How Fine-Tuning Objectives Shape Layer-Wise Information in LLM Hallucination Detection

Kaiyang Wan[♠♣] **Forrest Sheng Bao**[♣] **Amin Ahmad**[♣] **Yuxia Wang**[♠]
[♠]INSAIT, Sofia University "St. Kliment Ohridski", Sofia, Bulgaria
[♣]PAIR Systems, Inc., Mountain View, California
{kaiyang.wan, yuxia.wang}@insait.ai {forrest, amin.ahmad}@pairsys.ai

Abstract

Under matched conditions, classifier-head fine-tuning (CLS) paradigm consistently outperforms the LLM-as-Judge sequence-to-sequence (S2S) on binary hallucination detection. We trace this gap to the representations induced by the two objectives. Static analysis shows that CLS learns a task-specific discriminative direction, pushing hidden states into a clear two-cluster geometry with stronger class separation and linearly decodable label information. By contrast, S2S relies on the pretrained language-modeling head to make decisions. This minimizes the pressure to reorganize representations, leaving the hidden states much closer to the pretrained backbone. Our analysis also reveals that the CLS signal saturates before the final layer, after which upper layers form a plateau of statistically interchangeable and largely redundant representations. However, endpoint redundancy alone does not show whether these layers are dispensable during training. Further training-dynamics analysis reveals 1) a lower pretrained substrate, 2) a narrow upper-middle **InfoWindow** where task information emerges, and 3) a later-synchronizing upper plateau that mirrors this signal. This mechanism motivates us to remove layers above the InfoWindow. Truncating 31% of the network leaves accuracy unchanged within seed noise on Qwen3-4B and 8B and cuts latency by $1.4\times$. The same depth fraction, fixed before any analysis of the target models, transfers to Llama-3.2-3B and Mistral-7B-v0.3, and the layer structure behind it reappears there and on QNLI and SST-2. We hope this helps guide efficient task-specific LLM design.

1 Introduction

Hallucination detection is to identify whether a model produces unsupported content relative to its input or factually false claims contradicting world knowledge [Manakul et al., 2023, Wang et al., 2024b]. It is central to the built-in RAG in LLMs [Lewis et al., 2020, Gao et al.] and agentic systems [Madaan et al., 2023, Wan et al., 2025, 2026], where downstream modules and users demand reliable, efficient verification of model outputs [Belyi et al., 2025, Orgad et al.]. We study hallucination detectors fine-tuned using labeled data based on pretrained LLM backbones, comparing two paradigms: **1)** LLM-as-Judge sequence-to-sequence fine-tuning (S2S) generates a verbal verdict such as `Yes` or `No` [Gu et al., Zheng et al., 2023]; and **2)** Classifier-head fine-tuning (CLS) attaches a task-specific linear head for binary prediction [Houlsby et al., 2019, Tang et al., 2024].

At first sight, S2S appears to benefit from a stronger inductive prior. S2S preserves the pretrained LLM head and optimizes the same objective. It directly exploits representations shaped by large-scale next-token prediction [Radford et al., 2019, Brown et al., 2020]. CLS, in contrast, replaces the generative readout with a randomly initialized binary projection, introducing new parameters and an objective departing from LLM's pretraining objective [Wolf et al., 2020], which weakens the transfer of pretrained knowledge. This comparison would suggest that S2S should provide a stronger detector. Yet under matched fine-tuning conditions, CLS outperforms S2S on every backbone we test, and the

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

gap survives reseeding, likelihood scoring, and calibration. This empirical reversal motivates our central question: *why does a "weaker" discriminative head yield stronger hallucination detectors?*

In a static view of the fine-tuned representation space, we find that the two objectives induce markedly different hidden-state geometries. Under CLS, the model learns a task-specific discriminative direction and reorganizes hidden states into a linearly decodable two-cluster structure. Under S2S, the model instead mediates the binary decision through a pretrained full-vocabulary language-modeling head and leaves the hidden representation substantially closer to the pretrained backbone. We quantify this divergence with Fisher scores [Fisher, 1936], which reveal substantially stronger class separation by CLS. We further use layer-wise $\mathcal{V}$-information [Xu et al.] and linear probing [Alain and Bengio, 2016, Belinkov, 2022] to show that CLS exposes considerably more linearly decodable label information.

The static analysis also shows that the CLS signal saturates in the mid-to-upper layers, well before the final layer, forming a plateau of statistically interchangeable representations. This raises whether the plateau contains complementary task information or redundant encodings of the same binary signal. The answer implies different interventions: complementarity favors layer fusion, whereas redundancy favors upper-layer truncation or early exiting [Xin et al., 2020, Fan et al.]. We test these possibilities with bootstrap layer-interchangeability tests [Tibshirani and Efron, 1993] and partial information decomposition (PID), which separates redundant, unique, and synergistic contributions across sources [Williams and Beer, 2010, Bertschinger et al., 2014]. Both analyses support redundancy, suggesting that the plateau mainly preserves an already learned binary signal.

However, static redundancy alone cannot infer that the upper layers are dispensable during fine-tuning. These layers could still be necessary during optimization if they form task signals early and propagate gradients that influence how lower layers update. We therefore examine the training trajectory to identify where the classification signal first emerges and how it propagates across depth. Our training-dynamics analysis reveals that task-relevant information grows only within a narrow upper-middle band of three to four layers. Below this band, label decodability remains close to the pretrained level. Within the band, label information accumulates rapidly, so we name it the **InfoWindow**. Above it, layers become aligned with the task only at later stage of training and mainly inherit the signal produced within the InfoWindow. Together with the endpoint redundancy analysis, this temporal ordering suggests that the upper plateau reflects a signal that has already formed below.

This mechanism yields a principled early-exit strategy. We retain the lower substrate, the InfoWindow, and a short stabilization margin, and remove the redundant layers above them. On Qwen3-4B and 8B, this truncation removes 31% of the network, keeps accuracy within seed noise of full depth, and lowers latency by $1.4\times$. Since the static and dynamic analyses require substantial computation, we further evaluate whether the derived rule generalizes beyond the analyzed backbone. We fix the exit depths for Llama-3.2-3B and Mistral-7B-v0.3 as fractions of depth before running any analysis on them, and the transfer succeeds. Per-layer probing run afterwards finds the same substrate, InfoWindow, and plateau in both models, and again on QNLI and SST-2, suggesting that InfoWindow-guided truncation provides a practical route toward efficient task-specific LLM fine-tuning.

Our contributions are as follows: **i)** We measure the CLS–S2S gap under matched conditions and locate it in the learned representations, where CLS induces a stronger two-cluster geometry and higher label decodability that a fresh linear head cannot recover from S2S features (Sections 3 and 4). **ii)** We show that the CLS signal saturates into a statistically interchangeable and largely redundant upper plateau, as measured by bootstrap confidence intervals and PID (Section 5). **iii)** We identify an **InfoWindow** training dynamic, in which task information accumulates in a narrow upper-middle band and later synchronizes across the upper plateau (Section 5.3). **iv)** We turn these analyses into an early-exit rule that removes 31% of Qwen3 layers with accuracy unchanged within seed noise and $1.4\times$ lower latency, and transfers as a depth prior to Llama-3.2-3B and Mistral-7B-v0.3, where probing confirms the structure the prior assumes (Section 6, Appendix F).

2 Preliminaries

Problem Formulation. We formulate binary hallucination detection as sequence classification. Each example is a context–claim pair $x = (c, s)$ with label $Y \in \{0, 1\}$, where $Y = 1$ indicates that context c supports claim s, and $Y = 0$ indicates otherwise. A detector maps x to a prediction $\hat{Y}$. We fine-tune a pretrained decoder-only LLM backbone f_θ with L transformer blocks, and denote by $H_\ell(x) \in \mathbb{R}^D$ the hidden state of the final input token after block ℓ.

Fine-Tuning Paradigms. We compare two head–objective pairs under matched training conditions, isolating the fine-tuning paradigm as the only experimental variable.

The **LLM-as-Judge paradigm** (S2S) keeps the pretrained language-modeling head $W_{\text{LM}} \in \mathbb{R}^{|V| \times D}$ and casts detection as verdict generation. Given input x, the model generates a verbalized label sequence $y_{1:T}$ corresponding to `yes` or `no` and minimizes

$$\mathcal{L}_{\text{S2S}}(\theta) = -\sum_{t=1}^{T} \log \text{softmax}(W_{\text{LM}} h_t)_{y_t} . \tag{1}$$

The **classifier-head paradigm** (CLS) replaces the generative head with a newly initialized two-way linear head $W_{\text{CLS}} \in \mathbb{R}^{2 \times D}$ on the final hidden state $H_L(x)$ and minimizes

$$\mathcal{L}_{\text{CLS}}(\theta, W_{\text{CLS}}) = -\log \text{softmax}(W_{\text{CLS}} H_L(x))_Y . \tag{2}$$

Thus, we ask how replacing a pretrained full-vocabulary generative readout with a task-specific binary discriminative readout reshapes the hidden representations used for hallucination detection.

3 Empirical Gap Between CLS and S2S

Table 1: **Hallucination detection under matched fine-tuning conditions.** Macro-F_1 (%) is reported as mean ± standard deviation over three fine-tuning seeds (42, 43, 44); BASE is evaluated once. ID, OOD, and All are unweighted averages over the corresponding corpora, computed separately for each seed. Δ reports paired CLS−S2S differences.

Backbone	Method	In-distribution			Out-of-distribution			Average		
		RAGTruth	ClaimVerify	FactCheck	AggreFact	FaithBench	FavaBench	ID	OOD	All
Qwen3-4B	Base	55.7	70.5	71.0	80.9	24.3	69.4	65.7	58.2	62.0
	S2S LoRA	81.7 ±1.1	79.3 ±0.5	72.5 ±3.0	77.9 ±1.1	27.3 ±1.5	45.2 ±4.2	77.8 ±1.0	50.1 ±2.1	64.0 ±0.9
	CLS LoRA	86.8 ±0.1	79.4 ±1.0	76.9 ±2.1	83.1 ±0.3	32.7 ±0.3	96.7 ±0.5	81.0 ±1.0	70.8 ±0.4	75.9 ±0.5
	Δ (CLS−S2S)	+5.1 ±1.1	+0.1 ±1.5	+4.4 ±2.0	+5.2 ±1.1	+5.4 ±1.6	+51.5 ±4.0	+3.2 ±0.7	+20.7 ±2.0	+11.9 ±1.2
Qwen3-8B	Base	50.6	69.4	72.2	80.0	21.8	86.6	64.1	62.8	63.4
	S2S LoRA	83.1 ±0.2	78.3 ±0.4	73.7 ±1.4	79.7 ±1.6	32.5 ±0.4	81.2 ±7.6	78.4 ±0.5	64.5 ±3.0	71.4 ±1.3
	CLS LoRA	86.8 ±0.3	80.0 ±0.6	77.1 ±1.5	83.7 ±0.8	34.2 ±4.4	94.1 ±2.5	81.3 ±0.7	70.7 ±2.4	76.0 ±0.9
	Δ (CLS−S2S)	+3.7 ±0.4	+1.7 ±0.7	+3.4 ±2.2	+3.9 ±1.3	+1.7 ±4.1	+12.9 ±5.7	+2.9 ±0.7	+6.2 ±2.4	+4.6 ±0.9
Llama-3.2-3B	Base	41.4	36.0	44.1	29.1	43.5	43.4	40.5	38.7	39.6
	S2S LoRA	78.3 ±0.4	77.5 ±0.9	70.5 ±1.2	78.6 ±1.3	28.6 ±2.1	40.0 ±7.0	75.4 ±0.8	49.1 ±2.8	62.3 ±1.6
	CLS LoRA	84.6 ±0.7	79.6 ±0.5	73.2 ±2.3	77.6 ±1.9	29.8 ±1.6	49.6 ±18.8	79.1 ±0.8	52.3 ±7.4	65.7 ±3.3
	Δ (CLS−S2S)	+6.3 ±0.3	+2.1 ±1.3	+2.8 ±2.2	-1.0 ±2.1	+1.2 ±3.7	+9.5 ±25.4	+3.7 ±1.0	+3.2 ±10.1	+3.5 ±4.8
Mistral-7B-v0.3	Base	38.4	46.8	32.3	45.7	21.7	3.2	39.2	23.5	31.4
	S2S LoRA	81.4 ±0.6	79.5 ±0.6	72.8 ±0.6	79.6 ±1.3	25.6 ±0.6	63.0 ±5.4	77.9 ±0.2	56.1 ±2.1	67.0 ±1.0
	CLS LoRA	83.7 ±2.8	77.7 ±0.5	75.6 ±0.8	75.9 ±6.2	25.7 ±4.1	50.6 ±24.3	79.0 ±1.3	50.7 ±11.5	64.8 ±6.3
	Δ (CLS−S2S)	+2.2 ±2.5	-1.9 ±0.9	+2.8 ±1.3	-3.7 ±5.8	+0.1 ±4.7	-12.5 ±26.0	+1.1 ±1.1	-5.4 ±12.1	-2.2 ±6.6

Experimental Setup. We compare CLS and S2S under matched conditions such as data and hardware. We train rank-32 LoRA [Hu et al., 2022] adapters with $\alpha = 64$ on all projection matrices in every transformer block for three epochs with AdamW at learning rate 10^{-4}, using 22,827 training examples pooled from RAGTruth [Niu et al., 2024], ClaimVerify [Liu et al., 2023], and FactCheck-GPT [Wang et al., 2024a]. We report macro-F_1 on the eval-half of the corresponding in-distribution test splits and on the eval-half of three unseen out-of-distribution corpora, AggreFact [Tang et al., 2023], FavaBench [Mishra et al.], and FaithBench [Bao et al., 2025]; the remaining probe-half is reserved for the analyses in Sections 4–5. Every backbone–paradigm cell in Table 1 is trained with three fine-tuning seeds (42, 43, 44); we report the mean and standard deviation and compute the CLS−S2S gap per seed before averaging. We provide full experimental details in Appendix C.

Results. Table 1 shows a consistent gap between the two paradigms. On the in-distribution corpora, CLS leads S2S on every backbone, by 3.2 ± 0.7, 2.9 ± 0.7, 3.7 ± 1.0, and 1.1 ± 1.1 macro-F_1 points over three fine-tuning seeds, and the gap is positive in 11 of the 12 individual runs. The out-of-distribution gap is larger on average but far less stable. FavaBench has a positive rate of 0.20, and at the fixed 0.5 threshold individual seeds of either paradigm can collapse toward one class there (Appendix F.3). A paired bootstrap that excludes FavaBench still gives a gap of $+2.2$ to $+3.6$

points with $p \leq 0.004$ on all four backbones (Table 11). Stronger S2S readouts do not close the gap, where likelihood-ratio scoring, verbalizer ensembles, and per-corpus threshold calibration applied to both paradigms leave CLS ahead by 2.2 to 10.4 points, and so does a LoRA-adapted LM head (Appendix F.2). Throughout, S2S emits a single verdict token, so the head and loss are the only variables, and rationale-based judges fall outside this comparison (Section 8). This pattern suggests that the two objectives induce different internal representations. We next test this hypothesis through layer-wise geometry, information, redundancy, and training-dynamics analyses. We use endpoint diagnostics of class geometry and linearly decodable label information to trace the performance gap to objective-induced representation differences.

4 Objective-Induced Representation Geometry

Setup. We analyze the same Qwen3-4B and Qwen3-8B variants from Section 3: pretrained BASE, S2S-fine-tuned, and CLS-fine-tuned. We feed byte-identical inputs to all variants, cache final-token residual-stream states at each layer, and evaluate probes on held-out splits that are disjoint from the benchmark evaluation splits. We provide the full details in Appendix D.

4.1 Geometric Class Separation

From a representation perspective, if fine-tuning shapes hidden states into a binary task representation, the two classes should become more separable in hidden space. This criterion is natural here because both paradigms produce a binary verdict through a linear logit contrast in the final hidden state, with $w_1 - w_0$ for CLS and $w_{\texttt{Yes}} - w_{\texttt{No}}$ for our single-token S2S verdicts. We therefore compare how strongly each objective aligns hidden states with a binary decision direction. We use a shared CLS-defined projection for visualization and a direction-aligned Fisher score for quantification.

Visualization. For each scale, we set $w^{\text{disc}} \propto w_1 - w_0$ as the decision-normal axis and choose $w^{\perp}$ as the leading residual direction orthogonal to it. We then project BASE, S2S, and CLS onto the same axes with implementation details in Appendix D.1. Figure 1 shows that CLS yields the clearest separation along w^{disc} while BASE and S2S remain more mixed. This shared-axis comparison suggests that the binary geometry is induced most strongly by the CLS objective.

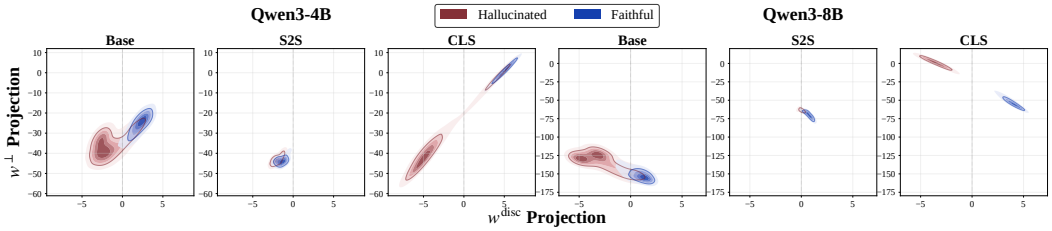

Figure 1: **Last-layer hidden-state geometry across paradigms and scales.** All variants at a fixed scale share the same CLS-defined projection. CLS shows the clearest separation along w^{disc} than S2S.

