ZG-STR: A Zero-Gradient Method for Detecting and Repairing Under-Trained LLM Tokens

Anonymous ACL submission

Abstract

Language models depend on fixed vocabularies, but a group of tokens appears so rarely during pre-training that models struggle to learn their meanings. This can lead to incorrect outputs during inference. Current solutions either need costly full fine-tuning or parameter-efficient fine-tuning (PEFT). To mitigate this issue, we introduce ZG-STR, a zero-gradient method used at inference time that fixes these weak tokens. ZG-STR finds these fragile tokens using Fisher information and repairs those tokens by using a healthy proxy token and injecting a steering vector. In our diverse assessment, ZG-STR boosts fragile-token understanding accuracy by as much as 12.5% with Llama-3.1-8B, showing improvements in five model families. Model's output consistency, measured by KL divergence, improves in 15 out of 17 models. We also discovered two architectural features, QKV bias and QK-Norm, that independently block steering. ZG-STR does not require parameter tuning and maintains safety, providing the first no-cost solution for a widespread issue in language models.

1 Introduction

Language models, particularly transformer-based architectures (Vaswani et al., 2017), have demonstrated remarkable performance across a wide range of tasks. Yet a persistent vulnerability arises from their fixed token vocabularies: a subset of tokens—termed glitch tokens or under-trained tokens—appear so rarely during pre-training that models fail to learn meaningful representations (Land and Bartolo, 2024). We identified these tokens as fragile tokens. In inference time, when such a token appears in the input stream, the hidden state at that position is noisy enough to generate unpredictable or degraded outputs (Land and Bartolo, 2024). Such fragile tokens exist in every tokenizer (Land and Bartolo, 2024) and can be triggered by ordinary user input.

These fixes will be very costly in terms of financial investment. Fine-tuning requires many hours of training on GPUs and also comes with the risk of catastrophic forgetting of the pre-trained model. The LoRA method (Hu et al., 2021), although parameter-efficient, still requires gradient calculation and adds new trainable parameters. Glitch-Cleaner (Fan et al., 2026) is a specific implementation of LoRA in fixing fragile tokens, which requires less than 0.1% extra trainable parameters. None of these fixes offers an inference-time fix to repair the culprit of the error, the corrupt token representations.

Our contribution is the proposal: *"**Z**ero-**G**radient **S**tructural **T**oken **R**e-Routing (ZG-STR)"*. This method functions by detecting and repairing fragile token representations during inference time. Specifically, the proposed approach utilizes diagonal Fisher information on a limited calibration dataset to detect fragile tokens. It then employs the cosine similarity in the frozen embedding space to locate healthy proxy tokens. Finally, the steering vector, mathematically, is the difference between the proxy's hidden-state centroid and the fragile token's hidden state, added to the residual stream at the optimal layers. This inference-time meddling requires no gradient computation and introduces no trainable parameters. The assessment of ZG-STR on 17 models with sizes from 0.8B to 20B across 10 research labs includes pure dense architectures (OLMo-3, Llama-3.1, IBM Granite-4.1, Falcon3, Phi-4, Mistral-7B), gated-delta network hybrid architectures (Qwen3.5 family), and mixture-of-experts (GPT-OSS). The accuracy of ZG-STR for fragile-token understanding on our

[0]Code for this project was developed with assistance from Claude Opus 4.6. All AI-generated code was reviewed, verified, and edited by the authors. The authors remain solely responsible for correctness and originality.

proposed custom benchmark reaches up to **12.5%** (Llama-3.1-8B) improvements over baseline. We observe improvement on five different model families: **Llama-3.1 (+12.5%), IBM Granite-4.1 (+8.8%), Falcon3 (+3.8%), GPT-OSS (+3.7%), and Gemma-4 (+2.8%)**. In addition, consistency, measured using the KL divergence between outputs of fragile and proxy models, is improved on 15 out of 17 models. Additionally, we discover two architectural factors that alone can prevent steering from working: QKV-bias (Vaswani et al., 2017) and QK-Norm (Henry et al., 2020). These observations are consistent with previous mechanistic studies on the interaction between steering vectors in the output-value circuit (Cheng et al., 2026). ZG-STR is the first study to investigate the effectiveness of zero-gradient approach in fragile tokens.

2 Related Work

The diagonal empirical Fisher has been used for model pruning (Theis et al., 2018; Singh and Alistarh, 2020), calibration (Guo et al., 2017), and identifying important parameters (Kirkpatrick et al., 2017) to a great extent. The term glitch token was introduced by Rumbelow and Watkins (2023), and subsequent works developed unsupervised methods to detect them via tokenizer analysis and weight-based indicators (Land and Bartolo, 2024). To mitigate these effects, Fan et al. (2026) proposed GlitchCleaner, a technique that uses LoRA to repair glitch-token impacts. For behavioral control, injecting steering vectors to the residual stream was introduced by Subramani et al. (2022) and Turner et al. (2023). As an empirical method, Contrastive Activation Addition (CAA; Rimsky et al., 2024) steers models by computing the mean difference of positive and negative example activations, while Representation Engineering (RepE; Zou et al., 2023) extends this technique to a broader range of behaviors. Recent works have also clarified when steering can succeed or fail; Cheng et al. (2026) showed that steering vectors act through the output-value (OV) circuit while largely ignoring the query-key (QK) circuit, whereas Liu (2026) reported that five families of fixed linear steering yield a negligible effect on Qwen2.5-7B (Qwen Team, 2024) and Qwen2 models in general Yang et al., 2024.

However, these existing approaches present distinct limitations. Mitigating glitch tokens via GlitchCleaner (Fan et al., 2026) requires gradient computation and adds a small fraction of extra parameters. Meanwhile, standard activation steering techniques (Subramani et al., 2022; Turner et al., 2023; Rimsky et al., 2024; Zou et al., 2023) completely fail to address token-level representational failures. Furthermore, the null results of fixed linear steering on architectures like Qwen2 (Liu, 2026) have been attributed generally to representational entanglement, without isolating specific architectural bottlenecks.

Our findings and improvements address these gaps directly. We introduce ZG-STR, which utilizes Fisher information for sorting tokens by how much the model's loss depends on them, thereby identifying fragile tokens. To resolve the limitations of existing repair methods, ZG-STR detects and repairs fragile tokens with zero gradient computation and no extra parameters. It applies the same mathematical principle as standard steering–a difference of activation centroids–specifically to repair the noisy hidden states caused by fragile tokens. Furthermore, our analysis goes beyond general entanglement to identify two specific architectural features behind the steering null results: QKV bias (Vaswani et al., 2017) and QK-Norm (Henry et al., 2020). We demonstrate that QKV bias dilutes the steering signal with a constant offset, while QK-Norm erases the steering contribution through variance normalization before it reaches attention.

3 Method

ZG-STR functions in four stages: (1) Fisher-based detection of fragile tokens, (2) discovery of healthy proxy tokens, (3) computation of steering vectors, and (4) inference-time injection at the optimal layer. Figure 1 gives an overview of the pipeline.

3.1 Fragile Token Detection

Given a calibration corpus $\mathcal{C}$ (1000 sentences from C4 (Raffel et al., 2020)) we compute the diagonal empirical Fisher information for each token embedding. For each sentence, we perform a forward pass with the standard language modeling cross-entropy loss. The gradient with respect to the embedding $\mathbf{e}_i$ of token i is $\nabla_{\mathbf{e}_i}\mathcal{L}$ and the diagonal Fisher score is:

$$F_{ii} = \frac{1}{|\mathcal{C}|}\sum_{x\in\mathcal{C}}(\nabla_{\mathbf{e}_i}\mathcal{L}(x))^2 \quad (1)$$

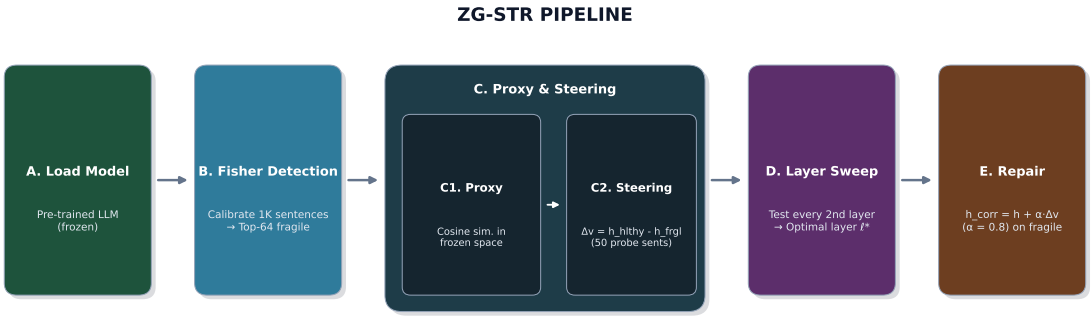

Figure 1: Overview of the ZG-STR pipeline. (A) Model loading, (B) Fisher-based fragile token detection, (C1) healthy proxy discovery, (C2) steering vector computation, (D) optimal layer selection via sweep, (E) inference-time steering injection.

