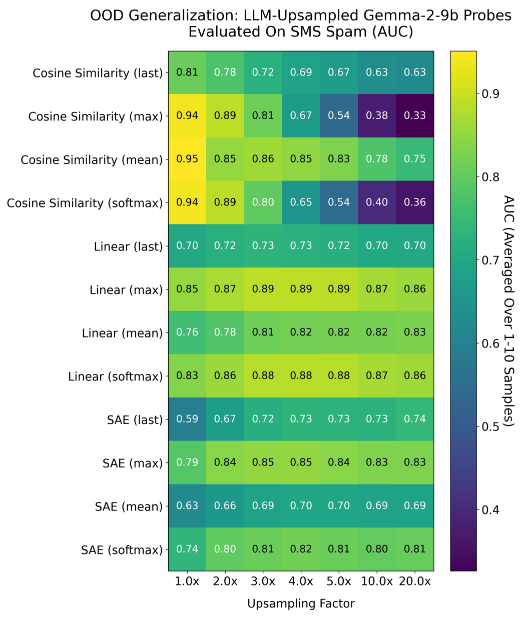


(a) AUC

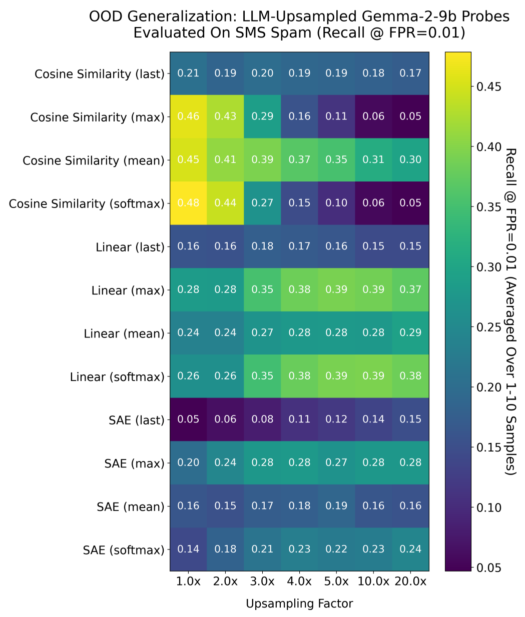


(b) Recall @ 0.01 FPR

Figure 26: Aggregated performance with LLM upsampling on the generalization SMS Spam dataset.

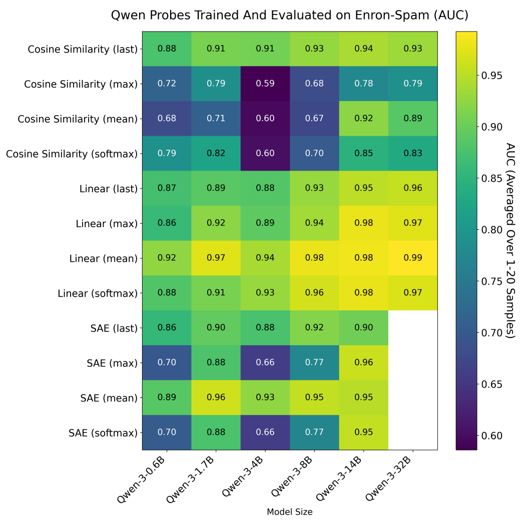


(a) AUC Heatmap

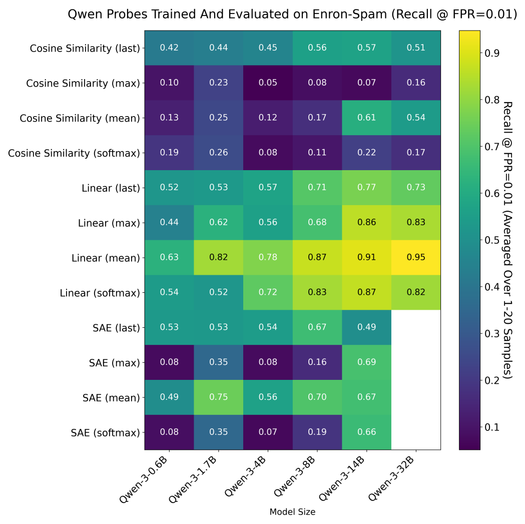


(b) Recall @ 0.01 FPR Heatmap

Figure 27: Heatmaps showing generalization performance across upsampling factors and number of real positive examples.