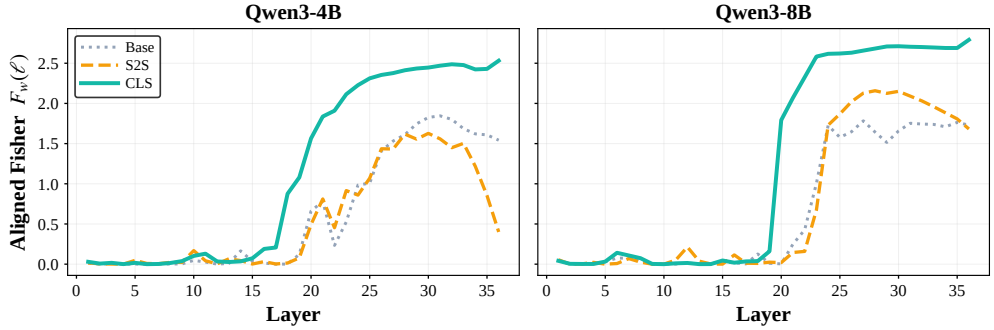

Figure 2: **Per-layer direction-aligned Fisher score.** CLS shows an earlier and stronger rise in separation along the learned decision direction than S2S.

Quantification. The shared projection provides a visual intuition of class separation. We next quantify this separation across layers by projecting each layer's hidden states onto the same CLS

decision direction w^{disc} and computing

$$F_w(\ell) \;=\; \frac{\big(w^{\text{disc}}\cdot(\mu_1(\ell)-\mu_0(\ell))\big)^2}{w^{\text{disc}\,\top}\big(\Sigma_0(\ell)+\Sigma_1(\ell)\big)w^{\text{disc}}}, \tag{3}$$

where $\mu_y(\ell)$ and $\Sigma_y(\ell)$ denote the class-conditional mean and covariance at layer ℓ. The score F_w extends the horizontal separation in Fig. 1 to every layer by measuring decision-axis class separation normalized by within-class spread, with full details provided in Appendix D.2. Figure 2 shows that CLS rises earlier and reaches a higher aligned-separation plateau than S2S at both scales.

Together, the visualization and Fisher analysis show that CLS more strongly shapes hidden states into a binary geometry than S2S. However, geometry is a structural diagnostic which does not measure how much label information a downstream readout can extract. We therefore next quantify linearly extractable label information across layers.

4.2 V-Information and Linear-Probe Decodability

Geometry captures class separation but does not directly measure how much label information a simple readout can extract. For every layer ℓ, we train a two-way logistic probe $g_\ell : \mathbb{R}^D \to \Delta^2$ on the final-token hidden state $h_\ell(x) \in \mathbb{R}^D$ and evaluate it on the in-domain held-out split. We report macro-F_1 for classification and $\mathcal{V}$-information [Xu et al.] for calibrated label information:

$$I_{\mathcal{V}}(\ell) = H(Y) - \mathrm{CE}(g_\ell), \tag{4}$$

where $H(Y)$ is the label entropy and $\mathrm{CE}(g_\ell)$ is the temperature-scaled cross-entropy of the layer-ℓ probe. Thus, $I_{\mathcal{V}}(\ell)$ measures how much label uncertainty a linear readout removes from the hidden state. We provide the full probing details in Appendix D.3.

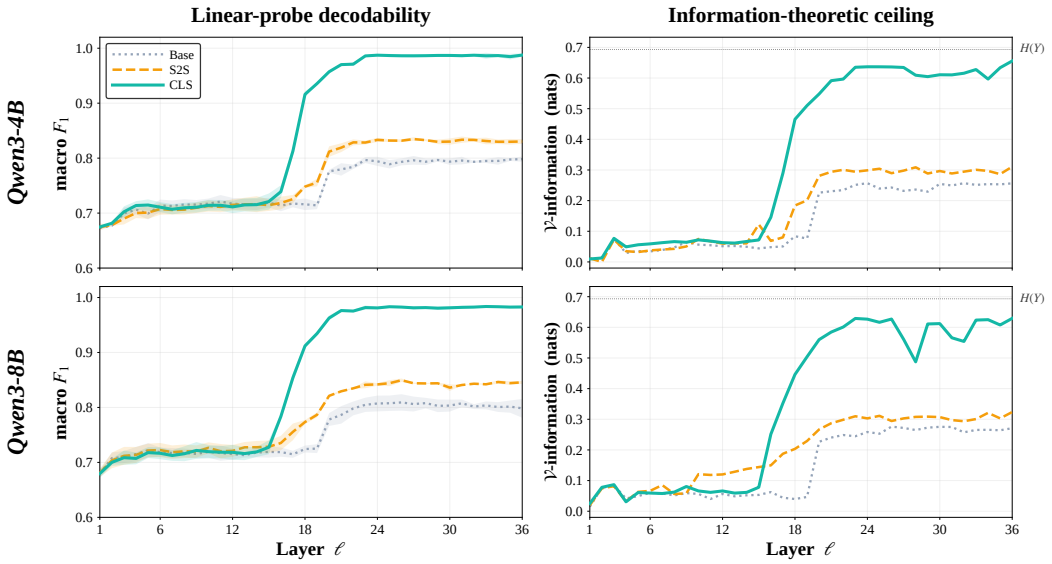

Figure 3: **Per-layer probe F_1 and $\mathcal{V}$-information.** CLS rises sharply in the mid-to-upper layers and saturates near the label-entropy ceiling, whereas S2S improves modestly over BASE.

Figure 3 shows that CLS exposes more label information to a linear readout than S2S. Probe F_1 and $I_{\mathcal{V}}$ rise sharply in the mid-to-upper CLS layers and saturate near the label-entropy ceiling, while S2S improves gradually over the pretrained backbone and plateaus lower. Together with the Fisher analysis, this shows that CLS aligns hidden states with a binary decision direction and makes the signal more linearly decodable, whereas S2S remains closer to the pretrained representation. The same ordering holds when both heads are discarded and one fresh linear readout is fit on frozen features: the S2S representation trails CLS by 0.13 to 0.15 F_1 on all four backbones (Appendix F.4).

The layer-wise curves also reveal a second, more actionable pattern: **CLS *reaches near-final label decodability well before the final layer***. Once the signal becomes decodable, the key question is what

the remaining layers add. If they contribute complementary evidence, the plateau is informationally heterogeneous and may benefit from layer fusion. If they mainly preserve the same binary signal, the plateau is homogeneous and may be safely truncated. Therefore in next section, we analyze the information structure of the saturated CLS representations.

5 The Upper Plateau in CLS Representations

The early saturation in Section 4.2 raises a new question: ***what do the remaining upper layers add after the task signal is already decodable?*** In this section, we address this question in three steps.

5.1 Upper-Layer Saturation and Plateau Interchangeability

The $\mathcal{V}$-information curves of Sec. 4.2 visually saturate well before the final layer. We now formalize that observation as a statistical claim: ***from which layer onward does a single linear readout match the final-layer readout?***

For each candidate layer ℓ, we train a logistic probe on the layer-ℓ hidden states and evaluate its macro-F_1 on the in-domain held-out split. Let $L_{\max}$ denote the final layer. We form the per-layer F_1 gap to the final layer,

$$\Delta_{\text{final}}(\ell) \;=\; F_1(\ell) - F_1(L_{\max}),$$

and compute its 95% confidence interval (CI) by paired bootstrap with 1,000 replicates on the same test samples, which preserves sample-level correlation between the two probes' predictions. We define the saturation layer as the smallest layer whose F_1 gap to the final layer is statistically indistinguishable from zero,

$$L^\star \;=\; \min\big\{\, \ell \,:\, 0 \in \widehat{\mathrm{CI}}_{0.95}\big(\Delta_{\text{final}}(\ell)\big) \,\big\}.$$

The candidate range is the band where fine-tuning measurably raises linear-probe decodability over the pretrained backbone (L16 onward). We provide the full protocol in Appendix D.4.

Table 2: **Bootstrap plateau test for upper-layer saturation in CLS.** $L^\star$ is the lowest layer whose probe F_1 is statistically indistinguishable from the final-layer under a 1000-sample bootstrap 95% CI. A confidence interval containing zero indicates readout equivalence to the final layer.

Backbone	$L^\star$	$L^\star/L_{\max}$	Δ_{final} *[95% CI]*	Equiv.
Qwen3-4B	**L22**	**61%**	$\mathbf{+0.001}$ $[-0.007, +0.008]$	✓
Qwen3-8B	**L23**	**64%**	$\mathbf{-0.002}$ $[-0.009, +0.005]$	✓

Table 2 shows that CLS reaches final-layer-equivalent readout performance well before the final layer. $L^\star$ lies at L22 for Qwen3-4B and L23 for Qwen3-8B, corresponding to 61–64% of the backbone, and both 95% CIs for Δ_{final} comfortably contain zero. The upper CLS layers therefore reach a readout-equivalent saturation point at $L^\star$ for the synchronous-saturation band.

However, readout equivalence does not imply information redundancy. Two layers can produce the same probe F_1 while encoding different task-relevant evidence, and that distinction has opposite engineering implications: heterogeneous upper layers favor fusion, homogeneous upper layers favor truncation. Therefore, we next test which case actually holds.

5.2 Joint Probing and Partial Information Decomposition

The plateau test in Section 5.1 shows that earlier upper CLS layers can match the final layer as single-layer readouts. However, readout equivalence does not specify the information structure of the plateau. Layers with the same probe F_1 may encode similar task information, or they may encode different task-relevant evidence that a joint readout could exploit. This distinction determines how the plateau should be used: homogeneous information would support truncation, whereas heterogeneous information would support fusion. We therefore ask: ***is the upper plateau informationally homogeneous or heterogeneous?*** We study this question at two levels.

First, we test the information structure of the plateau with joint probing and partial information decomposition (PID) [Williams and Beer, 2010]. For each candidate layer ℓ and a plateau layer p, we

Table 3: **PID-style and fusion tests support upper-plateau homogeneity in CLS.** At the synergy-optimal layer $\ell^\star$, the candidate layer contributes zero unique information once a plateau layer is available, and the synergy budget remains small. Fair OOD ΔF_1 reports fusion minus single-layer readout, with 10,000-sample paired-bootstrap 95% CI stratified by dataset. Neither CLS backbone shows a positive fusion gain, supporting truncation rather than layer fusion.

Backbone	$\ell^\star$	$\widehat{UI}_\ell$	$\widehat{\mathcal{S}}$ *(nats)*	Fair OOD ΔF_1 *[95% CI, pp]*
Qwen3-4B	**L18**	**0**	**0.053**	= **-0.2 pp** $[-0.6, +0.2]^{\text{ns}}$
Qwen3-8B	**L12**	**0**	**0.049**	↓ **0.9 pp** $[-1.7, -0.0]$

train a fusion probe on (H_ℓ, H_p) and two marginal probes on H_ℓ and H_p alone. We then decompose the joint $\mathcal{V}$-information into redundant, unique, and synergistic components:

$$I_\mathcal{V}\big(H_\ell, H_p; Y\big) \;=\; R \;+\; UI_\ell \;+\; UI_p \;+\; \mathcal{S},$$

where R is shared information, UI_ℓ and UI_p are layer-unique contributions, and $\mathcal{S}$ is synergy. We implement this decomposition with a framework-free min-MI bound and report the estimated atoms $\widehat{R}$, $\widehat{UI}_\ell$, $\widehat{UI}_p$, and $\widehat{\mathcal{S}}$; Appendix D.5 gives the estimator and compatibility conditions. A heterogeneous plateau should yield non-trivial unique or synergistic information, whereas a homogeneous plateau should concentrate most task information in the redundant component. To give the strongest chance for detecting heterogeneity, we report the decomposition at the synergy-optimal candidate layer,

$$\ell^\star = \arg\max_\ell \widehat{\mathcal{S}}(\ell).$$

If any layer carries information complementary to the plateau, this choice should make it most visible.

Second, we test the corresponding deployment implication. If plateau layers provide complementary evidence, a trained fusion readout should outperform a single-layer readout under the same data and optimization budget. We therefore compare a fusion probe against the corresponding single-layer probe on the six-corpus held-out split, covering both in-domain and out-of-domain datasets. We report fusion-minus-single-layer macro-F_1 with a 10,000-sample paired bootstrap stratified by dataset. Appendix D.5 gives the full implementation details.

Table 3 supports the homogeneous-plateau interpretation. For both Qwen3 scales, the synergy-optimal layer contributes zero unique information once a plateau layer is available, and the synergy remains below 0.06 nats, less than 9% of the binary label entropy $H(Y) = \ln 2$. Repeating the decomposition with different plateau layers gives the same conclusion within seed noise. The fusion test reaches the same conclusion at the deployment level. Fusion changes macro-F_1 by -0.2 pp on Qwen3-4B, with a CI containing zero, and by -0.9 pp on Qwen3-8B. Thus, plateau layers behave less like complementary sources of evidence, which favors truncation over fusion.

However, static endpoint redundancy does not imply that upper layers are unnecessary during training. They could still help the lower or middle layers form the task signal before becoming redundant at the endpoint. We therefore next examine the training trajectory.

5.3 Training Dynamics of the Plateau

The endpoint analyses show that the upper CLS layers contain largely redundant label information. However, static redundancy alone does not show how this plateau forms. The upper layers might still play an essential role during training if they create the task-aligned signal or sequentially refine complementary evidence across depth. We therefore turn to the training trajectory and ask ***when different layers acquire linearly decodable label information*** during fine-tuning.

To describe how the task signal emerges, we track when each layer gains linearly decodable label information during fine-tuning. At each checkpoint t, we extract final-token hidden states at every layer, fit a fast linear probe, and compute $I_\mathcal{V}(\ell, t)$. This gives a layer-by-time trajectory of label information. We summarize each layer's trajectory with two quantities. The first is the total information gain,

$$\Delta I_\mathcal{V}(\ell) = I_\mathcal{V}(\ell, T) - I_\mathcal{V}(\ell, 0),$$

which measures how much additional label information becomes linearly extractable during training. The second is the saturation time $t_\ell^{0.9}$, defined as the first checkpoint at which layer ℓ reaches 90%

of its final $I_{\mathcal{V}}$ value. Together, these quantities tell us whether a layer learns a task-aligned signal, and when that signal becomes decodable. We use these trajectories to compare when different layers acquire linearly decodable label information. We first inspect the layer-by-time structure of $I_{\mathcal{V}}(\ell, t)$ and then use the information-gain and saturation-time statistics to summarize the pattern. For reference, we use $\tau = 0.10$ to mark non-trivial information gain and $\tau_{\text{step}} = 1000$ training steps to mark near-synchronous saturation. We provide the full details in Appendix D.6.

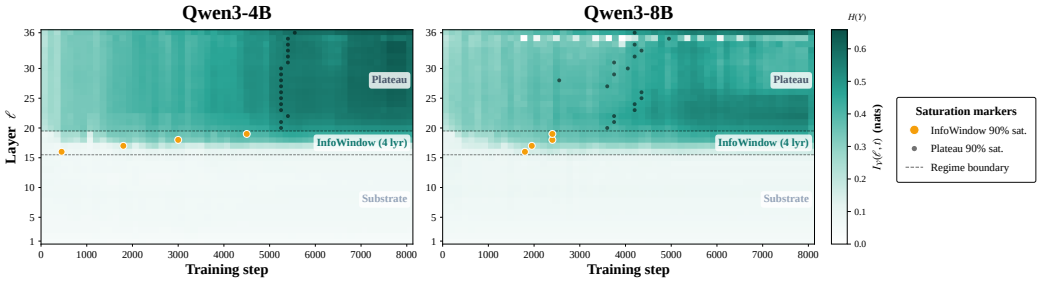

Figure 4: **Per-layer $\mathcal{V}$-information across the CLS training trajectory.** The figure plots layers as rows and saved checkpoints as columns, with color indicating $I_{\mathcal{V}}(\ell, t)$ in nats. The lower substrate (L1–L15) remains near its pretrained level, the InfoWindow (L16–L19) shows staggered task-information growth, and the upper plateau (L20–L36) synchronizes later within a tight training-step cluster.

Figure 4 reveals three dynamic regimes. **1) Lower substrate.** Layers L1–L15 remain close to their pretrained level and gain little additional linearly decodable label information, suggesting that they mainly provide general features. **2) InfoWindow.** Layers L16–L19 show the first non-trivial growth in $I_{\mathcal{V}}$ and saturate in a staggered manner across depth, so we refer to this signal-forming band as the **InfoWindow**. Qwen3-4B exhibits an upward-moving signal through this band, while Qwen3-8B shows the same depth-staggered pattern in temporally compressed form. **3) Synchronous upper plateau.** Layers L20–L36 reach high $\mathcal{V}$-information later and within a tight training-step window, forming a synchronous plateau. This temporal pattern suggests that the upper plateau aligns to an already decodable signal instead of constructing heterogeneous representation.

This temporal order connects the dynamic evidence with the endpoint redundancy analysis. At the endpoint, the upper plateau contains largely homogeneous label information; during training, the same plateau appears only after the InfoWindow has already formed a decodable binary signal. Thus, the upper layers look less like the source of task information and more like late redundant carriers of an already formed representation. The resulting partition motivates the early-exit rule in the next section, where we preserve the substrate and InfoWindow and test whether removing layers above the saturated plateau boundary yields a parameter-efficient detector.

6 From Plateau Analysis to Early Exit

In this section, we turn the InfoWindow into an early-exit intervention. On the analyzed Qwen3 backbones, the InfoWindow gives a principled exit rule. For other backbones, the exit depth of Qwen3 can serve as a depth-fractional prior for selecting useful analysis-free candidate exits.

InfoWindow-Guided Exits on Qwen3. On Qwen3, the InfoWindow spans L16–L19 and the synchronous upper plateau begins at L20, while the readout-equivalence point $L^\star$ appears slightly later at L22/L23 for 4B/8B. This gap suggests that although task information first emerges inside the InfoWindow, the representation needs a short stabilization margin before it reaches final-layer-level readout quality. We therefore select three proof-of-concept exits aligned with these regimes: L15 cuts before the signal-forming window, L20 reads out at the first plateau layer, and L25 keeps the InfoWindow plus a five-layer stabilization margin. Thus, L25 removes roughly 31% of the forward-pass depth while preserving the signal-forming band and its post-InfoWindow buffer. These three quantities are measured on the probe half alone, and the eval half enters only the final numbers in Table 4 (Appendix E).