In practice, a smaller Fisher value means that the loss function is relatively insensitive to changes in the input related to that token embedding, and that the model does not have much certainty about that token's embedding. All the tokens in our vocabulary are sorted according to F_{ii} (in ascending order), and the top 64 of those are selected as the most fragile tokens. Calculating the Fisher requires one backpropagation per sentence, where we backpropagate through the embeddings. Parameter updates are not done.

3.2 Healthy Proxy Discovery

For each fragile token t, we find its nearest well-trained neighbor in the embeddings. Let $\mathbf{W} \in \mathbb{R}^{V \times d}$ be the embedding matrix. We compute cosine similarity between the fragile token's embedding $\mathbf{W}_t$ and every other token:

$$s(t, j) = \frac{\mathbf{W}_t \cdot \mathbf{W}_j}{\|\mathbf{W}_t\| \|\mathbf{W}_j\|} \tag{2}$$

We ignored special tokens and all other fragile tokens, then selected the token with the highest similarity as the healthy proxy p.

3.3 Steering Vector Construction

For recovering the fragile tokens, the hidden representations of the healthy proxies in different layers are required. 50 probe sentences are created, which have no meaning but contain the placeholder `{proxy}` in them (for example, "The word `{proxy}` occurred in the text"). For every token t, its proxy p is placed inside each probe sentence. The model processes every sentence to get the residual stream hidden representation at different candidate layers ℓ for the place where p has been put. We calculate the centroid of the hidden representations of the proxy:

$$\mathbf{c}_\ell = \frac{1}{K} \sum_{k=1}^{K} \mathbf{h}_{\ell,k}^{\text{proxy}} \tag{3}$$

The steering vector for layer ℓ is then:

$$\Delta \mathbf{v}_\ell = \mathbf{c}_\ell - \mathbf{h}_\ell^{\text{fragile}} \tag{4}$$

where $\mathbf{h}_\ell^{\text{fragile}}$ is the hidden state of the fragile token at layer ℓ.

3.4 Optimal Layer Selection

Application of ZG-STR can happen in any layer, although its effectiveness may differ from one to another. We conduct a search through all layers spaced every k steps ($k = 2$ by default) in the transformer model. For every candidate layer ℓ, we construct 50 different prompts for testing by inserting a unique fragile token at its position. We calculate the loss only for fragile positions with and without the steering step. The chosen layer is the one with minimal difference of losses:

$$L^* = \arg\min_\ell \left(\mathcal{L}_{\text{steered}}^\ell - \mathcal{L}_{\text{baseline}} \right) \tag{5}$$

3.5 Inference-Time Application

During the inference process, once the presence of a fragile token t has been identified, a forward hook in the optimal layer ℓ_* captures the hidden state $\mathbf{h}$ of the residual stream and adds the calculated steering vector to it. The new hidden state value is computed as follows:

$$\mathbf{h}' = \mathbf{h} + \alpha \cdot \Delta \mathbf{v}_{\ell*} \tag{6}$$

Replacing the formula for the steering vector $\Delta \mathbf{v} = \mathbf{c} - \mathbf{h}$ in the correction formula results

Model	Size	Architecture	Lab
Qwen3.5-0.8B	0.8B	Dense	Alibaba
Qwen3.5-4B	4B	Dense	Alibaba
OLMo-3-7B-Inst.	7B	Dense	Allen AI
Llama-3.1-8B-Inst.	8B	Dense	Meta
IBM Granite-4.1-8B	8B	Dense	IBM
Falcon3-10B-Inst.	10B	Dense	TII
Phi-4 (14B)	14B	Dense	Microsoft
GPT-OSS-20B	20B	MoE	OpenAI
Qwen2.5-7B-Inst.	7B	Dense (QKV bias)	Alibaba
Qwen3-14B	14B	Dense (QK-Norm)	Alibaba

Table 1: Representative assessed models. The full list of 17 models appears in Appendix A.

in $\mathbf{h}' = (1-\alpha)\mathbf{h} + \alpha\mathbf{c}$. With the value of $\alpha = 0.8$, the result is $\mathbf{h}' = 0.2\mathbf{h} + 0.8\mathbf{c}$, or an 80/20 interpolation between the centroid derived from a healthy proxy token and the original activation. This approach exhibits two key advantages. Firstly, by keeping 20% of the original activations, the method retains positional and syntactic information particular to the prompt. Secondly, since the interpolation keeps 80% of the proxy centroid, the semantic information that was missing in the original activation will now be added to the output. Previous work in activation steering suggests moderate scaling factors within the range 0.5–1.0 (Rimsky et al., 2024; Zou et al., 2023). We chose $\alpha = 0.8$ because of its clear interpolation meaning. The same α was applied uniformly across all models.

4 Experimental Setup

Models. We benchmark ZG-STR on heterogeneous decoder-only transformer language models spanning from 0.8B to 20B parameters. Table 1 shows the representative models discussed in the main text; a complete list of all 17 models is provided in Appendix A. All models are benchmarked in bfloat16 on a single NVIDIA RTX PRO 6000 GPU.

Calibration. 1,000 sentences are arbitrarily chosen from the C4 corpus (Raffel et al., 2020) to detect fragile tokens. The same calibration set is used for all models.

Probe/Test sentences. 50 semantically neutral probe templates (e.g., "The word {proxy} appeared in the text.") filled with a healthy proxy token for each fragile token are used for steering vector computation. Direct benchmark uses 50 heterogeneous test templates (e.g., "What is the meaning of {}?"). All templates are provided in Appendix B.

Benchmark. We create a four-choice question asking "What is the meaning of {fragile}?" where the correct answer is the proxy token. For the other three slots, we used distractors that are randomly sampled words that are neither the fragile token nor the proxy. The number of questions varies per model (15–40) depending on the number of fragile tokens that qualify as word-like and have valid proxies.

Assessment metrics. We report a total of five metrics. (1) Δ_{loss}: the change in cross-entropy loss computed only at positions immediately after fragile tokens. (2) Improved: the fraction of test prompts where steering reduces the fragile-position loss. (3) Recovery: the fraction of fragile-token positions where the model's next-token prediction flips from incorrect (without steering) to correct (with steering). (4) Custom benchmark accuracy: the fraction of multiple-choice questions answered correctly. (5) Consistency KL: the KL divergence between the output logit distributions produced by the fragile token and its healthy proxy for the prompt "Define the word {word}."

Steering details. The steering scale is set to α=0.8 for all models based on the interpretation interpolation described in Section 3.5. The optimal injection layer is selected via the layer sweep described in Section 3.4, testing every second layer.

Hardware. The training of all models was carried out using only one NVIDIA RTX PRO 6000 GPU (96 GB VRAM) and a 12th-generation Intel Core i7-12700KF CPU with 64 GB of RAM. The total computational cost of all 17 models did not exceed 25 GPU-hours, which included the download of models, Fisher calibration, extraction of hidden states of probes, and sweeping of layers. On average, it took about 1 hour and a half hours to run one model, such as Granite-4.1-8B, and less than an hour to train smaller models.

5 Results

ZG-STR is assessed against three criteria: improvement in losses directly for fragile tokens, performance on the custom task-specific fragility comprehension test set, and robustness, measured using KL divergence. The main findings are shown in Tables 2–4. More information is given in Appendix C.