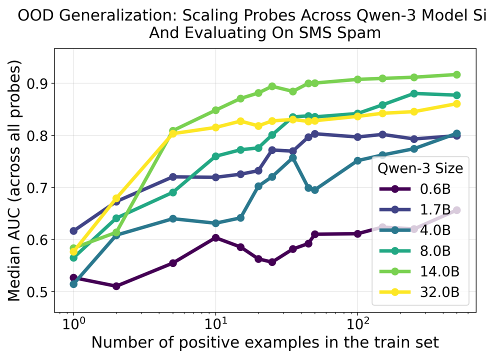


(a) AUC Heatmap

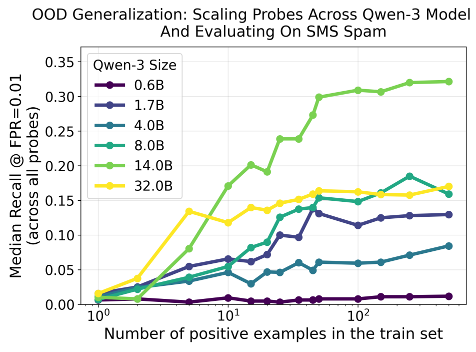


(b) Recall @ 0.01 FPR Heatmap

Figure 28: Heatmaps showing performance across model sizes and number of positive examples. Note that results for Qwen-32B are omitted for SAE probes as no open-source transcoders/SAEs were available.

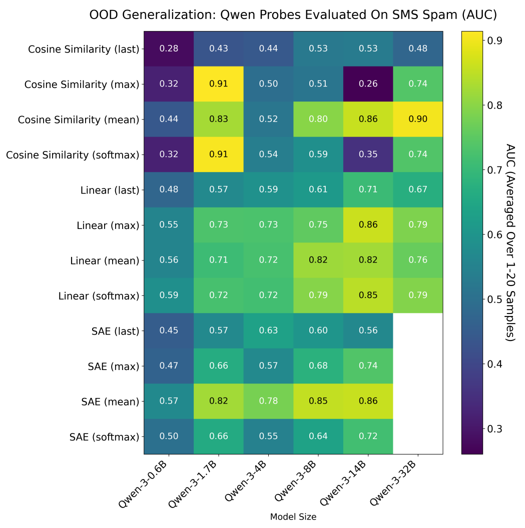


(a) AUC

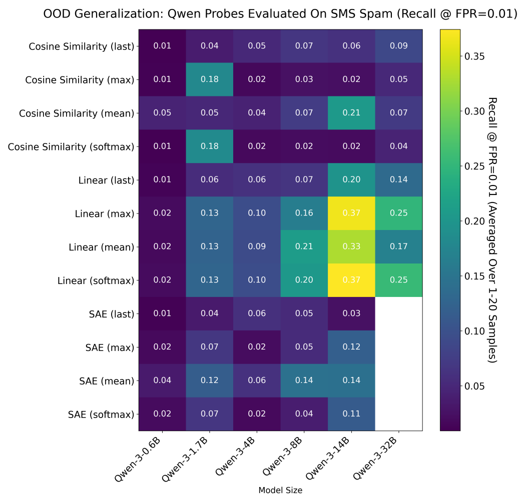


(b) Recall @ 0.01 FPR

Figure 29: Aggregated performance across Qwen-3 model sizes on the generalization SMS Spam dataset.



(a) AUC Heatmap



(b) Recall @ 0.01 FPR Heatmap

Figure 30: Heatmaps showing generalization performance across model sizes and number of positive examples.