Analysis-Free Transfer via a Depth Prior. For Llama-3.2-3B ($L = 28$) and Mistral-7B-v0.3 ($L = 32$), we fix the exits before running any representation analysis on either model, so the transfer

Table 4: **Early-exit accuracy and retained depth.** Macro-F_1 (%) is reported as mean $\pm$ standard deviation over three fine-tuning seeds (42, 43, 44), except for L15 (seed 42 only). "Saved" is the fraction of transformer blocks removed; Δ is the six-corpus average difference from the full-depth model, paired by seed. Exit depths are held fixed across runs.

Backbone	Exit position	Verdict	Layer	Saved	RAGTruth	ClaimVerify	FactCheck	AggreFact	FaithBench	FavaBench	Avg.	Δ
Analyzed backbones — exit chosen from the per-model InfoWindow analysis (Section 5.3)												
Qwen3-4B	Full CLS (baseline)	—	36	—	86.8 ±0.1	79.4 ±1.0	76.9 ±2.1	83.1 ±0.3	32.7 ±0.3	96.7 ±0.5	75.9 ±0.5	—
	Before InfoWindow	× **too early**	15	58%	84.9	79.1	74.5	75.7	28.0	52.9	65.9	-9.9 (n=1)
	First plateau layer	∼ **borderline**	20	44%	85.0 ±1.3	80.1 ±1.5	76.8 ±1.2	79.2 ±2.4	27.7 ±4.5	79.2 ±18.8	71.4 ±4.3	-4.5 ±4.8
	InfoWindow + margin	✓ **prescribed**	25	31%	86.4 ±0.9	80.2 ±0.4	76.7 ±0.4	82.3 ±1.5	32.0 ±2.6	94.6 ±2.9	75.4 ±1.1	-0.5 ±1.4
Qwen3-8B	Full CLS (baseline)	—	36	—	86.8 ±0.3	80.0 ±0.6	77.1 ±1.5	83.7 ±0.8	34.2 ±4.4	94.1 ±2.5	76.0 ±0.9	—
	Before InfoWindow	× **too early**	15	58%	83.3	79.9	72.5	75.1	28.7	37.5	62.8	-14.4 (n=1)
	First plateau layer	∼ **borderline**	20	44%	85.9 ±0.2	80.3 ±0.5	74.9 ±0.4	82.9 ±0.3	30.6 ±2.0	95.7 ±0.9	75.1 ±0.6	-0.9 ±1.2
	InfoWindow + margin	✓ **prescribed**	25	31%	86.8 ±0.2	80.8 ±0.2	75.1 ±0.9	85.1 ±0.1	35.8 ±2.4	97.2 ±0.6	76.8 ±0.4	+0.8 ±1.1
Transfer backbones — candidate exits from the depth prior ($\lceil 0.70L \rceil$ / round$(0.60L)$); selected before per-model probing												
Llama-3.2-3B	Full CLS (baseline)	—	28	—	84.6 ±0.7	79.6 ±0.5	73.2 ±2.3	77.6 ±1.9	29.8 ±1.6	49.6 ±18.8	65.7 ±3.3	—
	Conservative (f=0.70)	—	20	29%	85.5 ±0.6	79.4 ±0.4	74.2 ±2.2	78.5 ±0.5	32.2 ±0.5	34.4 ±5.9	64.0 ±0.9	-1.7 ±3.8
	Aggressive (f=0.60)	—	17	39%	84.9 ±0.2	81.3 ±0.5	73.9 ±1.4	78.9 ±0.1	30.0 ±0.4	55.7 ±14.7	67.5 ±2.5	+1.7 ±1.9
Mistral-7B-v0.3	Full CLS (baseline)	—	32	—	83.7 ±2.8	77.7 ±0.5	75.6 ±0.8	75.9 ±6.2	25.7 ±4.1	50.6 ±24.3	64.8 ±6.3	—
	Conservative (f=0.70)	—	23	28%	84.7 ±0.1	78.2 ±0.5	74.9 ±1.0	81.0 ±1.0	29.1 ±1.2	74.9 ±10.7	70.4 ±1.8	+5.6 ±5.9
	Aggressive (f=0.60)	—	19	41%	81.6 ±3.3	77.3 ±3.2	75.0 ±1.1	77.3 ±4.3	28.0 ±2.2	65.5 ±14.2	67.4 ±3.7	+2.6 ±2.7

is an out-of-sample test of the depth prior. Instead, we transfer the Qwen3 exit as two pre-specified retained-depth ratios, with conservative $\ell_{\text{exit}} = \lceil 0.70L \rceil$ and aggressive $\ell_{\text{exit}} = \text{round}(0.60L)$. We use the conservative rule to approximate Qwen3's retained ratio $25/36 \approx 0.69$, and we use the aggressive rule to test a deeper cut. These rules give Llama exits at L20/L17 and Mistral exits at L23/L19. We then test whether this {conservative, aggressive} depth prior identifies at least one competitive efficient model per backbone. We provide full training details in Appendix E.

Early-Exit Results. Table 4 reports macro-F_1 over three fine-tuning seeds. The Qwen3 results follow the InfoWindow prediction: the L15 exits remove the signal-forming band and drop sharply, the L20 exits keep too little plateau margin and fluctuate across seeds (-4.5 ± 4.8 on 4B), and the L25 exits preserve the InfoWindow plus a short margin and stay within seed noise of full depth (-0.5 ± 1.4 and $+0.8 \pm 1.1$) with 31% fewer layers. The saving is realized in wall-clock time as well: with the blocks above the exit removed, per-example latency falls by $1.4\times$ at L25 and $1.8\times$ at L20 on both an H200 and an RTX A6000 (Appendix F.6). The transfer results show that the Qwen3-derived depth prior identifies one competitive efficient variant for each additional backbone, although single-seed cells move by several points, so we read the ratio as a prior over candidates rather than a prescription. Together, these results show that the InfoWindow provides a principled early-exit rule for analyzed models and a lightweight depth prior for unanalyzed models.

Discussion and Analysis. After the transfer results, we ran the per-layer probing of Section 4.2 on both backbones, and then on Qwen3-4B fine-tuned on QNLI and SST-2 with the same recipe. Both transfer backbones show the same substrate, InfoWindow, and plateau, with the CLS saturation layer at 0.43 and 0.56 of depth. All four pre-specified exits lie above it and all four stay within seed noise of full depth or above it, whereas the one Qwen3 exit below its saturation layer, L20, is the unstable one. QNLI and SST-2 reproduce the structure with the saturation layer at 0.61 and 0.50 of depth, while the CLS advantage shrinks to 0.02 and 0.00 F_1 because the pretrained backbone already decodes both tasks at 0.89 to 0.94. Appendix F reports these probes together with the seed-level runs behind Tables 1 and 4, the stronger S2S readouts, the FavaBench diagnosis, and the measured latency.

7 Related Work

Hallucination detection. Hallucination detection has been studied through factuality benchmarks, self-checking methods, LLM-as-judge protocols, and lightweight detectors over model states or outputs [Manakul et al., 2023, Niu et al., 2024, Wang et al., 2024b, Tang et al., 2024, Belyi et al., 2025]. Prior work suggests that truthfulness or factuality signals can be exposed either from internal LLM representations or from model self-evaluations, including activation-based classifiers, unsupervised truth directions, activation interventions, and verbalized $P(\text{True})$ estimates [Azaria and Mitchell, 2023, Burns et al., Kadavath et al., 2022, Li et al., 2023]. We build on this evidence and study how such signals are shaped during CLS and S2S fine-tuning.

Layer-wise probing and representation analysis. Layer-wise probing is widely used to study what information neural representations make accessible to simple readouts [Alain and Bengio, 2016, Belinkov, 2022, Tenney et al., 2019, Hewitt and Manning, 2019]. Information-theoretic variants further quantify usable label information and distinguish shared from complementary signals across representations [Xu et al., Williams and Beer, 2010, Bertschinger et al., 2014]. We build on these tools to ask how two fine-tuning objectives shape the same backbone. We track Fisher separation, $\mathcal{V}$-information, and redundancy-related measurements across checkpoints, connecting endpoint layer-wise structure to the training dynamics that produced it.

Early exit for transformers. Early-exit methods reduce transformer inference cost by adding intermediate classifiers, dropping layers, or gating computation at inference time [Xin et al., 2020, Liu et al., 2020, Fan et al., Xin et al., 2021, Zhou et al., 2020, Elhoushi et al., 2024, Chen et al., 2024]. They make exit decisions adaptively for each input and choose exits or thresholds using validation performance or confidence criteria. We use layer-wise training dynamics to choose a single static cutoff for all inputs. The InfoWindow trajectory provides a representation-based criterion for choosing this cutoff, complementing confidence- or validation-based early-exit rules.

8 Conclusion and Future Work

We studied why classifier-head fine-tuning (CLS) outperforms LLM-as-Judge sequence-to-sequence fine-tuning (S2S) on binary hallucination detection under matched conditions. Our analyses show that CLS reshapes the backbone into a more discriminative representation, with clearer two-cluster geometry, stronger class separation, and more linearly decodable label information, whereas S2S leaves hidden states closer to the pretrained representation. The layer-wise results further show that the CLS signal emerges in a narrow upper-middle **InfoWindow** and saturates before the final layer. The layers above this region form a statistically interchangeable and largely redundant plateau. This plateau structure motivates a simple static early-exit rule that removes layers above the InfoWindow. The resulting truncation removes 31% of the network on Qwen3-4B and 8B with accuracy unchanged within seed noise and latency reduced by $1.4\times$, and the same depth fraction, fixed before any analysis of the target models, transfers to Llama-3.2-3B and Mistral-7B-v0.3. Probing those models afterwards finds the same substrate, InfoWindow, and plateau, and so does probing Qwen3-4B fine-tuned on QNLI and SST-2, which places the structure beyond the task and backbone family it was derived on. Together, these findings suggest that objective-induced representation structure can help explain detector accuracy and guide efficient task-specific LLM design.

Several directions remain open. First, we restrict S2S to a single verdict token so that the objective is the only variable. Judges that generate a rationale before the verdict add supervision and inference cost that our deployment setting does not allow, and comparing them with CLS where that cost is acceptable is a natural next step. Second, the layer structure is general, but the CLS advantage is not. It is large on hallucination detection and near zero on QNLI and SST-2 (Section 6, Appendix F.5), and whether the exit rule itself transfers to such tasks remains untested. Third, our early-exit experiments use proof-of-concept cutoff choices and one static exit for all inputs. Future work could select exits from representation-level signals more systematically, combine InfoWindow-guided truncation with input-adaptive exits, distillation, or pruning, and design objectives that shape these representations more directly. Fourth, we have traced how label information and redundancy are distributed across layers. It remains open how they are distributed across features within a layer and which features carry the signal.

References

Guillaume Alain and Yoshua Bengio. Understanding intermediate layers using linear classifier probes. *arXiv preprint arXiv:1610.01644*, 2016.

Amos Azaria and Tom Mitchell. The internal state of an llm knows when it's lying. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 967–976, 2023.

Forrest Bao, Miaoran Li, Renyi Qu, Ge Luo, Erana Wan, Yujia Tang, Weisi Fan, Manveer Singh Tamber, Suleman Kazi, Vivek Sourabh, et al. Faithbench: A diverse hallucination benchmark for summarization by modern llms. In *Proceedings of the 2025 Conference of the Nations*

of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 2: Short Papers), pages 448–461, 2025.

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

Masha Belyi, Robert Friel, Shuai Shao, and Atindriyo Sanyal. Luna: A lightweight evaluation model to catch language model hallucinations with high accuracy and low cost. In *Proceedings of the 31st international conference on computational linguistics: industry track*, pages 398–409, 2025.

Nils Bertschinger, Johannes Rauh, Eckehard Olbrich, Jürgen Jost, and Nihat Ay. Quantifying unique information. *Entropy*, 16(4):2161–2183, 2014.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

Collin Burns, Haotian Ye, Dan Klein, and Jacob Steinhardt. Discovering latent knowledge in language models without supervision. In *The Eleventh International Conference on Learning Representations*.

Yanxi Chen, Xuchen Pan, Yaliang Li, Bolin Ding, and Jingren Zhou. Ee-llm: Large-scale training and inference of early-exit large language models with 3d parallelism. In *International Conference on Machine Learning*, pages 7163–7189. PMLR, 2024.

Mostafa Elhoushi, Akshat Shrivastava, Diana Liskovich, Basil Hosmer, Bram Wasti, Liangzhen Lai, Anas Mahmoud, Bilge Acun, Saurabh Agarwal, Ahmed Roman, et al. Layerskip: Enabling early exit inference and self-speculative decoding. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12622–12642, 2024.

Angela Fan, Edouard Grave, and Armand Joulin. Reducing transformer depth on demand with structured dropout. In *International Conference on Learning Representations*.

Ronald A Fisher. The use of multiple measurements in taxonomic problems. *Annals of eugenics*, 7 (2):179–188, 1936.

Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Jiawei Sun, Haofen Wang, et al. Retrieval-augmented generation for large language models: A survey.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Jiawei Gu, Xuhui Jiang, Zhichao Shi, Hexiang Tan, Xuehao Zhai, Chengjin Xu, Wei Li, Yinghan Shen, Shengjie Ma, Honghao Liu, et al. A survey on llm-as-a-judge. *The Innovation*.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. In *International conference on machine learning*, pages 1321–1330. PMLR, 2017.

John Hewitt and Christopher D Manning. A structural probe for finding syntax in word representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4129–4138, 2019.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for nlp. In *International conference on machine learning*, pages 2790–2799. PMLR, 2019.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Liang Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. *Iclr*, 1(2):3, 2022.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mistral 7b, 2023. URL `https://arxiv.org/abs/2310.06825`.

Saurav Kadavath, Tom Conerly, Amanda Askell, Tom Henighan, Dawn Drain, Ethan Perez, Nicholas Schiefer, Zac Hatfield-Dodds, Nova DasSarma, Eli Tran-Johnson, et al. Language models (mostly) know what they know. *arXiv preprint arXiv:2207.05221*, 2022.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33: 9459–9474, 2020.

Kenneth Li, Oam Patel, Fernanda Viégas, Hanspeter Pfister, and Martin Wattenberg. Inference-time intervention: Eliciting truthful answers from a language model. *Advances in Neural Information Processing Systems*, 36:41451–41530, 2023.

Nelson F Liu, Tianyi Zhang, and Percy Liang. Evaluating verifiability in generative search engines. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 7001–7025, 2023.

Weijie Liu, Peng Zhou, Zhiruo Wang, Zhe Zhao, Haotang Deng, and Qi Ju. Fastbert: a self-distilling bert with adaptive inference time. In *Proceedings of the 58th annual meeting of the association for computational linguistics*, pages 6035–6044, 2020.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, et al. Self-refine: Iterative refinement with self-feedback. *Advances in neural information processing systems*, 36:46534–46594, 2023.

Potsawee Manakul, Adian Liusie, and Mark Gales. Selfcheckgpt: Zero-resource black-box hallucination detection for generative large language models. In *Proceedings of the 2023 conference on empirical methods in natural language processing*, pages 9004–9017, 2023.

Abhika Mishra, Akari Asai, Vidhisha Balachandran, Yizhong Wang, Graham Neubig, Yulia Tsvetkov, and Hannaneh Hajishirzi. Fine-grained hallucination detection and editing for language models. In *First Conference on Language Modeling*.

Cheng Niu, Yuanhao Wu, Juno Zhu, Siliang Xu, KaShun Shum, Randy Zhong, Juntong Song, and Tong Zhang. Ragtruth: A hallucination corpus for developing trustworthy retrieval-augmented language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 10862–10878, 2024.

Hadas Orgad, Michael Toker, Zorik Gekhman, Roi Reichart, Idan Szpektor, Hadas Kotek, and Yonatan Belinkov. Llms know more than they show: On the intrinsic representation of llm hallucinations. In *The Thirteenth International Conference on Learning Representations*.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

Liyan Tang, Tanya Goyal, Alex Fabbri, Philippe Laban, Jiacheng Xu, Semih Yavuz, Wojciech Kryściński, Justin Rousseau, and Greg Durrett. Understanding factual errors in summarization: Errors, summarizers, datasets, error detectors. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 11626–11644, 2023.

Liyan Tang, Philippe Laban, and Greg Durrett. Minicheck: Efficient fact-checking of llms on grounding documents. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 8818–8847, 2024.

Ian Tenney, Dipanjan Das, and Ellie Pavlick. Bert rediscovers the classical nlp pipeline. In *Proceedings of the 57th annual meeting of the association for computational linguistics*, pages 4593–4601, 2019.

Robert J Tibshirani and Bradley Efron. An introduction to the bootstrap. *Monographs on statistics and applied probability*, 57(1):1–436, 1993.

Kaiyang Wan, Honglin Mu, Rui Hao, Haoran Luo, Tianle Gu, and Xiuying Chen. A cognitive writing perspective for constrained long-form text generation. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 9832–9844, 2025.

Kaiyang Wan, Lang Gao, Honglin Mu, Preslav Nakov, Yuxia Wang, and Xiuying Chen. A fano-style accuracy upper bound for llm single-pass reasoning in multi-hop qa. In *International Conference on Learning Representations*, volume 2026, pages 52600–52621, 2026.

Yuxia Wang, Revanth Gangi Reddy, Zain Muhammad Mujahid, Arnav Arora, Aleksandr Rubashevskii, Jiahui Geng, Osama Mohammed Afzal, Liangming Pan, Nadav Borenstein, Aditya Pillai, et al. Factcheck-bench: Fine-grained evaluation benchmark for automatic fact-checkers. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 14199–14230, 2024a.

Yuxia Wang, Minghan Wang, Muhammad Arslan Manzoor, Fei Liu, Georgi Nenkov Georgiev, Rocktim Jyoti Das, and Preslav Nakov. Factuality of large language models: A survey. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen, editors, *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 19519–19529, Miami, Florida, USA, November 2024b. Association for Computational Linguistics.

Paul L Williams and Randall D Beer. Nonnegative decomposition of multivariate information. *arXiv preprint arXiv:1004.2515*, 2010.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 conference on empirical methods in natural language processing: system demonstrations*, pages 38–45, 2020.