Direct metrics. We report the difference in

Model	Δ_{loss}	Impr.%	Rec.%
Qwen3.5-0.8B	−0.025	4.00	0.47
Qwen3.5-4B	−0.030	3.55	0.59
OLMo-3-7B	**−0.039**	2.81	0.19
Llama-3.1-8B	+0.009	4.82	0.14
IBM Granite-4.1-8B	−0.002	**6.87**	**0.72**
Falcon3-10B	−0.009	1.60	0.00
Phi-4 (14B)	+0.004	3.79	0.07
GPT-OSS-20B	−0.008	2.98	0.58
Qwen2.5-7B (QKV)	0.000	0.00	0.00
Qwen3-14B (QK-N)	0.000	0.00	0.00

Table 2: Direct metrics (Δ_{loss}, Improved%, Recovery%) for representative models. Best Δ_{loss} in bold.

Model	Base	ZG-STR	Δ
Llama-3.1-8B	72.5	85.0	**+12.5**
IBM Granite-4.1-8B	79.4	88.2	+8.8
Falcon3-10B	84.6	88.5	+3.8
GPT-OSS-20B	66.7	70.4	+3.7
Gemma-4-E4B	72.2	75.0	+2.8
Qwen2.5-7B (QKV)	60.9	60.9	0.0
Qwen3-14B (QK-N)	77.3	77.3	0.0

Table 3: Custom benchmark accuracy (%). Five models show improvement; QKV-bias and QK-Norm models show no change.

fragile position loss as computed after steering on the layer from which we are allowed to change activations with respect to the raw losses (Δ_{loss} column of Table 2). ZG-STR obtains lower loss on 10 out of the 17 models, with $\Delta_{\text{loss}} \in [-0.039 \text{ (OLMo-3-7B)}; 0.000 \text{ (Qwen3-14B)}]$. Rate of improved test prompts is 0.0%–6.9% (IBM Granite-4.1-8B). We find token-level recovery rates to be fairly low (0.1%–0.7%), supporting that loss reduction mostly distills into better local probability calibration, not next-token predictions flipped.

Custom benchmark accuracy. Table 3 reports accuracy on the fragile-token comprehension benchmark, which directly tests whether the model understands the meaning of fragile tokens. ZG-STR improves accuracy on five models: Llama-3.1-8B (+12.5%), IBM Granite-4.1-8B (+8.8%), Falcon3-10B (+3.8%), GPT-OSS-20B (+3.7%), and Gemma-4-E4B (+2.8%). In contrast, models with QKV bias (Qwen2.5-7B, $\Delta = 0.000$) and QK-Norm (Qwen3-14B, $\Delta = 0.000$) show no change, confirming the architectural analysis in Section 2.

Layer sensitivity. Figure 2 shows the layer-wise Δ_{loss} for Olmo-3-7B, representative of the U-shaped pattern observed across all bias-free dense

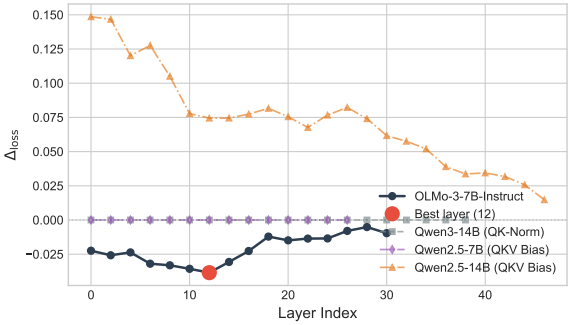

Figure 2: Layer sensitivity of ZG-STR. Bias-free dense models (OLMo-3-7B shown) exhibit a U-shaped curve with an optimal mid-network injection point. Architectures with QK-Norm (Qwen3-14B) and 7B QKV bias (Qwen2.5-7B) show a flatline of no effect ($\Delta_{\text{loss}} = 0$). At 14B scale, QKV bias (Qwen2.5-14B) results in a positive loss delta curve ($\Delta_{\text{loss}} > 0$), where steering increases the loss (harmful effect).

models. The optimal injection point lies at layer 12, in the middle of the network, with earlier and later layers producing smaller effects. This U-shape is consistent with prior contemporary findings that mid-network layers provide the most effective steering intervention (Rimsky et al., 2024). Architectures with QKV bias show a lack of improvement: while Qwen2.5-7B shows a flatline of $\Delta = 0.000$ at every layer where the constant bias term in the QKV projections dilutes the steering signal uniformly, regardless of injection depth, Qwen2.5-14B shows a positive Δ_{loss} across all layers, indicating that steering is actually harmful rather than producing a flatline. Architectures with QK-Norm (e.g., Qwen3-14B) also show $\Delta = 0.000$ across all layers, because variance normalization erases the steering contribution before it reaches attention. The qualitative difference between the U-shaped curves of bias-free models and the flatlines or positive loss deltas of QKV-bias and QK-Norm architectures provides evidence that these specific features determine whether ZG-STR succeeds or not. Full U-curves for all models are provided in Appendix C.

6 Analysis

Why ZG-STR fails on certain architectures:

The results in Section 5 uncover a clear separation that ZG-STR improves accuracy on bias-free models, but produces zero or near-zero impact on models with QKV bias or QK-Norm. This is not a matter of scale or training distribution because Qwen3.5-0.8B (0.8B parameters,

Model	KL_{base}	KL_{ZG}	Δ_{KL}
Qwen3.5-0.8B	0.100	0.095	−0.005
Qwen3.5-4B	0.050	0.041	−0.009
OLMo-3-7B	0.867	0.863	−0.004
Llama-3.1-8B	0.154	0.147	−0.007
IBM Granite-4.1-8B	0.325	0.221	**−0.104**
Falcon3-10B	0.157	0.154	−0.003
Phi-4 (14B)	0.199	0.196	−0.002
GPT-OSS-20B	0.176	0.144	−0.032
Qwen2.5-7B (QKV)	2.455	2.421	−0.034
Qwen3-14B (QK-N)	0.381	0.359	−0.023

Table 4: Consistency KL divergence for representative models. Lower KL indicates the fragile token's output distribution is closer to its healthy proxy. Best Δ_{KL} in bold.

same training regime as Qwen3-14B) shows meaningful Δ_{loss}, while Qwen3-14B shows none and Qwen2.5-7B/14B fail to improve. The difference is mainly architectural.

QKV bias adds a learned constant offset to the query, key, and value projections in the attention mechanism (Vaswani et al., 2017):

$$\begin{aligned} \mathbf{Q} &= \mathbf{W}_Q\mathbf{h} + \mathbf{b}_Q, \\ \mathbf{K} &= \mathbf{W}_K\mathbf{h} + \mathbf{b}_K, \\ \mathbf{V} &= \mathbf{W}_V\mathbf{h} + \mathbf{b}_V. \end{aligned} \tag{7}$$

When a steering vector $\Delta\mathbf{v}$ is added to the residual stream, the query becomes $\mathbf{Q}' = (\mathbf{W}_Q\mathbf{h} + \mathbf{b}_Q) + \mathbf{W}_Q\Delta\mathbf{v}$. The bias term $\mathbf{b}_Q$ is independent of the input. At 7B scale, this bias term is sufficient to dilute the steering contribution $\mathbf{W}_Q\Delta\mathbf{v}$ and nullify steering entirely ($\Delta = 0.000$ at every layer for Qwen2.5-7B). However, at 14B scale, the magnitude of the bias grows large enough to actively misdirect the steering signal, leading to the positive $\Delta_{loss} > 0$ observed across all layers for Qwen2.5-14B, where steering is actually harmful rather than producing a simple flatline.

QK-Norm (Henry et al., 2020) applies RMSNorm to queries and keys before the attention dot product: $\mathbf{Q}_{norm} = \text{RMSNorm}(\mathbf{W}_Q\mathbf{h})$. RMSNorm rescales its input to unit variance, and because $\text{RMSNorm}(\mathbf{h} + \Delta\mathbf{v}) \neq \text{RMSNorm}(\mathbf{h}) + \text{RMSNorm}(\Delta\mathbf{v})$, the steering contribution is erased before reaching attention. Qwen3-14B shows $\Delta = 0.000$ across all layers, confirming this mechanism.