Ji Xin, Raphael Tang, Jaejun Lee, Yaoliang Yu, and Jimmy Lin. Deebert: Dynamic early exiting for accelerating bert inference. In *Proceedings of the 58th annual meeting of the association for computational linguistics*, pages 2246–2251, 2020.

Ji Xin, Raphael Tang, Yaoliang Yu, and Jimmy Lin. Berxit: Early exiting for bert with better fine-tuning and extension to regression. In *Proceedings of the 16th conference of the European chapter of the association for computational linguistics: Main Volume*, pages 91–104, 2021.

Yilun Xu, Shengjia Zhao, Jiaming Song, Russell Stewart, and Stefano Ermon. A theory of usable information under computational constraints. In *International Conference on Learning Representations*.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in neural information processing systems*, 36:46595–46623, 2023.

Wangchunshu Zhou, Canwen Xu, Tao Ge, Julian McAuley, Ke Xu, and Furu Wei. Bert loses patience: Fast and robust inference with early exit. *Advances in Neural Information Processing Systems*, 33: 18330–18341, 2020.

A Limitations and Broader Impact

Limitations. This study focuses on binary hallucination detection and evaluates a representative set of decoder-only LLM backbones. Future work could extend the analysis to more fine-grained factuality labels, additional architectures, and other verification tasks. Our early-exit rule uses a static cutoff shared by all inputs; combining InfoWindow-guided truncation with input-adaptive exiting may further improve the efficiency–accuracy trade-off. We also restrict S2S to a single verdict token as a matched control, and the CLS advantage we measure is large on hallucination detection and small on QNLI and SST-2 (Section 8).

Broader impact. Hallucination detection is an important component of trustworthy LLM systems, especially in retrieval-augmented generation and agentic workflows where model outputs are often passed to downstream users or tools. By studying how fine-tuning objectives shape detector representations, this work provides evidence for designing stronger and more efficient verification models. The early-exit results suggest that hallucination detectors can be made cheaper without sacrificing accuracy, which may make verification more accessible in resource-constrained deployments. As with any automated verifier, these detectors should complement rather than replace task-specific evaluation and human oversight in high-stakes applications.

B Existing assets and licenses

All datasets, model backbones, and software packages used in this work are existing publicly available research assets. We cite the original sources for these assets and use them in accordance with the access conditions and usage terms specified by their original providers or hosting platforms.

C Experimental Setup Details

This section provides the full experimental protocol underlying the compact description in §3. We emphasize the details that affect the validity of the objective comparison: model choice, data provenance, split construction, fine-tuning parity, and the separation between analysis and benchmark evaluation.

Models. Our main representation analysis uses Qwen3-4B and Qwen3-8B [Yang et al., 2025], two decoder-only transformers from the same family. Both models have 36 transformer blocks but differ in width, which lets us compare per-layer measurements across scales without changing the layer index. To test whether the main CLS versus S2S pattern depends on this model family, we also replicate the headline comparison on Llama-3.2-3B [Grattafiori et al., 2024] and Mistral-7B-v0.3 [Jiang et al., 2023]. All backbones run in bfloat16 with SDPA attention and a 2048-token training context window.

Data sources. We build the fine-tuning set from three binary hallucination-detection corpora: RAGTruth [Niu et al., 2024], ClaimVerify [Liu et al., 2023], and FactCheck-GPT [Wang et al., 2024a], for a total of $22{,}827$ raw training rows. For out-of-distribution evaluation, we use three held-out corpora that are never used during fine-tuning: AggreFact [Tang et al., 2023] for summarization factuality, FavaBench [Mishra et al.] for fine-grained hallucination types, and FaithBench [Bao et al., 2025] for summarization. All six corpora provide paired training and testing files, except AggreFact, FaithBench, and FavaBench, which contribute test files only. Table 5 reports the per-file row counts and class balance after preprocessing.

Test split construction: deduplication followed by stratified 50/50 splitting. For every test file, we first deduplicate examples by the $(\text{premise}, \text{hypothesis})$ fingerprint. This step removes literal duplicate pairs before any split is created; in particular, we found 7 duplicate pairs in RAGTruth and 51 in AggreFact. Without this deduplication, identical content could appear in both halves even when the example indices are disjoint. We then split each deduplicated file into a stratified $50/50$ *probe-half* and *eval-half* using a fixed seed, `SPLIT_SEED=42`. Since both halves are generated from the same deduplicated seeded permutation, they are disjoint at the content level.

The two halves serve different purposes throughout the paper. The probe-half is used only for the representation analyses in §4 and §5, including probe fitting, $\mathcal{V}$-information estimation, Fisher scores, and PID. The eval-half is used only for benchmark reporting in Table 1 and Table 4. Thus, no probe-half example is used to fine-tune a model or report benchmark performance, and no eval-half example is used to fit an analysis probe.

Training/validation split for fine-tuning. Fine-tuning uses only the dedicated training files from RAGTruth, ClaimVerify, and FactCheck-GPT. We concatenate these files, shuffle them with seed 42, and hold out 5% as an in-training validation slice for logging. This yields approximately $21{,}686$ training rows and $1{,}141$ validation rows. The validation slice is not used for model selection: all reported results use final-step checkpoints. This choice avoids introducing checkpoint-selection differences between the two fine-tuning paradigms.

Length truncation. Both paradigms tokenize examples with the model's native chat template and use left padding. The training-time context window is capped at $2{,}048$ tokens using HuggingFace truncation, with `truncation=True` and `max_length=2048`. Hidden-state extraction for the representation analyses uses a $4{,}096$-token window. Under the Qwen3 tokenizer, more than 99% of training examples fit within $2{,}048$ tokens. For the rare overflowing examples, tokenizer truncation removes tokens from the left, preserving the trailing `Answer:` cue and the readout position used by the CLS head.

Per-dataset statistics. Table 5 summarizes the row counts after deduplication and the per-half sizes used in the experiments. The in-distribution probe pool contains $2{,}870$ rows and the OOD probe pool contains $6{,}506$ rows; the corresponding eval-half sizes differ by at most one row because of odd corpus sizes. Class balance varies substantially across corpora, with $p(Y{=}1)$ ranging from 0.20 on FavaBench to 0.74 on ClaimVerify. We therefore report macro-F_1 rather than accuracy throughout the paper.

Table 5: **Per-corpus row counts.** Training corpora (top) feed fine-tuning under a $95/5$ train/validation split. Test corpora (bottom) are deduplicated by $(\text{premise}, \text{hypothesis})$ fingerprint, then split stratified $50/50$ into a probe-half for static and dynamic analyses and an eval-half for benchmark evaluation. $p(Y{=}1)$ is the positive-class rate measured on the deduplicated corpus.

Corpus	Domain	Role	Raw	Deduped	Probe-half	Eval-half	$p(Y{=}1)$
Training corpora (95% train / 5% in-training validation)							
$\text{RAGTruth}_{\text{train}}$	general	train	14,949	—	—	—	0.554
$\text{ClaimVerify}_{\text{train}}$	general	train	6,341	—	—	—	0.720
$\text{FactCheck-GPT}_{\text{train}}$	general	train	1,537	—	—	—	0.173
Total			22,827	—	—	—	0.487
In-distribution test corpora (probe-half → analysis; eval-half → benchmark)							
$\text{RAGTruth}_{\text{test}}$	general	probe + eval	2,681	2,674	1,337	1,337	0.649
$\text{ClaimVerify}_{\text{test}}$	general	probe + eval	1,517	1,503	751	752	0.741
$\text{FactCheck-GPT}_{\text{test}}$	general	probe + eval	1,566	1,565	782	783	0.240
OOD test corpora (probe-half → analysis; eval-half → benchmark)							
$\text{AggreFact}_{\text{test}}$	summarization	probe + eval	9,421	9,370	4,685	4,685	0.592
$\text{FaithBench}_{\text{test}}$	summarization	probe + eval	459	459	229	230	0.231
$\text{FavaBench}_{\text{test}}$	fine-grained	probe + eval	3,185	3,185	1,592	1,593	0.200
Total (test)			18,829	18,756	9,376	9,380	—

Fine-tuning recipe shared by both paradigms. Both CLS and S2S use the same parameter-efficient fine-tuning recipe. We attach LoRA adapters [Hu et al., 2022] with rank $r = 32$, scaling $\alpha = 64$, and dropout 0.05 to all seven projection matrices in every transformer block: `q_proj`, `k_proj`, `v_proj`, `o_proj`, `gate_proj`, `up_proj`, and `down_proj`. The base model weights remain frozen. AdamW optimizes the adapters for 3 epochs with learning rate 10^{-4}, weight decay 0.01, and a linear warmup over the first 3% of steps. Each run uses an effective batch size of 8 sequences, groups sequences by length to reduce padding, and shuffles training examples with seed 42.

Each run uses a single H200 GPU. For a fixed backbone, the two paradigms see the same examples in the same order for the same number of optimization steps. The only intended difference is the readout and loss defined in §2: CLS trains a binary classifier head on the final hidden state, whereas S2S trains the pretrained language-modeling head to generate the verdict token. This matched setup isolates the objective and readout choice as the experimental contrast.

Heads and decoding. For CLS, we replace HuggingFace's default score layer with a bias-free linear head $w \in \mathbb{R}^{D\times 2}$ applied in fp32 to the hidden state of the last input token at layer L. Left padding ensures that this readout position always corresponds to the final column. Predictions use

$$\hat{Y} = \arg\max_{y} (w^{\top} H_L)_y.$$

For S2S, we keep the pretrained language-modeling head and train with `trl`'s `SFTTrainer`. Prompt tokens are masked to -100, so the loss applies only to the `Yes`/`No` verdict. At test time, we greedily decode and parse the first verdict token after the `Answer:` cue. Both paradigms receive the same chat-formatted NLI prompt, so their inputs are byte-identical up to the readout point.

Prompt template. Each $(\text{premise}, \text{hypothesis})$ pair is rendered as a single user-turn chat message with no system prompt:

```
Can the hypothesis be inferred from the premise? Answer using Yes or No only.

Premise: {premise}
Hypothesis: {hypothesis}
Answer:
```

The rendered string is then passed through each model's native chat template. The CLS and S2S variants share this rendering byte-for-byte; they differ only after the `Answer:` cue, where CLS reads the hidden state at the trailing token and S2S predicts the next `Yes`/`No` token.

Reporting. We report macro-F_1 on each of the six test corpora and their unweighted average. Every backbone–paradigm cell in Table 1 and every exit in Table 4 except L15 is fine-tuned with three seeds (42, 43, 44) under an otherwise identical recipe and evaluated at its final-step checkpoint; we report the mean and population standard deviation and compute gaps and early-exit deltas per seed before averaging. The static and dynamic analyses refit their probes with three independent seeds and report paired bootstrap 95% confidence intervals. As a result, the benchmark tables evaluate held-out examples that are never used for probe fitting, while the layer-wise analysis reports uncertainty for the statistics that support the mechanistic claims.

D Extended Methodology: Analysis Tools

This section gives the formal definitions and implementation details for the representation-analysis tools used in the paper. The common design principle is to keep the data split, readout position, probe protocol, and calibration procedure fixed whenever we compare layers, checkpoints, model scales, or fine-tuning objectives.

D.1 Visualization Details

This appendix describes the implementation of Fig. 1. For each model scale, we construct a two-dimensional coordinate system from the corresponding CLS-fine-tuned model and then reuse the same axes for the pretrained BASE, S2S-fine-tuned, and CLS-fine-tuned variants. This shared coordinate system makes the panels comparable: visual differences reflect changes in the hidden states, not changes in the projection chosen for each paradigm.

Projection axes. For each scale, we load the trained CLS score head $W_{\text{CLS}} \in \mathbb{R}^{2\times D}$ and define the discriminative axis as

$$w^{\text{disc}} = \frac{W_{\text{CLS}}[1] - W_{\text{CLS}}[0]}{\|W_{\text{CLS}}[1] - W_{\text{CLS}}[0]\|_2}. \tag{5}$$

This axis is the unit normal of the linear decision hyperplane learned by the CLS head. We then extract last-layer hidden states from the same CLS-fine-tuned model, remove the component along w^{disc}, and compute the leading principal direction of the residual:

$$\tilde{H} = H - (Hw^{\text{disc}})w^{\text{disc}\top}, \qquad w^{\perp} = \text{PC}_1(\tilde{H}). \tag{6}$$

We normalize $w^{\perp}$ to unit length. In implementation, $w^{\perp}$ is the top right singular vector of the centered residual matrix. Thus, the horizontal axis is aligned with the trained CLS decision direction, while the vertical axis captures the largest remaining direction of variation orthogonal to it.

Shared coordinates across paradigms. For a fixed scale, every example from each paradigm is projected onto the same two axes:

$$z_1(x) = H_L(x)^{\top} w^{\text{disc}}, \qquad z_2(x) = H_L(x)^{\top} w^{\perp}. \tag{7}$$

The axes are shared across BASE, S2S, and CLS. This prevents each paradigm from selecting its own favorable projection and makes the three panels a direct geometric comparison. The 4B and 8B models use separately constructed axes because their hidden dimensions differ.

Density rendering. For each class and panel, we estimate a two-dimensional Gaussian kernel density over (z_1, z_2) with bandwidth factor 0.30. The density is evaluated on a 140×140 grid and each class density is normalized by its own maximum before contour plotting. This normalization makes the figure suitable for comparing class locations, overlap, and shape, but not for comparing absolute density magnitudes across classes or panels.

Axis limits and figure layout. Within each scale, we compute axis limits from the union of projected points across the three variants and apply the same limits to all three panels. Thus, differences among BASE, S2S, and CLS reflect geometry rather than rescaling. The main figure arranges the six panels as

$$\text{4B: Base, S2S, CLS} \quad \text{and} \quad \text{8B: Base, S2S, CLS.}$$

All panels use square aspect ratio, shared class colors, and a shared legend.

D.2 Direction-aligned Fisher score

Goal. For each layer ℓ of each model, we compute a scalar measure of how separable the two classes are along the direction used by the trained CLS classifier. This produces a geometry-aligned diagnostic that is directly tied to Fig. 1: the visualization shows class geometry in the CLS decision coordinate system, and the Fisher score quantifies class separation along that same decision direction.

Inputs. We use the held-out probe-half defined in Appendix C, which is disjoint from both the fine-tuning data and the benchmark eval-half. For every model

$$m \in \{\text{BASE}, \text{S2S}, \text{CLS}\} \times \{\text{4B}, \text{8B}\},$$

we cache the per-layer block-output hidden states at the final answer position,

$$H^{(m)} \in \mathbb{R}^{N \times (L+1) \times D},$$

where N is the number of probe examples, $L = 36$ for both Qwen3-4B and Qwen3-8B, and D is the hidden width ($D_{\text{4B}} = 2560$, $D_{\text{8B}} = 4096$). We discard the embedding output at index 0 and index transformer blocks as $\ell = 1, \ldots, L$, matching the layer convention used by the linear probes in Appendix D.3. Labels are $y_i \in \{0, 1\}$, with $0 = \text{FAITHFUL}$ and $1 = \text{HALLUCINATED}$. All arithmetic for this diagnostic is performed in `float64`.

Step 1: extract the decision normal. For each scale, we load the final-step CLS adapter checkpoint and read its score head

$$W_{\text{cls}} \in \mathbb{R}^{2 \times D}, \qquad W_{\text{cls}} = \begin{bmatrix} w_0^\top \\ w_1^\top \end{bmatrix}.$$

The logit margin between the hallucinated and faithful classes is the linear functional $h \mapsto (w_1 - w_0)^\top h$, so the separating hyperplane is normal to $w_1 - w_0$. We define the unit decision normal as

$$w^{\text{disc}} = \frac{w_1 - w_0}{\|w_1 - w_0\|_2} \in \mathbb{R}^D. \tag{8}$$

This vector is computed once per scale from the trained CLS head and then shared across BASE, S2S, and CLS. This is the only learned direction used by the metric.

Step 2: project hidden states. For each model and layer, we compute scalar projections onto the shared decision direction:

$$z_{i,\ell}^{(m)} = \langle w^{\text{disc}}, H_{i,\ell,\cdot}^{(m)} \rangle, \qquad i = 1, \ldots, N. \tag{9}$$

Because $\|w^{\text{disc}}\|_2 = 1$, the projection has the scale of a single hidden-coordinate direction. Differences along this axis correspond to changes in the CLS logit margin up to the multiplicative factor $\|w_1 - w_0\|_2$.

Step 3: compute class statistics in one dimension. Let $\mathcal{I}_+ = \{i : y_i = 1\}$ and $\mathcal{I}_- = \{i : y_i = 0\}$. For each model and layer, we compute the projected class means

$$\mu^+_{m,\ell} = \frac{1}{|\mathcal{I}_+|} \sum_{i \in \mathcal{I}_+} z^{(m)}_{i,\ell}, \qquad \mu^-_{m,\ell} = \frac{1}{|\mathcal{I}_-|} \sum_{i \in \mathcal{I}_-} z^{(m)}_{i,\ell},$$

the signed mean gap

$$\Delta_{m,\ell} = \mu^+_{m,\ell} - \mu^-_{m,\ell},$$

and the unbiased projected variances

$$\sigma^{2,+}_{m,\ell} = \frac{1}{|\mathcal{I}_+| - 1} \sum_{i \in \mathcal{I}_+} \left(z^{(m)}_{i,\ell} - \mu^+_{m,\ell}\right)^2, \qquad \sigma^{2,-}_{m,\ell} = \frac{1}{|\mathcal{I}_-| - 1} \sum_{i \in \mathcal{I}_-} \left(z^{(m)}_{i,\ell} - \mu^-_{m,\ell}\right)^2.$$

Step 4: direction-aligned Fisher score. The per-layer score is

$$F^{(m)}_w(\ell) = \frac{\Delta^2_{m,\ell}}{\sigma^{2,+}_{m,\ell} + \sigma^{2,-}_{m,\ell}} = \frac{\left(w^{\text{disc}\,\top}(\mu^+_{m,\ell} - \mu^-_{m,\ell})\right)^2}{w^{\text{disc}\,\top}(\Sigma^+_{m,\ell} + \Sigma^-_{m,\ell})w^{\text{disc}}}. \tag{10}$$

The second expression writes the same quantity in terms of the original D-dimensional class means and covariances. In implementation, we use the 1D form in Eq. 10, which requires only an $O(NL)$ projection pass and an $O(NL)$ statistics pass. We do not normalize hidden states, smooth across layers, or shrink covariance estimates. If the denominator is exactly zero, which did not occur in our runs, we set the score to zero.

Step 5: best layer. For each model, we report the peak layer and peak score:

$$\ell^\star_m = \operatorname*{arg\,max}_{\ell \in \{1,\ldots,L\}} F^{(m)}_w(\ell), \qquad F^{\star,m}_w = F^{(m)}_w(\ell^\star_m).$$

This identifies the layer at which class separation along the CLS decision direction is largest.

Why use direction-aligned rather than trace-form Fisher. A conventional trace-form Fisher score would be

$$F^{(m)}_{\text{tr}}(\ell) = \frac{\|\mu^+_{m,\ell} - \mu^-_{m,\ell}\|_2^2}{\text{tr}(\Sigma^+_{m,\ell} + \Sigma^-_{m,\ell})}.$$

This form aggregates over all hidden dimensions. That aggregation creates two problems for our purpose. First, if CLS fine-tuning concentrates class information into a low-dimensional decision subspace while leaving high-variance task-irrelevant directions intact, the trace denominator dilutes the very signal we want to measure. Second, the trace denominator scales with hidden width under roughly isotropic variance, making raw trace-form scores difficult to compare across Qwen3-4B and Qwen3-8B without an additional normalization convention.

The direction-aligned score avoids both issues by measuring separation after projecting to the same 1D decision coordinate. It should therefore be read as a CLS-*discoverable* separability diagnostic: it measures separation along the direction that the trained CLS head actually uses. This is the appropriate convention for Sec. 4.1, whose claim concerns the geometry induced by CLS fine-tuning and read out by the CLS classifier. Complementary diagnostics in Appendix D.3 do not restrict the readout to this single direction.

Cross-paradigm convention. For BASE and S2S, w^{disc} is borrowed from the same scale's trained CLS head rather than relearned. This is essential for interpretation: $F^{\text{BASE}}_w(\ell)$, $F^{\text{S2S}}_w(\ell)$, and $F^{\text{CLS}}_w(\ell)$ are measurements along a common axis in the same representation space. Differences among the curves therefore reflect differences in the hidden states, not differences in the direction chosen by the metric. A paradigm-specific LDA direction, by contrast, would give each paradigm its own most favorable axis and would no longer support a direct geometric comparison.

Numerical and reproducibility notes. All projections, means, and variances are computed on CPU in `float64`. Sample variances use Bessel's correction. The `score.linear.weight` parameter has shape $(2, D)$, with row 0 corresponding to the FAITHFUL logit and row 1 corresponding to the HALLUCINATED logit. Therefore $w_1 - w_0$ points from FAITHFUL toward HALLUCINATED, and positive $\Delta_{m,\ell}$ indicates separation in the same orientation as the trained head. We save the arrays $\{F_w^{(m)}(\ell)\}_\ell$, $\{\Delta_{m,\ell}\}_\ell$, and $\{\sigma^{2,+}_{m,\ell} + \sigma^{2,-}_{m,\ell}\}_\ell$ as `.npy` files, together with a JSON summary recording the source CLS checkpoint, $\ell^\star_m$, $F_w^{\star,m}$, and the metric definition.

D.3 Layer-wise linear probe and $\mathcal{V}$-information

Goal. For each layer ℓ of each model, we estimate how much of the binary label $Y \in \{0, 1\}$ can be recovered by a linear readout from the last-token hidden state $H_\ell \in \mathbb{R}^D$. We instantiate this quantity with $\mathcal{V}$-information [Xu et al.] using a temperature-calibrated linear probe family. The resulting curve $\{I_\mathcal{V}(\ell)\}_{\ell=1}^L$ is the layer-wise decodability quantity plotted in Fig. 3 and recomputed over checkpoints in Sec. 5.3.

$\mathcal{V}$-information identity. $\mathcal{V}$-information measures the label information accessible to a restricted predictor family $\mathcal{V} \subseteq \{f : \mathbb{R}^D \to \Delta^{|\mathcal{Y}|}\}$:

$$I_\mathcal{V}(H_\ell \to Y) = H_\mathcal{V}(Y) - H_\mathcal{V}(Y \mid H_\ell), \tag{11}$$

where

$$H_\mathcal{V}(Y) = \inf_{f \in \mathcal{V}} \mathbb{E}[-\log f(\varnothing)(Y)]$$

and

$$H_\mathcal{V}(Y \mid H_\ell) = \inf_{f \in \mathcal{V}} \mathbb{E}[-\log f(H_\ell)(Y)].$$

We take $\mathcal{V}$ to be the family of linear probes with bias,

$$\mathcal{V} = \left\{ h \mapsto \mathrm{softmax}(Wh + b) : W \in \mathbb{R}^{2\times D},\ b \in \mathbb{R}^2 \right\}.$$

The bias term ensures that $\mathcal{V}$ contains constant predictors. Thus the unconditional term is the empirical label entropy, $H_\mathcal{V}(Y) = H(Y)$, and

$$I_\mathcal{V}(H_\ell \to Y) = H(Y) - \widehat{\mathrm{CE}}_\mathcal{V}(\ell), \tag{12}$$

where $\widehat{\mathrm{CE}}_\mathcal{V}(\ell)$ is the calibrated test cross-entropy of the best linear probe at layer ℓ. This quantity is bounded above by $H(Y) \le \log 2$ nats for binary labels. In finite samples, raw probe cross-entropy can exceed $H(Y)$ because of overconfidence or optimization noise, so we report the clipped estimate in Eq. 13 and store the unclipped value as a diagnostic.

Inputs. We use the held-out probe-half pooled across the six test corpora, comprising approximately 9,376 examples. This pool is disjoint from both the fine-tuning training corpus and the benchmark eval-half described in Appendix C. We optionally cap the pool at $N = 10{,}000$ using stratified sampling with `seed=42`; in our experiments the cap is non-binding. For every model m in the analysis registry, we cache per-layer last-token hidden states at the final answer position:

$$H^{(m)} \in \mathbb{R}^{N\times(L+1)\times D}, \qquad L = 36, \quad D_{4\mathrm{B}} = 2560, \quad D_{8\mathrm{B}} = 4096.$$

We discard the embedding output at index 0 and index layers as $\ell = 1, \ldots, L$. We also cache the post-attention residual stream after each block's self-attention and before its MLP, yielding a parallel tensor of shape (N, L, D). Unless otherwise stated, all probe results in the paper use block-output hidden states.

Stratified train/validation/test split. A single stratified split with `seed=42` partitions the probe pool into 70% train, 15% validation, and 15% test. The split indices are saved and reused across probe families, layers, model variants, and checkpoints. This makes $I_\mathcal{V}$ values comparable across layers, paradigms, scales, and training steps because all evaluations use the same held-out examples. Stratification preserves the empirical class prior in each split, so $H(Y)$ is a fixed constant for the corresponding probe dataset.

Probe head and training procedure. At each layer ℓ, the probe is

$$f_\ell(h) = \mathrm{softmax}(W_\ell h + b_\ell), \qquad W_\ell \in \mathbb{R}^{2\times D}, \quad b_\ell \in \mathbb{R}^2.$$

The head is initialized with PyTorch's default `nn.Linear` initialization. We train with Adam [Kingma and Ba, 2014], full-batch gradient descent, cross-entropy loss, weight decay 0, and a budget of 100 epochs. The learning rate is selected per layer from

$$\{10^{-4},\ 5\times 10^{-4},\ 10^{-3}\}$$

by validation macro-F_1. For each candidate learning rate, we record the epoch with the highest validation F_1; we then keep the head state from the candidate whose best validation F_1 is largest. Test cross-entropy and per-sample probabilities are computed from this selected head.

We do not apply representation normalization, layer-wise rescaling, or ℓ_2 regularization. Thus, the probe family matches the unconstrained linear-with-bias family in Eq. 11. We repeat the full procedure under three probe initialization seeds $\{42, 123, 456\}$, while keeping the train/validation/test split fixed. The shaded bands in Fig. 3 report mean and standard deviation over these seeds. For downstream PID and trajectory analyses, we use the median seed according to average F_1 over layers.

Temperature calibration. Upper-layer probes can become overconfident, producing cross-entropies that are poorly aligned with their classification accuracy. We therefore apply post-hoc temperature scaling [Guo et al., 2017]. For each saved layer head, we hold the probe weights fixed and fit a scalar temperature $T_\ell > 0$ on validation logits:

$$T_\ell^\star = \arg\min_{T>0} \mathbb{E}_{(h,y)\in\mathrm{val}} \left[-\log \mathrm{softmax}\left(\frac{1}{T} f_\ell^{\mathrm{logit}}(h)\right)_y \right].$$

We optimize this objective with LBFGS and strong-Wolfe line search, then evaluate calibrated cross-entropy on the test split:

$$\widehat{\mathrm{CE}}_{\mathcal{V}}(\ell) = \mathbb{E}_{(h,y)\in\mathrm{test}} \left[-\log \mathrm{softmax}\left(\frac{1}{T_\ell^\star} f_\ell^{\mathrm{logit}}(h)\right)_y \right].$$

The temperature is fit only on validation examples and then applied to test examples; test data never enters either head training or temperature fitting. We feed the calibrated CE into Eq. 12 and report

$$\widehat{I}_{\mathcal{V}}(\ell) = \max\Big(0,\ H(Y) - \widehat{\mathrm{CE}}_{\mathcal{V}}(\ell)\Big). \tag{13}$$

The unclipped value is saved separately to track overconfidence and optimization failure cases.

Trajectory variant. For Sec. 5.3 and Fig. 4, we re-extract last-token hidden states at every saved CLS checkpoint $t \in \{0, 150, 300, \dots\}$, using the same examples, split, prompt formatting, and answer-position convention. We refit the per-layer linear probes from scratch at each checkpoint, recalibrate them, and assemble

$$I_{\mathcal{V}}(\ell, t) \in \mathbb{R}^{L\times T}.$$

Because the data split is fixed across checkpoints, changes along a row of this matrix reflect changes in the model representation rather than changes in the probe dataset.

Joint and SwiGLU variants used by PID. The PID analysis in Sec. 5.2 requires a joint probe

$$\widehat{I}_{\mathcal{V}}(H_\ell \oplus H_p \to Y)$$

from a family $\mathcal{V}_{\mathrm{joint}}$ that contains the marginal probe families used on H_ℓ and H_p. Otherwise, monotonicity comparisons between marginal and joint $\mathcal{V}$-information would be hard to interpret. Our fusion head uses an asymmetric SwiGLU–Linear architecture: a SwiGLU expansion on the candidate-layer branch and a Linear branch on the plateau-layer branch. To match that architecture, we train two additional marginal probe families per layer under the same 70/15/15 split, learning-rate grid, and calibration protocol:

- a SwiGLU probe

$$\mathcal{V}_\ell = \{h \mapsto \mathrm{Linear}_2\left(\mathrm{Dropout}(\mathrm{SwiGLU}(h))\right)\},$$

  with hidden width matched to the fusion head, used for the candidate-layer marginal H_ℓ;

- the Linear-with-bias probe family above, used for the plateau-layer marginal H_p.

The PID inputs are the calibrated quantities

$$I_{\mathcal{V}_\ell}(H_\ell; Y), \qquad I_{\mathcal{V}_p}(H_p; Y), \qquad I_{\mathcal{V}_{\text{joint}}}(H_\ell, H_p; Y),$$

all evaluated on the same test split. The nonlinear probe families use AdamW, weight decay 5×10^{-3}, and 150 epochs to absorb their larger parameter count; all other choices match the linear-probe protocol.

Reported quantities. For each model and layer, the pipeline saves: raw and calibrated test CE, $\widehat{I}_{\mathcal{V}}(\ell)$ in nats, macro-F_1, accuracy, per-sample softmax probabilities, the selected learning rate, the selected epoch, the fitted temperature $T_\ell^\star$, and the probe head state. Multi-seed mean and standard deviation are stored alongside the median-seed snapshot used by downstream analyses.

Reproducibility notes. Probe training runs on a single GPU in `float32`; temperature fitting runs on CPU in `float64`. The 70/15/15 split is keyed by `seed=42` and is invariant to probe family, probe initialization seed, model checkpoint, and representation choice. This fixed split is what makes the layer-wise, cross-paradigm, cross-scale, and temporal comparisons numerically meaningful.

D.4 Upper-layer saturation and plateau interchangeability

This appendix details the plateau test reported in Sec. 5.1. We run this test on the CLS-fine-tuned models because the resulting plateau boundary is the input to the early-exit analysis in Sec. 6.

Goal. We test whether an intermediate layer of a CLS-fine-tuned model carries as much linearly decodable task information as the final layer. Operationally, a layer is considered interchangeable with the final layer if a linear probe trained on that layer matches the final-layer probe in macro-F_1 within a paired bootstrap confidence interval. The smallest such layer, denoted $L^\star$, marks the start of the upper plateau: above this point, deeper layers no longer improve the linear readout on the same held-out examples.

Information-gain localization for the search interval. Before running the bootstrap test, we localize where fine-tuning adds linearly decodable label information relative to the pretrained backbone. For a CLS-fine-tuned model f and its pretrained backbone f_0, both evaluated with the calibrated linear-probe protocol of Appendix D.3, we define

$$\Delta I_{\mathcal{V}}(\ell) = \max\Big(\widehat{I}_{\mathcal{V}}^{f}(\ell) - \widehat{I}_{\mathcal{V}}^{f_0}(\ell), 0\Big). \tag{14}$$

We mark a layer as having measurable gain if

$$\Delta I_{\mathcal{V}}(\ell) \geq \tau \cdot \max_{\ell'} \Delta I_{\mathcal{V}}(\ell'), \qquad \tau = 0.10.$$

On both Qwen3-4B and Qwen3-8B, this rule identifies a signal-forming region starting around L16, with rapid growth through the upper-middle layers and near-final values in the upper layers. We therefore use the gain region beginning at L16 as the candidate interval for the bootstrap plateau test.

Probe head and training procedure. At every candidate layer ℓ, we train a layer-specific linear probe

$$f_\ell(h) = \text{softmax}(W_\ell h + b_\ell), \qquad W_\ell \in \mathbb{R}^{2 \times D}, \quad b_\ell \in \mathbb{R}^2,$$

using the same linear-with-bias family as Appendix D.3. Training uses full-batch Adam, cross-entropy loss, weight decay 0, 100 epochs, and learning-rate selection from

$$\{10^{-4},\ 5 \times 10^{-4},\ 10^{-3}\}$$

by validation macro-F_1. We repeat the full training procedure under seeds $\{42, 123, 456\}$, changing probe initialization but not the data split. The bootstrap confidence interval is computed from the per-sample predictions of the median seed, so the interval reflects sample-level uncertainty. Across-seed variation is reported separately in Fig. 3.

Stratified train/validation/test split. The plateau test uses the same 70/15/15 split as Appendix D.3, jointly stratified by label and in-domain dataset. Joint stratification ensures that each label–dataset cell is represented in train, validation, and test with deterministic counts. The same split indices are reused for every layer, so macro-F_1 differences across layers are evaluated on identical examples.

Macro-F_1 on the held-out split. We report macro-F_1 on the held-out test portion of the in-domain probe split. Macro-F_1 averages the per-class F_1 scores for FAITHFUL and HALLUCINATED, weighting both labels equally. Predictions are obtained by argmax on calibrated softmax probabilities; no per-layer threshold tuning is used.

Paired bootstrap on the F_1 difference. For each candidate layer ℓ, we form the macro-F_1 gap to the final layer:

$$\Delta_{\text{final}}(\ell) = F_1(\ell) - F_1(L_{\max}). \tag{15}$$

We compute a 95% confidence interval by paired bootstrap with $B = 1000$ replicates. Each replicate draws one resample index vector over the held-out examples and applies the same vector to both the layer-ℓ and final-layer prediction streams. Macro-F_1 is recomputed for both layers on the resampled examples, and the bootstrap replicate records the difference. The confidence interval is the empirical $[2.5, 97.5]$ percentile range of these paired differences. Pairing preserves sample-level correlation and directly tests whether two probes differ on the same examples.

The $L^\star$ identification rule. We define the saturation layer as the earliest layer whose F_1 gap to the final layer is statistically indistinguishable from zero:

$$L^\star = \min \left\{ \ell \in \{1, \ldots, L_{\max} - 1\} : 0 \in \widehat{\text{CI}}_{0.95}\big(\Delta_{\text{final}}(\ell)\big) \right\}. \tag{16}$$

We use the symmetric "CI contains zero" criterion because either direction of a statistically reliable gap is informative: an intermediate layer that loses to the final layer would not yet be plateau-equivalent, while an intermediate layer that beats the final layer would also contradict a simple monotone-depth interpretation. The body reports $\Delta_{\text{final}}(L^\star)$ and its 95% CI as the plateau-membership statistic.

Reproducibility notes. Probe training runs in single-precision GPU arithmetic; the bootstrap loop runs on CPU in double precision. Bootstrap resampling uses a fixed seed offset by layer index. The 70/15/15 split is keyed by the dataset seed and is invariant to probe initialization seed and model checkpoint. The plateau results use $B = 1000$ replicates and the median of the three probe seeds $\{42, 123, 456\}$ as the per-sample prediction source.

D.5 Joint probing and partial information decomposition

This appendix describes the joint-probing and PID analysis reported in Sec. 5.2 and Table 3.

Goal and scope. Sec. 5.1 shows that, from $L^\star$ onward, the upper layers provide interchangeable single-layer readouts for a linear classifier. That result does not determine whether plateau layers contain the same task information or different information that happens to yield the same F_1. This distinction matters for deployment. If two layers carry complementary evidence, a fusion readout should improve over either layer alone. If they carry redundant evidence, fusion should add little, and truncating above the plateau onset should be safe. The PID analysis tests this distinction in calibrated information units; the fair OOD fusion test then asks whether the same conclusion holds for downstream F_1 under a deployment-style training protocol.