These findings are aligned with Cheng et al. (2026), who showed that steering vectors act basically through the output-value (OV) circuit while largely ignoring the query-key (QK) circuit. Our results provide the practical confirmation of this theory, namely *"Architectural features that target the QK pathway (QKV bias, QK-Norm) are precisely what prevent steering from working"*. However, models without these features show measurable improvement, regardless of training distribution.

Scaling behavior:

ZG-STR's effectiveness is not monotonic with model size. Small models (0.8B–7B) Qwen3.5 family (Qwen Team, 2026) and Allen AI (Allen Institute for AI, 2025) show consistent loss reduction ($\Delta_{loss} = -0.020$ to -0.039) but no accuracy gain on the custom benchmark. On the other hand, the internal repair is measurable, but not strong enough to flip downstream answers.

At 8–10B, the repair crosses a threshold: Llama-3.1-8B (Meta AI, 2024) (+12.5%), IBM Granite-4.1-8B (IBM Research, 2025) (+8.8%), and Falcon3-10B (Technology Innovation Institute, 2025) (+3.8%) all show behavioral improvement. The 8–10B range appears to be a *sweet spot* where the residual stream is well enough for steering to be successful, yet not so large that the steering signal is hurt. At 14B+, the effect diminishes: Phi-4-14B (Microsoft Research, 2024) shows no accuracy gain, and GPT-OSS-20B (OpenAI, 2025) shows a modest +3.7%. *We hypothesize that large-scale models have more redundant representations, requiring stronger steering signals or different injection strategies.*

Consistency. Consistency, measured by the KL divergence of fragile and proxy output distributions, improves on 15 out of 17 models, even those without accuracy gains. This indicates that steering produces a shift toward the proxy, regardless of whether that shift is large enough to change the final answer.

7 Conclusion

ZG-STR allows the recovery of the fragile token's representation using one zero-gradient steering vector, overcoming an issue that had previously been solved only through expensive fine-tuning. ZG-STR results in up to a 12.5% increase in fragile token understanding accuracy across five families of pre-trained models representing various architectures. In all cases, there is no trade-off in general world knowledge and no need for parameter tuning. The incompetence of ZG-STR on architectures with QKV bias or QK-Norm, aligned

with recent discoveries regarding the mechanism of representation steering (Cheng et al., 2026), allows us to determine the exact conditions for successful token repairs at inference time.

8 Limitations

Fisher calibration requires one backward pass through the embedding layer, which is not applicable for APIonly models. Developing a pure forwardpass fragility detection technique would extend ZGSTR's applicability.

Our assessment was on a total of 17 models with sizes from 0.8B to 20B. We did not assess models above 20B due to the memory overhead of Fisher matrix computation on our 96 GB GPU. In our preliminary experiments, INT8 quantization using bitsandbytes on a 27B model showed muted effects, likely due to quantization noise; largerscale bf16 assessment remains future work.

ZGSTR is ineffective on models with QKV bias (e.g., Qwen2.57B/14B) or QKNorm (e.g., Qwen314B), as these architectural features erase the steering signal. The method currently requires biasfree dense or MoE architectures.

The number of multiplechoice questions per model varies from 15 to 40, as it depends on how many fragile tokens qualify as wordlike and have valid healthy proxies. Some models with very high baseline accuracy (e.g., Gemma312B at 90%) leave limited space for measurable improvement.

We did not assess code generation, mathematical reasoning, or retrievalaugmented generation. Crosslingual and multidomain assessment is left to future work.

We used a fixed α=0.8 for all models. While this is justified by the 80/20 interpolation interpretation and prior work, permodel or perlayer tuning might yield stronger results.

References

Allen Institute for AI. 2025. Olmo 3: Technical report. *Preprint*, arXiv:2512.13961.

Stephen Cheng, Sarah Wiegreffe, and Dinesh Manocha. 2026. What drives representation steering? a mechanistic case study on steering refusal. *arXiv preprint arXiv:2604.09876*.

Yibo Fan, Jingru Li, and Huan Li. 2026. Glitchcleaner: Lightweight glitch tokens repairing by lossless gated lora in large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. 2017. On calibration of modern neural networks. In *Proceedings of the 34th International Conference on Machine Learning*, pages 1321–1330.

Alex Henry, Prudhvi Raj Dachapally, Shubham Shantaram Pawar, and Yuxuan Chen. 2020. Query-key normalization for transformers. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4246–4253. Association for Computational Linguistics.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

IBM Research. 2025. Granite 4.1: Ibm's open-source language models. https://huggingface.co/blog/ibm-granite/granite-4-1. Online Hugging Face announcement.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, and 1 others. 2017. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526.

Sander Land and Max Bartolo. 2024. Fishing for magikarp: Automatically detecting under-trained tokens in large language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 11631–11646, Miami, Florida, USA.

Ming Liu. 2026. Decodable but not corrected by fixed residual-stream linear steering: Evidence from medical llm failure regimes. *arXiv preprint arXiv:2605.05715*.

Meta AI. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Microsoft Research. 2024. Phi-4 technical report. *Preprint*, arXiv:2412.08905.

OpenAI. 2025. gpt-oss-120b & gpt-oss-20b model card. *Preprint*, arXiv:2508.10925.

Qwen Team. 2024. Qwen2.5 technical report. *Preprint*, arXiv:2412.15115.

Qwen Team. 2026. Qwen3.5-omni technical report. *Preprint*, arXiv:2604.15804.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Nina Rimsky, Nick Gabrieli, Julian Schulz, Meg Tong, Evan Hubinger, and Alexander Turner. 2024. Steering llama 2 via contrastive activation addition. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15504–15522, Bangkok, Thailand.

Jessica Rumbelow and Matthew Watkins. 2023. SolidGoldMagikarp (plus, prompt generation). https://www.alignmentforum.org/posts/aPeJE8bSo6rAFoLqg/solidgoldmagikarp-plus-prompt-generation.

Sidak Pal Singh and Dan Alistarh. 2020. WoodFisher: Efficient second-order approximation for neural network compression. In *Advances in Neural Information Processing Systems*, volume 33, pages 18098–18109.

Nishant Subramani, Nivedita Suresh, and Matthew Peters. 2022. Extracting latent steering vectors from pretrained language models. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 566–581, Dublin, Ireland.

Technology Innovation Institute. 2025. Falcon3: A new family of language models. https://huggingface.co/blog/falcon3. Online Hugging Face announcement.

Lucas Theis, Iryna Korshunova, Alykhan Tejani, and Ferenc Huszár. 2018. Faster gaze prediction with dense networks and Fisher pruning. *arXiv preprint arXiv:1801.05787*.

Alexander Matt Turner, Lisa Thiergart, David Udell, Gavin Leech, Ulisse Mini, and Monte MacDiarmid. 2023. Steering language models with activation engineering. *Preprint*, arXiv:2308.10213.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, and 1 others. 2024. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*.

Andy Zou, Long Phan, Sarah Chen, James Campbell, Phillip Guo, Richard Ren, Alexander Pan, Xuwang Yin, Mantas Mazeika, Ann-Kathrin Dombrowski, and 1 others. 2023. Representation engineering: A top-down approach to ai transparency. *Preprint*, arXiv:2310.01405.

Appendix

A Full Model List

B Probe and Test Templates

B.1 Probe Sentences (50)

The following semantically neutral sentences are used for steering vector computation. Each contains the placeholder `{proxy}`.