PID formal setup. Partial information decomposition [Williams and Beer, 2010] decomposes the joint information of two sources X_1, X_2 about a target Y into four atoms:

$$I(X_1, X_2; Y) = R + U_1 + U_2 + S, \tag{17}$$

where R is redundant information, U_i is information unique to source X_i, and S is synergistic information available only from the joint observation. The marginal mutual informations satisfy

$$I(X_1; Y) = R + U_1, \tag{18}$$

$$I(X_2; Y) = R + U_2. \tag{19}$$

Different PID definitions specify redundancy differently, and therefore produce different atom values. Our goal is not to adjudicate among PID axiomatizations, but to test whether the joint readout extracts information beyond the better single-layer readout.

Min-MI bound used here. We use a min-MI decomposition over calibrated $\mathcal{V}$-information quantities. Let $X_1 = H_\ell$ be a candidate layer, $X_2 = H_p$ be a plateau layer, and Y be the binary label. We define

$$\widehat{R} = \min\big(I_{\mathcal{V}_\ell}(H_\ell; Y), I_{\mathcal{V}_p}(H_p; Y)\big)\,, \tag{20}$$

$$\widehat{UI}_\ell = \max\big(0,\, I_{\mathcal{V}_\ell}(H_\ell; Y) - I_{\mathcal{V}_p}(H_p; Y)\big)\,, \tag{21}$$

$$\widehat{UI}_p = \max\big(0,\, I_{\mathcal{V}_p}(H_p; Y) - I_{\mathcal{V}_\ell}(H_\ell; Y)\big)\,, \tag{22}$$

$$\widehat{\mathcal{S}} = I_{\mathcal{V}_{\text{joint}}}(H_\ell, H_p; Y) - I_{\mathcal{V}_\ell}(H_\ell; Y) - I_{\mathcal{V}_p}(H_p; Y) + \widehat{R}. \tag{23}$$

The key diagnostic is whether the joint probe exceeds the better marginal probe:

$$\widehat{\mathcal{S}} > 0 \quad \Longleftrightarrow \quad I_{\mathcal{V}_{\text{joint}}}(H_\ell, H_p; Y) > \max\big(I_{\mathcal{V}_\ell}(H_\ell; Y), I_{\mathcal{V}_p}(H_p; Y)\big)\,. \tag{24}$$

This equivalence is a statement about the calibrated probes themselves and does not depend on choosing a particular exact PID solver. We also report the Practical Sufficiency Gap,

$$\mathrm{PSG}(\ell) = I_{\mathcal{V}_{\text{joint}}}(H_\ell, H_p; Y) - I_{\mathcal{V}_p}(H_p; Y) = \widehat{UI}_\ell + \widehat{\mathcal{S}},$$

which measures how much the joint readout improves over the plateau layer alone.

Probe-family compatibility. The marginal and joint probe families must be compatible: if the joint family cannot express the marginal probes, then comparisons such as

$$I_{\mathcal{V}_{\text{joint}}} \geq \max(I_{\mathcal{V}_\ell}, I_{\mathcal{V}_p})$$

would not be guaranteed by model-class inclusion. Our joint probe is an asymmetric SwiGLU–Linear fusion head: a SwiGLU branch processes H_ℓ, a Linear branch processes H_p, the two branches are concatenated, and a small fusion MLP produces logits. The matching marginal families are:

- $\mathcal{V}_\ell$: a SwiGLU $\to$ Dropout $\to$ Linear$_2$ probe trained on H_ℓ alone;
- $\mathcal{V}_p$: the Linear-with-bias probe trained on H_p alone;
- $\mathcal{V}_{\text{joint}}$: the asymmetric fusion head trained on (H_ℓ, H_p).

The marginal probes embed into the joint family through the standard single-branch construction: one branch can be set to a constant contribution while the other branch carries the active marginal readout. All three quantities entering Eqs. 20–23 use the same split and the same temperature-calibration protocol from Appendix D.3.

Per-layer landscape and synergy-optimal layer. We compute the PID quantities independently for every candidate layer

$$\ell \in \{1, \ldots, L_{\max} - 1\}$$

against a fixed plateau layer $H_p = H_{L_{\max}}$ by default. This produces per-layer arrays for $\widehat{R}$, $\widehat{UI}_\ell$, $\widehat{UI}_p$, $\widehat{\mathcal{S}}$, and PSG. Table 3 reports the atoms at the synergy-optimal layer

$$\ell^\star = \arg\max_\ell \widehat{\mathcal{S}}(\ell). \tag{25}$$

This is a conservative summary against the redundancy claim: if any candidate layer contains complementary information relative to the plateau, the synergy-optimal layer is where that evidence should be strongest. We also save the full PID landscape and the synergy-optimal, MI-optimal, and PSG-optimal layer indices.

Plateau-layer robustness. By default, the plateau branch uses $L_{\max}$, but the redundancy claim concerns the plateau as a region rather than one final layer. We therefore repeat the joint-probing pipeline while replacing H_p with plateau layers from $L_{\max} - 1$ down to $L^\star$. The PID atoms remain within seed-level variation across this sweep: $\widehat{UI}_\ell$ stays at zero and $\widehat{\mathcal{S}}$ remains small. This rules out the possibility that the result is an artifact of choosing the final layer as the plateau representative.

Fair OOD fusion test. Information-level redundancy does not by itself rule out a practical F_1 gain from a higher-capacity fusion model. We therefore run a separate fair OOD fusion test, reported in the right-most column of Table 3.

Architecture. The deployed fusion model takes $[H_\ell; H_p] \in \mathbb{R}^{2D}$ as input. The candidate branch maps H_ℓ through a SwiGLU $\to$ Dropout $\to$ Linear $\to$ LayerNorm stack into a 256-dimensional bottleneck. The plateau branch maps H_p through Linear $\to$ LayerNorm into the same bottleneck dimension. A gate

$$g \in [0,1]^{256}$$

is computed from

$$[z_\ell;\ z_p;\ |z_\ell - z_p|;\ z_\ell \odot z_p] \in \mathbb{R}^{1024},$$

and the fused representation is

$$z_{\text{fuse}} = z_p + g \odot W_{\text{tr}} z_\ell, \qquad W_{\text{tr}} \in \mathbb{R}^{256\times 256}.$$

A final Linear$(256 \to 2)$ layer emits classification logits. Auxiliary single-branch heads are trained jointly for branch supervision but discarded at inference.

Relation to the analysis-time joint probe. The analysis-time joint probe is designed for interpretability: its marginal families are matched to the PID calculation. The deployed fusion head is designed as a stronger engineering baseline, using residual gating and auxiliary supervision. Both use the same SwiGLU–Linear branch decomposition and the same 256-dimensional bottleneck, but they serve different purposes: the former supports the information decomposition, while the latter tests whether fusion helps in deployment.

Candidate-layer choice. For each cell, the deployed fusion model uses the candidate layer with the largest analysis-time synergy, $\ell^\star$. This gives fusion the most favorable layer according to the PID landscape. In Table 3, this corresponds to L18 and L12 for the 4B and 8B CLS models, L26 for the BASE controls at both scales, and L21 and L14 for the 4B and 8B S2S models.

Training data and evaluation. The fusion model is trained on the same in-domain training corpus used by the single-layer baseline: 22,827 examples from RAGTruth, ClaimVerify, and FactCheck-GPT, with the same 95/5 train/validation split. It uses the same LoRA-r32 backbone adaptation, optimizer family, learning rate, and number of epochs. Evaluation uses the six-corpus eval-half: RAGTruth, ClaimVerify, FactCheck-GPT, AggreFact, FaithBench, and FavaBench. We do not use OOD adaptation, dataset re-weighting, or per-corpus threshold tuning. Thus, any positive fusion gain must come from the additional candidate-layer branch rather than from extra access to the evaluation distribution.

Paired-bootstrap significance for ΔF_1. For each paradigm–scale cell in Table 3, we align per-sample predictions from the single-layer baseline and deployed fusion model by dataset and example index. The point estimate is the macro-averaged difference over the six evaluation datasets:

$$\widehat{\Delta F_1} = \frac{1}{6}\sum_{d=1}^{6} \left[{F_1}^d(\text{fusion}) - {F_1}^d(\text{last}) \right]. \tag{26}$$

For the 95% CI, we use a dataset-stratified paired bootstrap with $B = 10{,}000$ replicates and seed 42. Each dataset is resampled independently, preserving equal dataset weight in the final average. Within a dataset, the same resample indices are applied to the fusion and baseline prediction streams. This preserves sample-level correlation while preventing a small dataset such as FaithBench from dominating the aggregate uncertainty. We report the percentile confidence interval and a two-sided bootstrap p-value,

$$p = 2\min\left(\Pr[\Delta_b \le 0], \Pr[\Delta_b \ge 0]\right).$$

Cells with $p \ge 0.05$ are marked as non-significant.

What the fair OOD test rules out. A near-zero or negative ΔF_1 rules out the deployment-relevant version of the complementary-information hypothesis: even when fusion receives the most favorable candidate layer and a stronger gated architecture, it does not produce a generalizable advantage over the last-layer readout under matched training and evaluation conditions. Together with $\widehat{UI}_{\ell^\star} = 0$ and small $\widehat{\mathcal{S}}(\ell^\star)$, this supports the interpretation that the plateau is redundant for the label information needed by the detector.

Reproducibility notes. All V-information quantities in Eqs. 20–23 use the temperature-calibrated estimates from Appendix D.3. The analysis-time joint probes run on a single GPU in `float32`; PID atom computation runs on CPU in `float64`. The fair-fusion bootstrap uses `seed=42` and $B = 10{,}000$ replicates. Saved prediction files reproduce the CIs in Table 3 bit-for-bit.

D.6 Training dynamics of the plateau

This appendix describes the per-checkpoint trajectory analysis reported in Sec. 5.3 and Fig. 4. The analysis motivates the substrate / InfoWindow / upper-plateau partition used by the early-exit rule in Sec. 6.

Goal and scope. Endpoint analyses show that the upper layers of a trained CLS model contain largely redundant linearly decodable label information. They do not reveal how that redundancy emerges during training. Two mechanisms are possible. In the first, the upper layers create the binary signal and earlier layers later align to it. In the second, a middle band forms the signal first, and the upper layers later become redundant carriers of that signal. These mechanisms can produce similar endpoint probe curves, so we inspect the temporal order in which layers acquire decodable label information. The trajectory analysis supports the second mechanism.

Per-checkpoint $\mathcal{V}$-information matrix. At every saved CLS checkpoint

$$t \in \{0, t_1, t_2, \ldots, T\},$$

including $t = 0$ for the pretrained backbone, we extract last-token hidden states

$$H_\ell^{(t)} \in \mathbb{R}^{N \times D}$$

for every layer $\ell \in \{1, \ldots, L\}$ on the same in-domain probe pool used by the static analysis. We then refit the calibrated linear-probe estimate from Appendix D.3:

$$I_{\mathcal{V}}(\ell, t) = \max\Big(0,\ H(Y) - \widehat{\mathrm{CE}}_{\mathcal{V}}\big(H_\ell^{(t)}\big)\Big)\,. \tag{27}$$

This produces the layer-by-time matrix

$$I_{\mathcal{V}}(\cdot, \cdot) \in \mathbb{R}^{L \times T}$$

visualized in Fig. 4. Rows are layers, columns are checkpoints, and color denotes calibrated $\mathcal{V}$-information in nats.

What is held constant across checkpoints. Temporal comparisons are meaningful only if changes in $I_{\mathcal{V}}(\ell, t)$ reflect changes in the model representation. We therefore hold the following fixed across checkpoints:

- *Sample identity.* The same in-domain examples and the same 70/15/15 stratified split are reused at every checkpoint.
- *Probe family.* The linear-with-bias probe family from Appendix D.3 is used throughout.
- *Optimization protocol.* Each per-layer probe is trained from scratch with the same Adam, full-batch, learning-rate-grid, 100-epoch, validation-F_1 selection procedure. We do not warm-start probes from earlier checkpoints.
- *Calibration.* A fresh scalar temperature is fit for each layer and checkpoint on the validation split and applied to the test split.
- *Input processing.* Tokenization, chat template, prompt formatting, and answer-position selection are identical to the static run.

Under this design, the only intended source of temporal variation is the model checkpoint.

Checkpoint cadence. Checkpoints are saved every 150 optimizer steps, including the pretrained baseline at $t = 0$. Across the three-epoch CLS schedule with effective batch size 8, this yields approximately 55 columns per scale. This cadence is dense enough to resolve the staggered onset in the InfoWindow and the more synchronous saturation in the upper plateau. We keep all intermediate checkpoints rather than selecting only validation-best checkpoints, so the trajectory is not biased toward a particular operating point.

Per-layer trajectory summaries. We summarize each layer's trajectory using two quantities.

Total information gain. For each layer,

$$\Delta I_{\mathcal{V}}(\ell) = I_{\mathcal{V}}(\ell, T) - I_{\mathcal{V}}(\ell, 0). \tag{28}$$

This measures how much linearly decodable label information the layer gains over fine-tuning. We mark a layer as having non-trivial gain if

$$\Delta I_{\mathcal{V}}(\ell) \geq \tau \cdot \max_{\ell'} \Delta I_{\mathcal{V}}(\ell'), \qquad \tau = 0.10.$$

Saturation time. For layers that cross the gain threshold, we define

$$t_\ell^{0.9} = \min \left\{ t : I_{\mathcal{V}}(\ell, t) \geq 0.9 \cdot I_{\mathcal{V}}(\ell, T) \right\}. \tag{29}$$

Layers below the gain threshold are not assigned a saturation time because their trajectories are dominated by small fluctuations around the pretrained value. The saturation time is reported in optimizer steps.

Together, $\Delta I_{\mathcal{V}}(\ell)$ measures how much a layer learns, while $t_\ell^{0.9}$ measures when it learns it. The combination distinguishes a layer that forms the signal early from a layer that merely carries it later.

Three regimes from the trajectory matrix. The trajectory matrix yields a stable three-regime partition across Qwen3-4B and Qwen3-8B.

(R1) ***Lower substrate*** **($\ell \in [\mathrm{L1}, \mathrm{L15}]$).** These layers remain below the gain threshold. Their $I_{\mathcal{V}}$ trajectories stay close to their pretrained values, and no $t_\ell^{0.9}$ is assigned. We interpret this region as a lower substrate: fine-tuning reaches these layers, but it does not substantially increase their linear label-decodability.

(R2) ***InfoWindow*** **($\ell \in [\mathrm{L16}, \mathrm{L19}]$).** These are the first layers to exceed the gain threshold. Their saturation times follow a depth-staggered pattern: lower InfoWindow layers reach $t_\ell^{0.9}$ before upper InfoWindow layers. This staggered onset suggests that the binary signal is progressively constructed across this narrow band rather than appearing simultaneously across the network.

(R3) ***Synchronous upper plateau*** **($\ell \in [\mathrm{L20}, L_{\max}]$).** Upper layers also acquire label information, but their saturation times cluster tightly:

$$\max_{\ell \in \text{plateau}} t_\ell^{0.9} - \min_{\ell \in \text{plateau}} t_\ell^{0.9} \leq \tau_{\text{step}}, \qquad \tau_{\text{step}} = 1000. \tag{30}$$

Moreover, these saturation times occur after the InfoWindow has already crossed its own saturation threshold. Thus, the upper plateau is the last region to align to the task signal, not the first.

What the temporal order rules out. If the upper plateau were the source of the binary signal, we would expect its saturation times to precede those of the InfoWindow. Instead, the InfoWindow saturates first and does so in a depth-staggered manner, while the upper plateau saturates later and more synchronously. This temporal pattern supports the interpretation that the InfoWindow forms the task signal and the upper plateau later carries a redundant copy. Combined with the endpoint redundancy evidence from Sec. 5.2, the trajectory analysis identifies the upper plateau as a late carrier rather than the locus of signal formation.

Robustness of the partition. The three-regime split is stable under reasonable perturbations:

- *Probe seed.* Repeating the trajectory under probe seeds $\{42, 123, 456\}$ shifts individual $t_\ell^{0.9}$ values by at most one checkpoint for most layers and does not change the regime boundaries.
- *Gain threshold.* Sweeping $\tau \in [0.05, 0.20]$ moves the upper InfoWindow boundary by at most one layer and keeps the lower boundary at L16.
- *Synchrony threshold.* Tightening τ_{step} from 1000 to 500 optimizer steps still leaves most plateau layers in the synchrony band; loosening it to 2000 does not change the ordering that the InfoWindow saturates before the plateau.
- *Calibration.* Using uncalibrated CE changes the numerical scale of some upper-layer values but preserves the ordering of the regimes. We report calibrated values because they give a cleaner information-scale diagnostic.

Connection to the early-exit rule. The trajectory partition provides the dynamic basis for Sec. 6. The early-exit rule keeps the lower substrate and InfoWindow, where the model constructs the binary signal, and exits at or just above the plateau onset, where additional layers mainly act as redundant carriers. The trajectory analysis supplies what endpoint metrics alone cannot: evidence that removing the synchronous plateau removes late carriers rather than the source of the task signal.

Reproducibility notes. The trajectory matrix uses the same dataset seed, probe initialization seeds $\{42, 123, 456\}$, learning-rate grid, 100-epoch budget, and LBFGS calibration protocol as Appendix D.3. Probe training runs in `float32` on GPU; calibration runs in `float64` on CPU. Saturation times are computed from the median seed's trajectory, with multi-seed mean and standard deviation saved for reference. The pretrained column at $t = 0$ is obtained by applying the same probe protocol to the unmodified pretrained backbone, so Eq. 28 matches the cross-model gain definition in Appendix D.4 for the CLS trajectory.

E Early-Exit Experimental Details

This appendix gives the full design and implementation details for the early-exit experiments reported in Sec. 6 and Table 4. The goal is to make the early-exit numbers reproducible without relying on codebase-specific assumptions.

Goal and scope. For each backbone–exit pair (m, ℓ_{exit}), we train a fresh CLS adapter whose classification head reads from the layer-ℓ_{exit} residual stream instead of the final layer L. All other ingredients are matched to the full-depth CLS baseline: data, prompt, tokenizer, optimizer, LoRA configuration, classification-head architecture, training schedule, and evaluation protocol. Thus, the early-exit comparison isolates the effect of moving the readout depth.

The early-exit configurations evaluated in Table 4 are:

- *Qwen3-4B* ($L = 36$): $\ell_{\text{exit}} \in \{15, 20, 25\}$, covering a pre-InfoWindow exit, the first plateau-aligned exit, and a deeper InfoWindow-plus-margin exit.
- *Qwen3-8B* ($L = 36$): $\ell_{\text{exit}} \in \{15, 20, 25\}$, using the same regime-defined exits as Qwen3-4B.
- *Llama-3.2-3B* ($L = 28$): $\ell_{\text{exit}} \in \{17, 20\}$, corresponding to aggressive $\text{round}(0.60L)$ and conservative $\lceil 0.70L \rceil$ retained-depth rules.
- *Mistral-7B-v0.3* ($L = 32$): $\ell_{\text{exit}} \in \{19, 23\}$, using the same aggressive and conservative retained-depth rules.

This produces ten early-exit fine-tuning runs in total. Each run yields one adapter and one classifier head, evaluated on the same six-corpus eval-half as the full-depth baseline.

Forward pass and readout. Given a tokenized input of length T, the backbone produces residual-stream tensors

$$H_0, H_1, \ldots, H_L \in \mathbb{R}^{T \times D},$$

where H_0 is the embedding output and H_ℓ for $\ell \geq 1$ is the output of transformer block ℓ. With left padding, the final input position is always the last sequence index. The early-exit representation is therefore

$$\mathbf{h}_{\text{cls}}(x) = H_{\ell_{\text{exit}}}(x)[T-1, :] \in \mathbb{R}^D. \tag{31}$$

A bias-free linear head then produces the binary logits:

$$\text{logits}(x) = W_{\text{CLS}} \mathbf{h}_{\text{cls}}(x), \qquad W_{\text{CLS}} \in \mathbb{R}^{2 \times D}. \tag{32}$$

This head has the same architecture as the full-depth CLS head in Appendix C: no bias term, fp32 weights, and HuggingFace-default initialization.

Training-time gradient flow. During fine-tuning, we keep the full backbone instantiated for implementation simplicity. The forward pass still computes layers above ℓ_{exit}, but the loss depends only on $H_{\ell_{\text{exit}}}$. By the chain rule, LoRA parameters in blocks above ℓ_{exit} receive zero gradient. In practice, this means that adapters are inserted in all blocks for configuration parity, but adapters above

the exit layer are not updated. When distributed training is used, `find_unused_parameters=True` prevents errors from these unused parameters. This implementation choice does not change the learned early-exit adapter; a deployment implementation can physically truncate the backbone above ℓ_{exit}.

Compute-saving accounting. The "Saved" column of Table 4 is the fraction of removed transformer blocks, $(L - \ell_{\text{exit}})/L$. During fine-tuning and benchmark evaluation we keep the full forward pass for implementation simplicity; the loss depends only on $H_{\ell_{\text{exit}}}$, so the blocks above the exit receive no gradient and the trained adapter is the same as with a truncated stack. The wall-clock saving of the truncated stack is measured separately in Appendix F.6 and is within a few percent of the layer fraction.

Exit selection uses the probe half only. Three quantities determine the Qwen3 exits: the InfoWindow (L16–L19) and plateau onset (L20) from the trajectory analysis of Appendix D.6, and the readout-equivalence layer $L^\star$ (L22/L23) from Appendix D.4. All three are computed on the probe half, which is disjoint from the eval half (Appendix C). The eval half enters only the final numbers of Table 4; no benchmark result influenced the choice of L15, L20, or L25, and the Llama and Mistral candidates were fixed by the two depth ratios before any result on those models was observed.

Backbone configurations. Table 6 lists the constants needed to reproduce the early-exit runs. Here L is the number of transformer blocks and D is the residual-stream width. The classifier head has shape $2 \times D$.

Table 6: **Per-backbone constants for early-exit runs.** L is the number of transformer blocks; D is the hidden width; "Exits" lists the ℓ_{exit} values evaluated in Table 4.

Backbone	L	D	**Exits** ℓ_{exit}
Qwen3-4B	36	2560	15, 20, 25
Qwen3-8B	36	4096	15, 20, 25
Llama-3.2-3B	28	3072	17, 20
Mistral-7B-v0.3	32	4096	19, 23

LoRA configuration. Early-exit runs use the same LoRA configuration as the full-depth CLS baseline in Appendix C: rank $r = 32$, scaling $\alpha = 64$, dropout 0.05, no bias adaptation, and LoRA modules attached to all seven projection matrices in every transformer block:

`q_proj`, `k_proj`, `v_proj`, `o_proj`, `gate_proj`, `up_proj`, `down_proj`.

The base backbone weights remain frozen. The classification head W_{CLS} is the only non-LoRA trainable component and is registered through PEFT's `modules_to_save`, so its weights are saved with the adapter.

Score-head precision. The classification head runs in fp32 with autocast disabled, while the backbone forward pass runs in bfloat16. This choice follows the full-depth CLS setup and improves early optimization stability. The randomly initialized score head starts with small weights but reads from residual streams that may contain large-coordinate outliers, especially in Mistral-style backbones. Keeping the readout in fp32 reduces quantization error in the logits without changing the backbone precision.

Optimization hyperparameters. Table 7 gives the optimization hyperparameters. Qwen3 and Llama-3.2-3B use the default recipe. Mistral-7B-v0.3 uses the backbone-specific override described below.

Backbone-specific override for Mistral. Mistral-7B-v0.3 required a more conservative optimization recipe. With the default learning rate 10^{-4}, weight decay 0.01, and 3% warmup, the randomly initialized fp32 score head produced extremely large initial losses and unstable optimization. This

Table 7: **Optimization hyperparameters for early-exit runs.** "Default" covers Qwen3-4B, Qwen3-8B, and Llama-3.2-3B. The Mistral column changes only the listed optimization entries. The effective batch size is per-device batch size $\times$ gradient-accumulation steps $\times$ world size; all runs use one GPU, so the effective batch is $2 \times 4 \times 1 = 8$.

Hyperparameter	**Default (Qwen3, Llama)**	**Mistral-7B-v0.3**
Optimizer	AdamW	AdamW
Learning rate	1×10^{-4}	$\mathbf{6 \times 10^{-5}}$
Weight decay	0.01	**0.1**
Warmup ratio (linear)	0.03	**0.05**
LR scheduler	linear decay	linear decay
Number of epochs	3	3
Per-device train batch	2	2
Gradient accumulation steps	4	4
Effective batch (1 GPU)	8	8
Per-device eval batch	4	4
Backbone precision	bfloat16	bfloat16
Score-head precision	fp32, autocast disabled	fp32, autocast disabled
Max sequence length (train)	2048	2048
Truncation policy	left truncation of overlong inputs	left truncation of overlong inputs
Padding side	left	left
Group sequences by length	yes	yes
Random seed	42 (reseeds 43, 44)	42 (reseeds 43, 44)
Save-total-limit	unlimited	3

behavior is consistent with the known presence of large-magnitude activation outliers in some decoder-only backbones. We therefore use a lower learning rate, longer warmup, and stronger weight decay for all Mistral runs:

$$\text{lr} = 6 \times 10^{-5}, \qquad \text{weight decay} = 0.1, \qquad \text{warmup} = 5\%.$$

The override is applied uniformly to Mistral full-depth CLS, full-depth S2S, and both Mistral early-exit runs. Therefore, Mistral early-exit results remain directly comparable to the Mistral full-depth baseline within Table 4. No other backbone uses a special recipe.

Data, prompt, and tokenization. The early-exit runs use the same training corpus, split, prompt, and tokenization as the full-depth CLS baseline.

- *Training data.* We use 22,827 examples from $\text{RAGTruth}_{\text{train}}$, $\text{ClaimVerify}_{\text{train}}$, and $\text{FactCheck-GPT}_{\text{train}}$. The files are concatenated, shuffled with `seed=42`, and split into approximately 21,686 training rows and 1,141 in-training validation rows under the same $95/5$ split used by the baseline.
- *Prompt template.* Each example is rendered as a single user-turn chat message:

  ```
  Can the hypothesis be inferred from the premise?
  Answer using Yes or No only.

  Premise: {premise}
  Hypothesis: {hypothesis}
  Answer:
  ```

  The rendered string is passed through the native chat template of each backbone.
- *Tokenization.* We use each backbone's native tokenizer with left padding and a maximum training length of 2,048 tokens. Overlong inputs are truncated from the left, so the trailing `Answer:` cue and the final-token readout position in Eq. 31 are preserved. The data collator pads each batch to the longest sequence in the batch.

The early-exit setting changes only the residual-stream layer used for classification; it does not change which rows or tokens the model sees.

Schedule and checkpointing. With approximately $21{,}686$ training rows and an effective batch size of 8, three epochs correspond to approximately $8{,}133$ optimizer steps. We save checkpoints and run intermediate evaluation every 150 optimizer steps, giving about 55 snapshots per run. The HuggingFace trainer is configured with validation macro-F_1 for best-checkpoint tracking. However, to match the full-depth CLS reporting protocol in Sec. 3, Table 4 reports the final-step checkpoint rather than the validation-best checkpoint. Intermediate checkpoints are retained for trajectory analyses and do not enter the early-exit benchmark table. For Mistral, we keep only the three most recent checkpoints on disk to control storage; this does not alter the final-step adapter.

Evaluation. Evaluation follows the same readout as training. For each input, we left-pad, run the backbone with hidden states enabled, extract $H_{\ell_{\text{exit}}}[T-1,:]$, apply the trained head in Eq. 32, and take the argmax of the two logits as the predicted label. We report macro-F_1 on the eval-half of each of the six test corpora: RAGTruth, ClaimVerify, FactCheck-GPT, AggreFact, FaithBench, and FavaBench. Eval-half sizes are listed in Appendix C. We do not apply per-corpus threshold tuning, calibration, dataset reweighting, or post-processing.

Hardware. Each early-exit run uses a single NVIDIA H200 GPU with 141 GB HBM3e. The backbone runs in bfloat16. SDPA attention is used by default; FlashAttention-2 is used when available for the corresponding backbone and otherwise the run falls back to SDPA without changing any other configuration. Latency in Table 14 is measured separately on the physically truncated stack (Appendix F.6).

Reproducibility notes. The original runs use `seed=42` for data shuffling, the train/validation split, PyTorch and NumPy initialization, and dataloader worker seeding; the reseeded runs behind Tables 1 and 4 use seeds 43 and 44 for the same quantities and are otherwise identical. Layer indexing follows Appendix D.3: $\ell = 1, \ldots, L$ denotes post-block residual streams, while $\ell = 0$ is the embedding output and is never used as an exit. Tables 6 and 7 list the backbone dimensions and optimizer settings needed to reproduce the runs. A reimplementation with the same backbone IDs, data files, split seed, prompt template, and hyperparameters should reproduce Table 4 up to standard GPU nondeterminism in attention and matrix-multiplication kernels.

F Robustness and Generality Analyses

This appendix tests the results of Sections 3 and 6 against the choices they could depend on: the fine-tuning seed, the S2S readout, the composition of the test corpora, the backbone, and the task. Every run keeps the data, optimizer, LoRA configuration, and schedule of Appendix C and changes one variable at a time.

F.1 Seed variability

Tables 8 and 9 list the individual runs behind the means in Tables 1 and 4. The in-distribution gap is positive in 11 of 12 runs and its standard deviation never exceeds 1.1 points. The out-of-distribution gap varies by an order of magnitude more (standard deviation 10.1 on Llama-3.2-3B and 12.1 on Mistral-7B-v0.3), and the variation comes from single runs. The seed-44 CLS model of Mistral-7B-v0.3 collapses toward the positive class on FavaBench and averages 35.6 on the three OOD corpora, against 63.5 and 53.1 for the other two seeds; the seed-44 CLS model of Llama-3.2-3B does the same (42.4). Both collapses are threshold effects on an imbalanced corpus (Appendix F.3) and leave the in-distribution scores of the same runs intact (77.2 and 80.2). The same two runs produce the large standard deviations of the Mistral and Llama exits in Table 4, because each delta is taken against the same-seed full-depth model.

F.2 Stronger S2S readouts

We take the fine-tuned S2S adapters of Table 1 (seed 42) and replace single-token greedy decoding with three stronger readouts, keeping the adapters fixed. **(a) Likelihood-ratio scoring.** One forward

Table 8: **Individual full-depth runs.** Macro-F_1 (%) and paired differences in percentage points. Within each group, Δ is CLS$-$S2S.

Backbone	Seed	S2S ID	CLS ID	Δ ID	S2S OOD	CLS OOD	Δ OOD	S2S All	CLS All	Δ All
Qwen3-4B	42	78.09	80.24	2.14	52.43	71.27	18.84	65.26	75.75	10.49
Qwen3-4B	43	78.85	82.36	3.51	47.34	70.75	23.41	63.10	76.56	13.46
Qwen3-4B	44	76.52	80.40	3.87	50.54	70.38	19.84	63.53	75.39	11.86
Qwen3-8B	42	77.97	80.33	2.36	66.68	74.07	7.39	72.33	77.20	4.87
Qwen3-8B	43	79.02	81.47	2.46	60.22	68.56	8.34	69.62	75.02	5.40
Qwen3-8B	44	78.12	82.10	3.98	66.54	69.34	2.80	72.33	75.72	3.39
Llama-3.2-3B	42	74.57	78.19	3.61	45.83	60.15	14.32	60.20	69.17	8.97
Llama-3.2-3B	43	76.47	79.04	2.58	48.74	54.36	5.62	62.60	66.70	4.10
Llama-3.2-3B	44	75.29	80.22	4.92	52.61	42.41	-10.20	63.95	61.31	-2.64
Mistral-7B-v0.3	42	77.89	79.91	2.02	57.42	63.45	6.03	67.65	71.68	4.02
Mistral-7B-v0.3	43	78.14	79.82	1.68	53.06	53.06	0.00	65.60	66.44	0.84
Mistral-7B-v0.3	44	77.71	77.17	-0.54	57.72	35.59	-22.13	67.72	56.38	-11.34

Table 9: **Individual early-exit runs.** Macro-F_1 (%) and differences from the same-seed full-depth CLS model. Exit depths are held fixed across seeds.

Backbone	Exit	Seed	ID	OOD	All	Δ ID	Δ OOD	Δ All
Qwen3-4B	L15	42	79.52	52.19	65.85	-0.72	-19.08	-9.90
Qwen3-4B	L20	42	81.14	67.05	74.09	0.91	-4.22	-1.66
Qwen3-4B	L20	43	80.17	50.31	65.24	-2.19	-20.44	-11.32
Qwen3-4B	L20	44	80.72	68.76	74.74	0.33	-1.63	-0.65
Qwen3-4B	L25	42	81.40	72.11	76.76	1.17	0.84	1.00
Qwen3-4B	L25	43	81.01	67.42	74.22	-1.35	-3.33	-2.34
Qwen3-4B	L25	44	80.90	69.37	75.14	0.51	-1.01	-0.25
Qwen3-8B	L15	42	78.57	47.12	62.84	-1.76	-26.96	-14.36
Qwen3-8B	L20	42	79.99	69.39	74.69	-0.33	-4.68	-2.51
Qwen3-8B	L20	43	80.53	68.75	74.64	-0.94	0.19	-0.37
Qwen3-8B	L20	44	80.66	71.16	75.91	-1.44	1.82	0.19
Qwen3-8B	L25	42	80.68	73.00	76.84	0.35	-1.07	-0.36
Qwen3-8B	L25	43	80.76	73.75	77.25	-0.71	5.19	2.24
Qwen3-8B	L25	44	81.31	71.38	76.34	-0.79	2.04	0.62
Llama-3.2-3B	L17	42	80.20	61.71	70.96	2.01	1.57	1.79
Llama-3.2-3B	L17	43	80.21	52.00	66.11	1.17	-2.36	-0.60
Llama-3.2-3B	L17	44	79.69	50.89	65.29	-0.53	8.48	3.98
Llama-3.2-3B	L20	42	79.94	45.63	62.79	1.75	-14.51	-6.38
Llama-3.2-3B	L20	43	79.08	50.93	65.01	0.04	-3.43	-1.70
Llama-3.2-3B	L20	44	80.06	48.50	64.28	-0.16	6.09	2.96
Mistral-7B-v0.3	L19	42	77.88	64.38	71.13	-2.03	0.94	-0.55
Mistral-7B-v0.3	L19	43	80.18	57.28	68.73	0.37	4.22	2.29
Mistral-7B-v0.3	L19	44	75.80	49.05	62.42	-1.37	13.45	6.04
Mistral-7B-v0.3	L23	42	79.18	66.32	72.75	-0.73	2.87	1.07
Mistral-7B-v0.3	L23	43	79.06	57.52	68.29	-0.75	4.46	1.85
Mistral-7B-v0.3	L23	44	79.50	61.07	70.29	2.33	25.48	13.90

pass records the last-position logit of every verdict string that tokenizes to a single token (`Yes/No`, `yes/no`, `YES/NO`, `True/False`, `true/false`, `Correct/Incorrect`, `Right/Wrong`, with and without a leading space). The score is the log of the total probability mass on the positive strings minus the log of the mass on the negative strings (a log-sum-exp over each set), and an example is predicted positive when the score exceeds zero. The string set is fixed in advance and not selected on the eval half. **(b) Calibration.** Temperature scaling is monotone and changes no argmax decision, so we report it only for completeness. Label-shift calibration fits one decision threshold per corpus on the probe half and applies it to the eval half; we apply the same per-corpus fitting to the CLS logit margin so that both paradigms receive the same information. **(c) LM-head adaptation.** We add the unembedding matrix to the trainable parameters (LoRA on all projections plus a full update of `lm_head`) and retrain with the recipe of Appendix C, which frees the `Yes/No` direction that single-token S2S otherwise inherits from pretraining. For Qwen3-4B, whose unembedding is tied to the input embedding, the update unties the two matrices. This variant is read out by generation and verdict parsing, as in Table 1.