1. "{proxy}" is a word.
2. I saw "{proxy}" in the document.
3. The meaning of "{proxy}" is clear.
4. Please define "{proxy}".
5. The term "{proxy}" is used here.
6. I don't understand "{proxy}".
7. "{proxy}" appears in the list.
8. Explain "{proxy}" in detail.
9. The concept of "{proxy}" is interesting.
10. We discussed "{proxy}" yesterday.
11. You should know "{proxy}".
12. "{proxy}" has many uses.
13. I like the word "{proxy}".
14. The definition of "{proxy}" is simple.
15. She mentioned "{proxy}" in her speech.
16. The teacher taught "{proxy}".
17. Find the word "{proxy}" in the text.
18. "{proxy}" is a noun.
19. It is important to understand "{proxy}".
20. The example shows "{proxy}".
21. I wrote a sentence with "{proxy}".
22. "{proxy}" can be a subject.
23. The book includes "{proxy}".
24. We learned about "{proxy}" today.
25. The spelling of "{proxy}" is easy.
26. Say "{proxy}" out loud.
27. The article uses "{proxy}" incorrectly.
28. I need to research "{proxy}".
29. "{proxy}" is not a verb.
30. The dictionary defines "{proxy}".
31. He asked about "{proxy}".
32. The question involved "{proxy}".
33. I cannot recall "{proxy}".
34. "{proxy}" is in the glossary.
35. The lecture focused on "{proxy}".
36. Write down the word "{proxy}".
37. The paragraph contains "{proxy}".
38. I think "{proxy}" is a common term.
39. The discussion centered on "{proxy}".
40. "{proxy}" is an example.
41. The student defined "{proxy}".
42. Look up the meaning of "{proxy}".
43. The list includes "{proxy}" twice.
44. I remember learning "{proxy}".
45. "{proxy}" is a placeholder here.
46. The phrase "{proxy}" is unfamiliar.
47. Repeat the word "{proxy}".
48. The assignment uses "{proxy}".
49. I am studying "{proxy}".
50. The word "{proxy}" is important.

B.2 Test Templates (50)

The following heterogeneous test templates are used for benchmark evaluation. Each contains the placeholder `{}`.

1. What is the definition of {}?
2. Explain {} in your own words.
3. Give an example of {} in a sentence.
4. How is {} used in everyday conversation?
5. Translate {} into Spanish.
6. What is the antonym of {}?
7. Describe the origin of {}.
8. Is {} a concrete or abstract noun?
9. What field of study is {} most associated with?
10. If {} is a verb, what is its past tense?
11. Compare {} with its synonym.
12. What is the opposite of {}?
13. Write a short poem that includes {}.
14. What does {} symbolize in literature?
15. How would you teach {} to a child?

Model Name	HuggingFace ID	Size	Architecture	Lab	License	Prec.
Qwen2.5-14B	Qwen/Qwen2.5-14B	14B	Dense (QKV bias)	Alibaba	Apache-2.0	bf16
Qwen2.5-7B-Instruct	Qwen/Qwen2.5-7B-Instruct	7B	Dense (QKV bias)	Alibaba	Apache-2.0	bf16
Qwen3-14B	Qwen/Qwen3-14B	14B	Dense (QK-Norm)	Alibaba	Apache-2.0	bf16
Qwen3.5-0.8B	Qwen/Qwen3.5-0.8B	0.8B	Dense	Alibaba	Apache-2.0	bf16
Qwen3.5-4B	Qwen/Qwen3.5-4B	4B	Dense	Alibaba	Apache-2.0	bf16
Qwen3.6-27B	Qwen/Qwen3.6-27B	27B	Dense	Alibaba	Apache-2.0	int8
OLMo-3-7B-Instruct	allenai/Olmo-3-7B-Instruct	7B	Dense	Allen AI	Apache-2.0	bf16
Gemma-3-12B-IT	google/gemma-3-12b-it	12B	Dense	Google	Gemma	bf16
Gemma-4-E2B	google/gemma-4-E2B	2B	Dense	Google	Gemma	bf16
Gemma-4-E4B	google/gemma-4-E4B	4B	Dense	Google	Gemma	bf16
IBM Granite-4.1-8B	ibm-granite/granite-4.1-8b	8B	Dense	IBM	Apache-2.0	bf16
Llama-3.1-8B-Inst.	meta-llama/Llama-3.1-8B-Instruct	8B	Dense	Meta	Llama-3.1	bf16
Phi-4	microsoft/Phi-4	14B	Dense	Microsoft	MIT	bf16
Mistral-7B-Inst.	mistralai/Mistral-7B-Instruct-v0.3	7B	Dense	Mistral AI	Apache-2.0	bf16
GPT-OSS-20B	openai/gpt-oss-20b	20B	MoE	OpenAI	Apache-2.0	bf16
StableLM-2-12B	stabilityai/stablelm-2-12b-chat	12B	Dense	Stability AI	StableLM	bf16
Falcon3-10B-Inst.	tiiuae/Falcon3-10B-Instruct	10B	Dense	TII	Apache-2.0	bf16

Table 5: Full list of all 17 evaluated models with HuggingFace identifiers, sizes, architectures, originating labs, licenses, and precision used.

16. Use {} in a command.
17. What is the etymology of {}?
18. Can {} be used as both a noun and a verb? If so, give examples.
19. Describe a scenario where {} is appropriate.
20. What is the plural form of {}?
21. What are the key characteristics of {}?
22. How does {} relate to emotion?
23. What is a common misconception about {}?
24. Give three synonyms for {}.
25. What is the most common usage of {}?
26. Define {} technically.
27. What is the cultural significance of {}?
28. What does {} imply?
29. How does {} differ from a similar concept?
30. Translate {} into French.
31. What is the grammatical role of {} in a sentence?
32. If {} is an adjective, what noun does it modify?
33. Write a definition of {} for a dictionary.
34. Give a real-world example of {}.
35. What is the historical context of {}?
36. How is {} used in academic writing?
37. What is the primary function of {}?
38. Describe {} in one word.
39. What is the negation of {}?
40. What is a typical collocation with {}?
41. Does {} have a positive or negative connotation?
42. What is the metaphorical meaning of {}?
43. How would you explain {} to someone learning English?
44. What is the root word of {}?
45. Give an instruction that uses {}.
46. What does {} contrast with?
47. Write a haiku that features {}.
48. What is the binary opposite of {}?
49. What is the most important thing to know about {}?
50. How is {} used in a formal setting?

C Per-Model Detailed Results

C.1 Custom Benchmark and Consistency Details

Layer	Δ_{loss}	Impr.%
0	+0.149	0.05
2	+0.147	0.05
4	+0.120	0.00
6	+0.128	0.05
8	+0.105	0.05
10	+0.078	0.05
12	+0.074	0.00
14	+0.075	0.05
16	+0.077	0.05
18	+0.082	0.05
20	+0.075	0.05
22	+0.068	0.05
24	+0.077	0.05
26	+0.082	0.05
28	+0.074	0.05
30	+0.062	0.05
32	+0.058	0.05
34	+0.052	0.05
36	+0.039	0.05
38	+0.034	0.05
40	+0.034	0.05
42	+0.032	0.05
44	+0.026	0.19
46	**+0.015**	**0.23**

Table 6: Qwen2.5-14B: per-layer sweep. Best layer in bold.

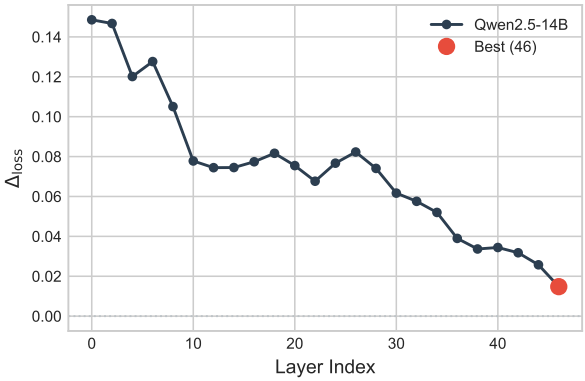

Figure 3: Qwen2.5-14B: layer sensitivity curve.

Layer	Δ_{loss}	Impr.%
0	**+0.000**	**0.00**
2	+0.000	0.00
4	+0.000	0.00
6	+0.000	0.00
8	+0.000	0.00
10	+0.000	0.00
12	+0.000	0.00
14	+0.000	0.00
16	+0.000	0.00
18	+0.000	0.00
20	+0.000	0.00
22	+0.000	0.00
24	+0.000	0.00
26	+0.000	0.00

Table 7: Qwen2.5-7B-Instruct: per-layer sweep. Best layer in bold.

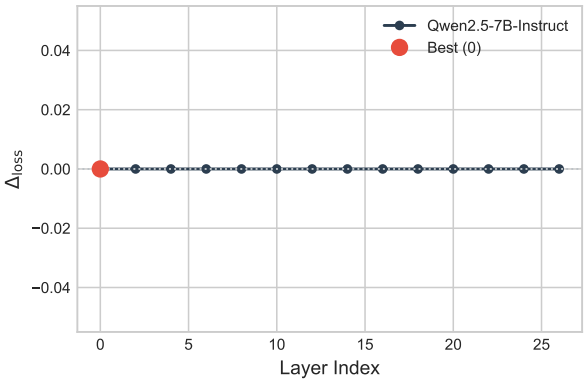

Figure 4: Qwen2.5-7B-Instruct: layer sensitivity curve.

Layer	Δ_{loss}	Impr.%
0	**+0.000**	**0.00**
2	+0.000	0.00
4	+0.000	0.00
6	+0.000	0.00
8	+0.000	0.00
10	+0.000	0.00
12	+0.000	0.00
14	+0.000	0.00
16	+0.000	0.00
18	+0.000	0.00
20	+0.000	0.00
22	+0.000	0.00
24	+0.000	0.00
26	+0.000	0.00
28	+0.000	0.00
30	+0.000	0.00
32	+0.000	0.00
34	+0.000	0.00
36	+0.000	0.00
38	+0.000	0.00

Table 8: Qwen3-14B: per-layer sweep. Best layer in bold.

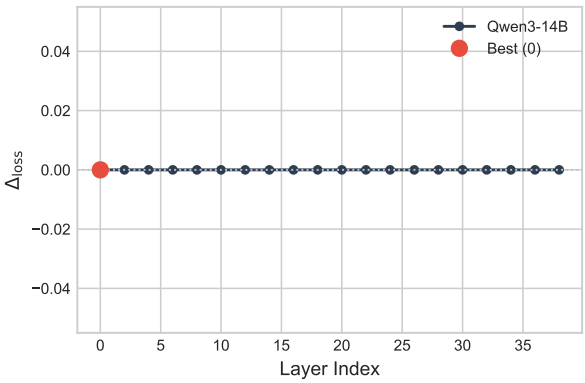

Figure 5: Qwen3-14B: layer sensitivity curve.

Layer	Δ_{loss}	Impr.%
0	**−0.025**	**4.00**
2	−0.012	3.07
4	−0.002	2.79
6	−0.001	2.51
8	+0.002	2.74
10	+0.005	2.47
12	+0.013	2.19
14	+0.016	2.14
16	+0.020	1.86
18	+0.019	1.49
20	+0.016	1.63
22	+0.018	1.53

Table 9: Qwen3.5-0.8B: per-layer sweep. Best layer in bold.

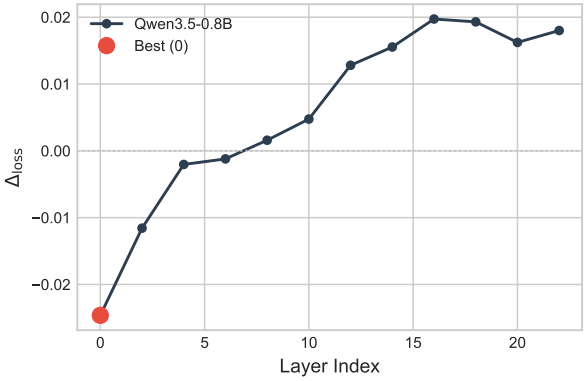

Figure 6: Qwen3.5-0.8B: layer sensitivity curve.

Layer	Δ_{loss}	Impr.%
0	**−0.030**	**3.55**
2	−0.025	3.64
4	−0.025	3.59
6	−0.027	3.86
8	−0.025	3.86
10	−0.021	3.59
12	−0.019	3.55
14	−0.019	3.50
16	−0.019	3.59
18	−0.005	2.14
20	−0.001	2.05
22	+0.002	1.68
24	+0.005	1.50
26	+0.007	1.50
28	+0.001	1.91
30	−0.006	2.27

Table 10: Qwen3.5-4B: per-layer sweep. Best layer in bold.

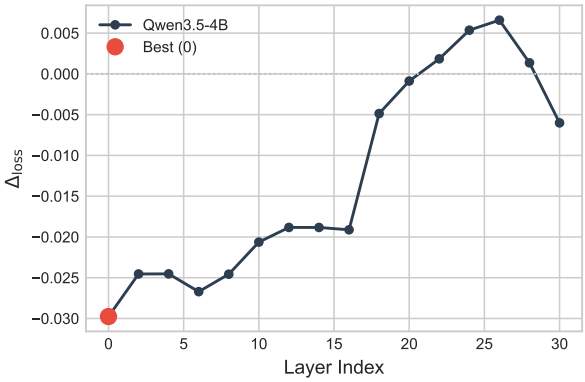

Figure 7: Qwen3.5-4B: layer sensitivity curve.

Layer	Δ_{loss}	Impr.%
0	+0.001	0.82
4	**−0.000**	**0.92**
8	+0.002	0.71
12	+0.000	1.02
16	+0.003	0.82
20	+0.002	0.82
24	+0.002	0.61
28	+0.002	0.51
32	+0.001	0.71
36	+0.002	0.41
40	+0.003	0.61
44	+0.003	0.82
48	+0.011	0.41
52	+0.015	0.51
56	+0.012	0.31
60	+0.013	0.20

Table 11: Qwen3.6-27B: per-layer sweep. Best layer in bold.

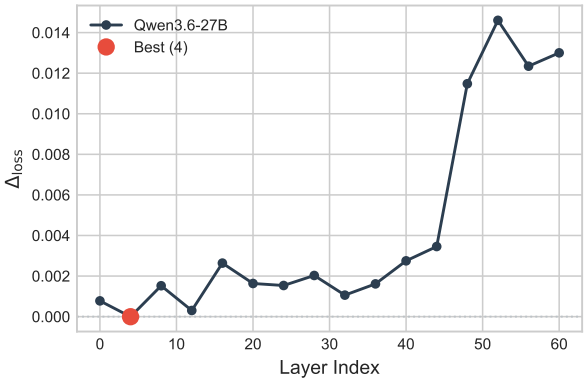

Figure 8: Qwen3.6-27B: layer sensitivity curve.

Layer	Δ_{loss}	Impr.%
0	−0.022	2.15
2	−0.026	2.26
4	−0.024	2.30
6	−0.032	2.56
8	−0.033	2.67
10	−0.036	2.93
12	**−0.039**	**2.81**
14	−0.031	2.37
16	−0.023	2.26
18	−0.012	1.85
20	−0.015	2.04
22	−0.014	1.74
24	−0.013	1.89
26	−0.008	1.52
28	−0.005	1.74
30	−0.010	2.00

Table 12: OLMo-3-7B-Instruct: per-layer sweep. Best layer in bold.

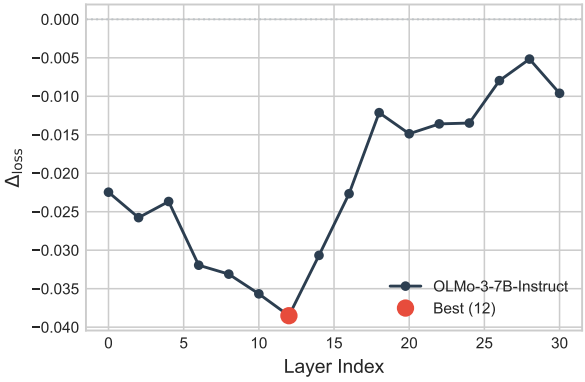

Figure 9: OLMo-3-7B-Instruct: layer sensitivity curve.