Table 10 reports six-corpus averages. Likelihood scoring moves S2S by less than one point. Label-shift calibration raises S2S by 5 to 14 points but raises CLS by 2 to 7 points as well. The adapted LM head changes S2S by at most 3 points (a second Qwen3-8B seed scores 67.9, further below CLS). CLS leads under every readout, by 2.2 to 5.3 points when both paradigms are calibrated and by 4.0 to 10.4 points when neither is.

Table 10: **Alternative S2S readouts and matched threshold calibration.** Six-corpus mean macro-F_1 (%), using fine-tuning seed 42. The verbalizer ensemble is fixed, without selection on the eval half. Per-corpus thresholds are fitted separately for each paradigm on labeled probe-half examples from that corpus. LM-head adaptation uses generation and verdict parsing.

Variant	Qwen3-4B	Qwen3-8B	Llama-3.2-3B	Mistral-7B
S2S, generation (paper protocol)	65.2	72.3	60.2	67.7
(a) S2S, Fixed verbalizer-ensemble LLR	65.3	71.6	59.5	67.7
(c) S2S, Generation + trainable LM head	68.0	71.6	61.7	69.5
(b) S2S, LLR + per-corpus label-shift calibration	74.0	76.9	73.5	75.5
CLS, fixed 0.5 threshold (no tuning)	**75.7**	**77.2**	**69.2**	**71.7**
CLS, same label-shift calibration as (b)	**79.3**	**79.5**	**76.6**	**77.7**
Δ uncalibrated: CLS@0.5 − (a)	+10.4	+5.6	+9.7	+4.0
Δ calibrated: CLS-LS − (b)	+5.3	+2.5	+3.1	+2.2

F.3 FavaBench diagnosis and significance tests

FavaBench is the most imbalanced test corpus, with a positive rate of 0.20, and contributes the largest single-corpus gaps in Table 1. Three explanations are possible for the low S2S scores there: verdicts that fail to parse, collapse to one class, and a miscalibrated threshold. Parsing is not the cause. Every fine-tuned model emits a valid verdict on 100% of FavaBench examples (Table 11). The predicted positive rate of S2S ranges from 31% to 89% against a true rate of 20%, so the loss is a threshold effect on an imbalanced corpus. The same effect reaches CLS: at the fixed 0.5 threshold, one CLS seed each of Llama-3.2-3B and Mistral-7B-v0.3 collapses toward the positive class on FavaBench (Appendix F.1). We therefore base the headline comparison on the in-distribution gap, on the calibrated comparison of Appendix F.2, and on the bootstrap below, and we report the fixed-threshold OOD average as what it is, a quantity that either paradigm can lose on a single seed.

Paired bootstrap. For each backbone we resample eval-half examples with replacement, independently within each corpus and paired between the CLS and S2S prediction streams (10,000 replicates), recompute each corpus's macro-F_1 and the average gap, and report the 95% percentile interval and the two-sided percentile p-value, once over all six corpora and once excluding FavaBench (Table 11). Excluding FavaBench shrinks the seed-42 gap from 4.0 to 10.5 points to 2.2 to 3.6 points, and every interval stays above zero.

F.4 Per-layer probing of the transfer backbones and the readout-swap control

Section 6 fixes the Llama-3.2-3B and Mistral-7B-v0.3 exits from a depth fraction before any analysis of those models. Here we run the per-layer linear probing of Appendix D.3 on both, unchanged: the

Table 11: **FavaBench predictions and sensitivity to excluding the corpus.** All results use fine-tuning seed 42. Left: invalid-verdict rate and predicted class proportions on FavaBench, whose positive-label rate is 20%. Right: paired CLS−S2S differences in mean macro-F_1 points, with 95% percentile intervals from 10,000 within-corpus bootstrap resamples. These intervals describe example-sampling uncertainty for fixed models.

	S2S on FavaBench			Gap, all six		Gap, excl. FavaBench	
Backbone	parse fail	pred. neg / pos	verdict	Δ [95% CI]	p	Δ [95% CI]	p
Qwen3-4B	0.0%	33% / 67%	over-positive	+10.5 [+9.1, +11.8]	<0.001	**+3.3** [+1.8, +4.9]	<0.001
Qwen3-8B	0.0%	69% / 31%	miscalibrated	+4.9 [+3.5, +6.2]	<0.001	**+3.6** [+2.0, +5.1]	<0.001
Llama-3.2-3B	0.0%	11% / 89%	collapse to positive	+9.0 [+7.6, +10.3]	<0.001	**+2.9** [+1.3, +4.4]	<0.001
Mistral-7B-v0.3	0.0%	49% / 51%	miscalibrated	+4.0 [+2.7, +5.3]	<0.001	**+2.2** [+0.7, +3.7]	0.004

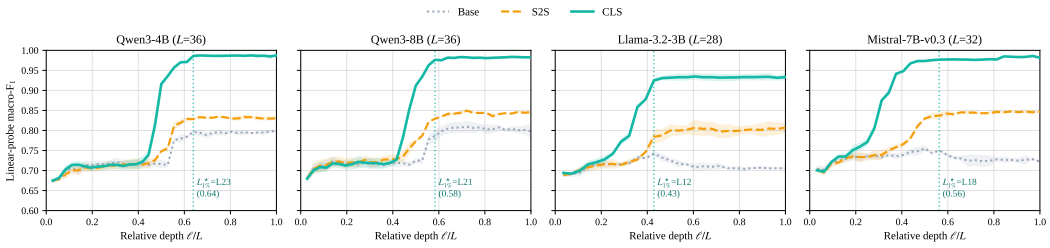

Figure 5: **Per-layer linear-probe F_1 on all four backbones.** BASE, S2S, and CLS probes on the in-domain probe half (mean of three probe seeds, bands show one standard deviation across probe seeds), plotted against relative depth ℓ/L. Every backbone shows the flat pretrained substrate, a sharp rise, and a plateau, with CLS above S2S above BASE; the dotted line marks the CLS saturation layer $L^\star_{1\%}$.

same probe family, the same 70/15/15 split of the probe half, and the same three probe seeds, for the BASE, S2S, and CLS models of Table 1. Figure 5 plots probe F_1 against relative depth for all four backbones and Table 12 lists the best-layer scores.

The curves share one shape. Probe F_1 stays near the pretrained level through the lower 25% to 45% of depth, rises within a few layers, and plateaus, with CLS above S2S above BASE on the plateau for every backbone. As an inference-only summary of where the rise ends we report $L^\star_{1\%}$, the earliest layer within 0.01 F_1 of the best layer. It is layer 12 of 28 for Llama-3.2-3B (0.43 of depth), 18 of 32 for Mistral-7B-v0.3 (0.56), and 23 and 21 of 36 for Qwen3-4B and 8B (0.64 and 0.58). $L^\star_{1\%}$ is a descriptive proxy for the bootstrap saturation layer $L^\star$ of Section 5.1, which we computed only for Qwen3; on Qwen3 the two differ by one to two layers. All four transfer exits of Table 4 lie above $L^\star_{1\%}$, which is why both the 0.60 and the 0.70 ratio land on the plateau for these two models.

Readout-swap control. Higher probe decodability under CLS could in principle reflect only that CLS trains a linear head on the hidden state directly. S2S, however, is also a linear objective on the same final state: its verdict is the sign of $(e_{\texttt{Yes}} - e_{\texttt{No}})^\top h$, a boundary whose direction is fixed by the unembedding. To separate the representation from the trained head, we discard both heads, freeze each backbone, and fit the same fresh linear probe, layer by layer, on the features of the S2S and CLS models. The last column of Table 12 reports the difference at the best layer: the S2S representation trails by 0.153, 0.134, 0.128, and 0.138 F_1 on Qwen3-4B, Qwen3-8B, Llama-3.2-3B, and Mistral-7B-v0.3. A fresh head does not recover the gap, so the gap sits in the representation.

F.5 Replication of the layer structure on QNLI and SST-2

To test whether the structure is specific to hallucination detection, we fine-tune Qwen3-4B with the unchanged CLS and S2S recipes on two further binary tasks. QNLI uses its sentence as context and its question as claim; SST-2 uses the review as context and the fixed claim "The review expresses positive sentiment." Each task uses 22,827 balanced training examples to match the hallucination training set, and we hold out 5,463 QNLI and 872 SST-2 test examples. We then fit a logistic-regression probe

Table 12: **Linear-probe accuracy and saturation depth across backbones.** Scores are the maximum over layers of mean probe macro-F_1 across three probe seeds, using the original fine-tuning runs. $L^\star_{1\%}$ is the earliest layer within 0.01 of that maximum. The final column subtracts the best-layer scores of separately fitted probes with the same architecture and training protocol. These are descriptive probe comparisons, not early-exit selection results.

Backbone	L	BASE	S2S	CLS	$L^\star_{1\%}$	$L^\star_{1\%}/L$	CLS−S2S (same probe family)
Qwen3-4B	36	0.798	0.835	**0.988**	L23	0.64	+0.153
Qwen3-8B	36	0.809	0.850	**0.984**	L21	0.58	+0.134
Llama-3.2-3B	28	0.742	0.807	**0.935**	L12	0.43	+0.128
Mistral-7B-v0.3	32	0.756	0.848	**0.986**	L18	0.56	+0.138

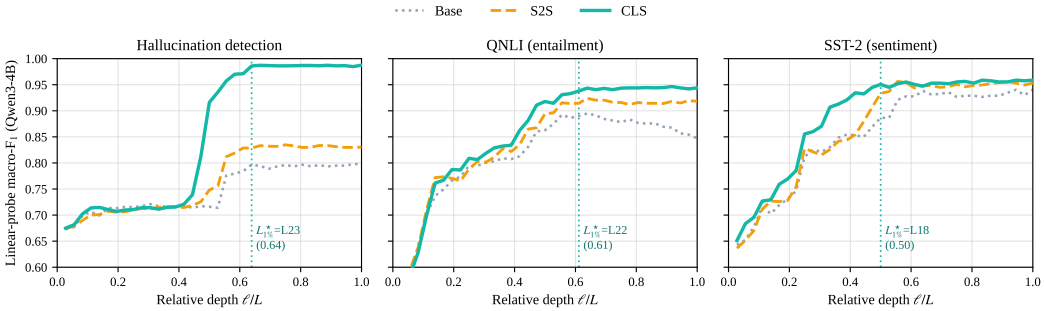

Figure 6: **The layer structure replicates on QNLI and SST-2.** Per-layer probe F_1 for Qwen3-4B fine-tuned on hallucination detection, QNLI, and SST-2 with the same CLS/S2S recipe. The shape and the location of $L^\star_{1\%}$ are shared; the CLS−S2S gap shrinks on the two easier tasks because the pretrained backbone already decodes them well.

per layer on 3,000 training examples and score macro-F_1 on up to 2,000 held-out examples for the BASE, S2S, and CLS models (one fine-tuning seed, one probe seed; Table 13, Figure 6).

Both tasks reproduce the substrate, the sharp rise, and the plateau, and $L^\star_{1\%}$ falls at 0.61 (QNLI) and 0.50 (SST-2) of depth, inside the 0.43 to 0.64 band of the four hallucination detectors. The CLS−S2S gap at the best layer, however, is 0.023 on QNLI and 0.002 on SST-2, against 0.13 to 0.16 on hallucination detection. The pretrained backbone already reaches 0.895 and 0.940 probe F_1 on the two tasks, so fine-tuning has little to reorganize; on hallucination detection it starts from 0.74 to 0.81. The shape of the curve is task-general, and the size of the CLS advantage tracks how much the task requires the representation to change.

Table 13: **Best-layer probe scores and saturation depths on three tasks.** QNLI and SST-2 use a single fine-tuning seed and the logistic-regression protocol described in Appendix F.5; hallucination detection uses the three-probe-seed protocol in Appendix F.4. Differences are computed before rounding. $L^\star_{1\%}$ summarizes early saturation and does not test statistical equivalence.

Task (Qwen3-4B)	BASE	S2S	CLS	CLS−S2S	$L^\star_{1\%}$(CLS)	$L^\star_{1\%}/L$
Hallucination detection	0.798	0.835	**0.988**	+0.153	L23	0.64
QNLI (entailment)	0.895	0.924	**0.947**	+0.023	L22	0.61
SST-2 (sentiment)	0.940	0.956	**0.959**	+0.002	L18	0.50

F.6 Measured latency of truncated backbones

Table 14 reports per-example forward latency of the CLS models with the transformer blocks above the exit physically removed before the adapter is loaded, at batch size 1 and sequence length 2,048 in bfloat16, averaged after five warm-up iterations, on an NVIDIA H200 and an RTX A6000. Removing 31% of the blocks (L25) cuts latency by 1.42 to 1.44× and removing 44% (L20) by 1.76 to 1.79×.

Batched throughput at batch size 16 rises accordingly, from 4.8 to 6.9 to 8.6 examples per second for Qwen3-4B on the A6000 and from 14.6 to 20.9 to 26.1 for Qwen3-8B on the H200. The speedups fall a few percent short of the layer fraction because the cost of the embedding and the readout does not shrink with depth.

Table 14: **Forward-pass latency of physically truncated backbones.** Mean time per forward pass at batch size 1 and sequence length 2,048, using bf16 and SDPA. Models use pretrained backbone weights and a two-class head, without task adapters. Parentheses give speedup over full depth. These measurements isolate the effect of reducing depth; they are not end-to-end detector timings.

GPU	**Model**	**Full depth**	**L25 (-31% blocks)**	**L20 (-44% blocks)**
H200 (datacenter)	Qwen3-4B	47.4 ms	33.0 ms (**1.44**×)	26.5 ms (**1.79**×)
H200 (datacenter)	Qwen3-8B	71.8 ms	50.1 ms (**1.43**×)	40.1 ms (**1.79**×)
RTX A6000 (constrained)	Qwen3-4B	220.6 ms	155.8 ms (**1.42**×)	125.0 ms (**1.76**×)

NeurIPS Paper Checklist

1. **Claims**

   Answer: [Yes]

   Justification: The abstract and introduction clearly state the paper's main empirical and analytical claims, including the performance gap between CLS and S2S, the objective-dependent changes in representation geometry, the emergence of an InfoWindow during training, and the resulting early-exit rule. Each claim is evaluated through controlled experiments and layer-wise analyses in Sections 3–6.

2. **Limitations**

   Answer: [Yes]

   Justification: The paper discusses limitations and future directions in Section 8 and Appendix A. These discussions clarify the scope of the present study and suggest that the observed representation-level patterns may offer useful insights for future research on hallucination detection and efficient model verification.

3. **Theory assumptions and proofs**

   Answer: [N/A]

   Justification: The paper does not present formal theoretical results or theorems. The mathematical expressions are used to define training objectives, diagnostic quantities, and empirical estimators for the representation and early-exit analyses.

4. **Open access to data and code**

   Answer: [No]

   Justification: The study builds on publicly available datasets and open-weight model backbones. The appendices provide detailed documentation of the experimental pipeline, including data processing, model configurations, training hyperparameters, probing protocols, bootstrap procedures, and evaluation settings, to support reproducibility and future research.

5. **Experimental setting/details**

   Answer: [Yes]

   Justification: The paper provides the main details needed to reproduce the experimental results, including datasets, splits, prompts, model backbones, LoRA configurations, optimization hyperparameters, probing protocols, bootstrap procedures, and early-exit settings. These details are reported in Appendices C–E.

6. **Experiment statistical significance**

   Answer: [Yes]

   Justification: The paper reports uncertainty estimates for the key layer-wise conclusions. In particular, paired bootstrap confidence intervals are used for plateau interchangeability and fusion analyses in Sections 5.1–5.2 and Appendices D.4–D.5.

7. **Experiments compute resources**

   Answer: [Yes]

   Justification: The paper reports the main compute resources used for training and evaluation, including single NVIDIA H200 GPUs, bfloat16 execution, training steps, batch sizes, and approximate wall-clock time for early-exit runs in Appendix E. Additional training, probing, and representation-analysis details are provided in Appendices C–D.6.

8. **Code of ethics**

   Answer: [Yes]

   Justification: The research uses public datasets and open-weight model backbones to study hallucination detection and efficient fine-tuning. The work is intended to improve understanding of model verification behavior and does not introduce new data collection, human-subject experiments, or high-risk deployment artifacts.

9. **Broader impacts**

   Answer: [Yes]

Justification: Appendix A discusses the broader implications of efficient hallucination detection for more trustworthy RAG and agentic systems. The paper also emphasizes that automated verifiers should complement, rather than replace, task-specific evaluation, careful deployment testing, and human oversight where appropriate.

10. **Safeguards**

    Answer: [N/A]

    Justification: The paper does not introduce or release a new high-risk generative model, scraped dataset, or dual-use foundation model. It analyzes hallucination-detection fine-tuning using existing public datasets and open-weight model backbones.

11. **Licenses for existing assets**

    Answer: [Yes]

    Justification: The paper cites the original sources for all datasets, model backbones, and software packages used in the study, and describes their roles in the experimental setup. Appendix B further states that these existing assets are publicly available research resources and that their use follows the access conditions and usage terms specified by the original providers or hosting platforms.

12. **New assets**

    Answer: [N/A]

    Justification: The study does not introduce a new dataset, benchmark, model checkpoint, or software package requiring separate asset documentation. All experiments are conducted with existing research datasets and model backbones.

13. **Crowdsourcing and research with human subjects**

    Answer: [N/A]

    Justification: The paper does not involve new crowdsourcing experiments or new research with human subjects. All experiments use existing datasets.

14. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Answer: [N/A]

    Justification: The paper does not involve new human-subject experiments, crowdsourcing, participant recruitment, or participant data collection.

15. **Declaration of LLM usage**

    Answer: [Yes]

    Justification: LLMs are central to the research methodology. The paper fine-tunes decoder-only LLM backbones under S2S and CLS objectives, analyzes their layer-wise representations, and evaluates early-exit variants. The models, training setup, and evaluation protocol are described in Sections 2–6 and Appendices C–E.