Layer	Δ_{loss}	Impr.%
0	+0.051	0.15
2	+0.053	0.18
4	+0.052	0.22
6	+0.050	0.15
8	+0.034	0.25
10	+0.033	0.25
12	+0.030	0.29
14	+0.031	0.25
16	+0.029	0.25
18	+0.030	0.22
20	+0.036	0.07
22	+0.039	0.04
24	+0.040	0.00
26	+0.036	0.00
28	+0.025	0.18
30	+0.016	0.29
32	+0.016	0.33
34	+0.014	0.33
36	+0.015	0.25
38	+0.018	0.18
40	+0.017	0.18
42	+0.015	0.18
44	+0.014	0.15
46	**+0.009**	**0.44**

Table 13: Gemma-3-12B-IT: per-layer sweep. Best layer in bold.

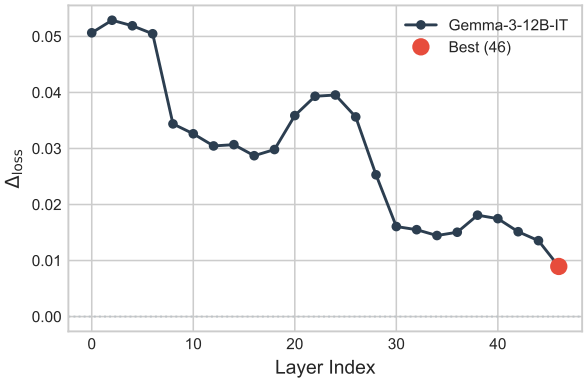

Figure 10: Gemma-3-12B-IT: layer sensitivity curve.

Layer	Δ_{loss}	Impr.%
0	+0.005	0.60
2	+0.004	0.75
4	+0.006	0.50
6	+0.006	0.40
8	+0.008	0.35
10	+0.008	0.50
12	+0.013	0.20
14	+0.011	0.10
16	+0.010	0.10
18	+0.013	0.00
20	+0.018	0.05
22	+0.013	0.10
24	+0.012	0.50
26	+0.012	0.55
28	+0.012	0.45
30	+0.010	0.55
32	+0.007	0.90
34	**−0.006**	**2.05**

Table 14: Gemma-4-E2B: per-layer sweep. Best layer in bold.

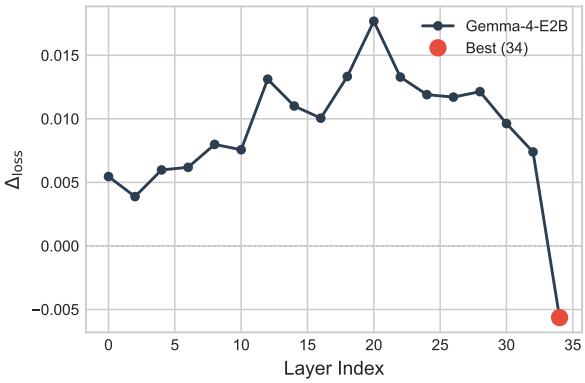

Figure 11: Gemma-4-E2B: layer sensitivity curve.

Layer	Δ_{loss}	Impr.%
0	+0.008	0.43
2	+0.008	0.51
4	+0.006	0.51
6	**+0.006**	**0.60**
8	+0.010	0.26
10	+0.008	0.26
12	+0.009	0.21
14	+0.009	0.17
16	+0.010	0.13
18	+0.013	0.09
20	+0.016	0.09
22	+0.015	0.17
24	+0.021	0.04
26	+0.023	0.13
28	+0.021	0.13
30	+0.022	0.13
32	+0.023	0.17
34	+0.022	0.26
36	+0.021	0.30
38	+0.021	0.34
40	+0.021	0.30

Table 15: Gemma-4-E4B: per-layer sweep. Best layer in bold.

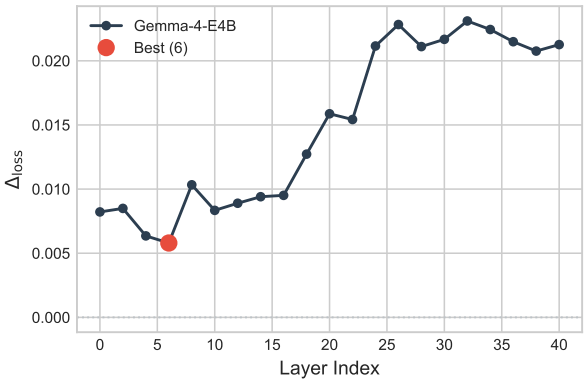

Figure 12: Gemma-4-E4B: layer sensitivity curve.

Layer	Δ_{loss}	Impr.%
0	**−0.002**	**6.87**
2	+0.006	6.25
4	+0.019	5.44
6	+0.024	4.87
8	+0.023	5.01
10	+0.019	4.96
12	+0.013	4.87
14	+0.012	4.77
16	+0.011	5.30
18	+0.016	5.30
20	+0.019	4.92
22	+0.030	4.30
24	+0.031	4.82
26	+0.040	4.30
28	+0.042	4.34
30	+0.052	4.11
32	+0.064	3.29
34	+0.066	3.29
36	+0.051	4.11
38	+0.019	6.63

Table 16: IBM Granite-4.1-8B: per-layer sweep. Best layer in bold.

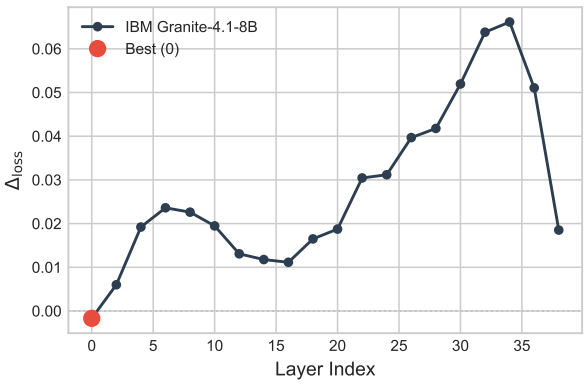

Figure 13: IBM Granite-4.1-8B: layer sensitivity curve.

Layer	Δ_{loss}	Impr.%
0	**+0.009**	**4.82**
2	+0.018	3.87
4	+0.010	4.04
6	+0.014	3.84
8	+0.024	3.19
10	+0.032	2.99
12	+0.038	2.78
14	+0.057	2.21
16	+0.065	2.41
18	+0.085	2.17
20	+0.094	2.04
22	+0.089	2.07
24	+0.073	2.41
26	+0.071	2.31
28	+0.060	2.78
30	+0.051	3.53

Table 17: Llama-3.1-8B-Instruct: per-layer sweep. Best layer in bold.

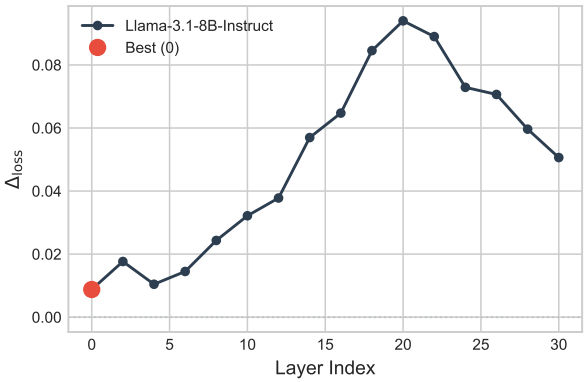

Figure 14: Llama-3.1-8B-Instruct: layer sensitivity curve.

Layer	Δ_{loss}	Impr.%
0	+0.038	2.67
2	+0.026	3.19
4	+0.020	3.40
6	+0.009	3.82
8	+0.009	3.68
10	+0.007	3.72
12	**+0.004**	**3.79**
14	+0.008	3.40
16	+0.010	3.37
18	+0.009	3.47
20	+0.017	2.95
22	+0.024	2.39
24	+0.029	2.32
26	+0.027	2.28
28	+0.018	2.63
30	+0.019	2.60
32	+0.010	3.33
34	+0.009	3.40
36	+0.008	3.37
38	+0.017	2.95

Table 18: Phi-4 (14B): per-layer sweep. Best layer in bold.

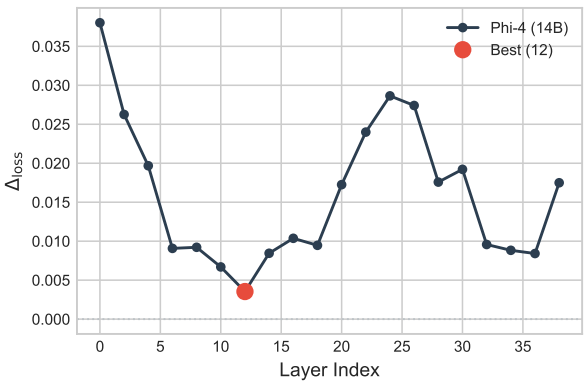

Figure 15: Phi-4 (14B): layer sensitivity curve.

Layer	Δ_{loss}	Impr.%
0	**−0.007**	**2.88**
2	+0.001	2.76
4	−0.003	2.96
6	−0.001	2.96
8	+0.004	2.76
10	+0.003	2.28
12	+0.002	2.60
14	+0.012	1.92
16	+0.012	1.88
18	+0.023	1.56
20	+0.034	0.88
22	+0.039	0.48
24	+0.032	0.56
26	+0.025	0.88
28	+0.026	0.80
30	+0.014	1.88

Table 19: Mistral-7B-Instruct: per-layer sweep. Best layer in bold.

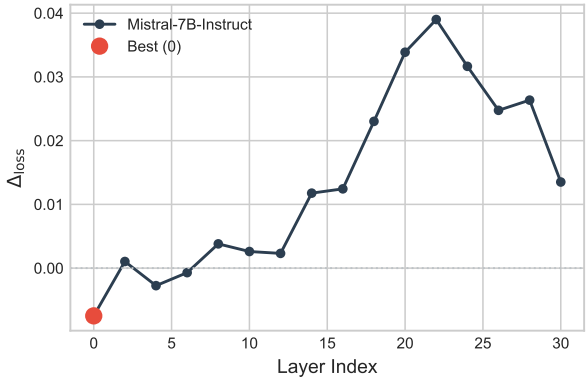

Figure 16: Mistral-7B-Instruct: layer sensitivity curve.

Layer	Δ_{loss}	Impr.%
0	−0.007	2.91
2	−0.008	2.87
4	**−0.008**	**2.98**
6	−0.006	2.76
8	−0.002	2.44
10	−0.003	2.87
12	−0.000	2.65
14	+0.008	2.11
16	+0.012	1.89
18	+0.016	1.93
20	+0.016	1.64
22	+0.017	1.67

Table 20: GPT-OSS-20B: per-layer sweep. Best layer in bold.

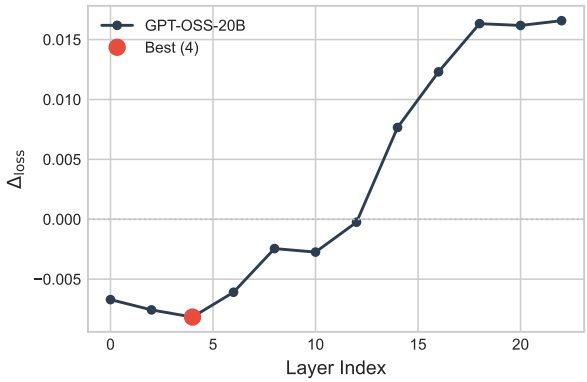

Figure 17: GPT-OSS-20B: layer sensitivity curve.

Layer	Δ_{loss}	Impr.%
0	+0.007	0.67
2	+0.010	0.63
4	+0.006	0.67
6	+0.007	0.88
8	−0.000	0.95
10	−0.000	0.88
12	+0.003	0.84
14	+0.004	0.84
16	−0.003	0.95
18	−0.001	0.91
20	−0.004	1.02
22	−0.007	0.95
24	−0.008	1.09
26	−0.012	1.16
28	−0.015	1.23
30	−0.019	1.33
32	−0.017	1.33
34	−0.015	1.30
36	−0.020	1.33
38	**−0.023**	**1.54**

Table 21: StableLM-2-12B: per-layer sweep. Best layer in bold.

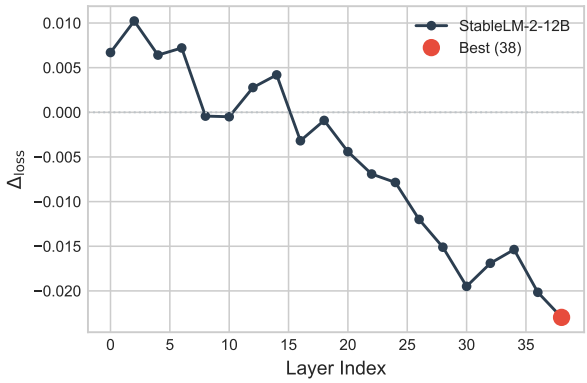

Figure 18: StableLM-2-12B: layer sensitivity curve.

Layer	Δ_{loss}	Impr.%
0	**−0.009**	**1.60**
2	+0.001	0.44
4	+0.001	0.24
6	−0.000	0.82
8	+0.000	0.44
10	−0.000	0.88
12	+0.001	0.58
14	+0.001	0.41
16	+0.000	0.58
18	+0.000	0.41
20	+0.000	0.48
22	+0.000	0.58
24	−0.000	0.65
26	+0.000	0.44
28	+0.000	0.24
30	+0.000	0.48
32	−0.000	0.54
34	−0.000	0.85
36	+0.000	0.17
38	+0.000	0.03

Table 22: Falcon3-10B-Instruct: per-layer sweep. Best layer in bold.

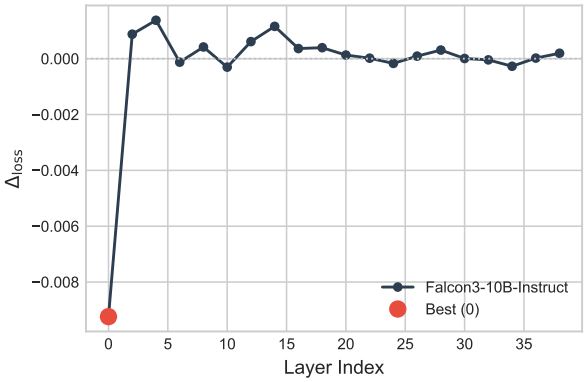

Figure 19: Falcon3-10B-Instruct: layer sensitivity curve.

Model	Base Acc	ZG Acc	ΔAcc	Qs	KL_{base}	KL_{ZG}	ΔKL
Qwen2.5-14B	43.5	43.5	0.0	23	2.102	2.097	−0.005
Qwen2.5-7B-Instruct	60.9	60.9	0.0	23	2.455	2.421	−0.034
Qwen3-14B	77.3	77.3	0.0	22	0.381	0.359	−0.023
Qwen3.5-0.8B	87.0	87.0	0.0	23	0.100	0.095	−0.005
Qwen3.5-4B	76.9	76.9	0.0	26	0.050	0.041	−0.009
Qwen3.6-27B	87.9	87.9	0.0	33	0.027	0.025	−0.002
OLMo-3-7B-Instruct	77.8	77.8	0.0	27	0.867	0.863	−0.004
Gemma-3-12B-IT	90.0	90.0	0.0	20	0.079	0.079	<0.001
Gemma-4-E2B	100.0	100.0	0.0	15	0.126	0.126	0.000
Gemma-4-E4B	72.2	75.0	+2.8	36	0.090	0.082	−0.008
IBM Granite-4.1-8B	79.4	88.2	+8.8	34	0.325	0.221	−0.104
Llama-3.1-8B-Inst.	72.5	85.0	+12.5	40	0.154	0.147	−0.007
Phi-4 (14B)	74.2	74.2	0.0	31	0.199	0.196	−0.002
Mistral-7B-Inst.	76.2	76.2	0.0	21	0.248	0.192	−0.056
GPT-OSS-20B	66.7	70.4	+3.7	27	0.176	0.144	−0.032
StableLM-2-12B	87.9	87.9	0.0	33	0.745	0.745	<0.001
Falcon3-10B-Inst.	84.6	88.5	+3.8	26	0.157	0.154	−0.003

Table 23: Custom benchmark accuracy (%) and consistency KL divergence for all 17 models. Qs = number of benchmark questions. Accuracy values rounded to one decimal place; KL values rounded to three decimal places.