WHEN LESS COMPUTE IS MORE: ADAPTIVE EARLY EXIT IMPROVES PRETRAINED OUTLIER DETECTION

Tianyang Zhou
Carnegie Mellon University
tzhou3@andrew.cmu.edu

Leman Akoglu
Carnegie Mellon University
lakoglu@andrew.cmu.edu

ABSTRACT

Pretrained tabular foundation models process every dataset at a fixed depth, with inference costs growing with dataset size. To address this, we present the first study of depth-adaptive early-exit for pretrained outlier detection models. While early-exit is typically motivated by efficiency, we uncover a surprising benefit: exiting at the optimal intermediate layer *can also improve detection performance* on diverse real-world benchmarks by 4.7–7.3% on average, consistent across three distinct foundation models. First, we investigate the factors driving these gains, and identify a key mechanism: *context pollution*, i.e., the presence of outliers among in-context samples. Our analysis reveals that nearby in-context samples exert increasing influence on query predictions at greater depths, consistent with a retrieval-based view of these models. In effect, early-exit alleviates the adverse effects of retrieving accurate-yet-polluted neighbors, with gains of 13–21% when context pollution matches the natural outlier rate. Motivated by these findings, we pretrain a plug-in router to select a dataset-specific exit layer, using query outlier labels as privileged information available only during router training. The router operates *post hoc*, leaving the base model parameters and prediction head unchanged. Experiments on three large real-world benchmarks show that, on clean context, the router recovers up to 45% of the oracle gain with up to 1.8× speedup across three pretrained backbones, with larger gains as context pollution increases.

1 INTRODUCTION

Empirical neural scaling laws (Kaplan et al., 2020) have motivated the rapid scaling of foundation models in both parameter capacity and depth, often reaching billions of parameters (Hoffmann et al., 2022). However, increasing model sizes also raise the computational cost of inference, even for inputs that may not require the model's full capacity. This motivates *compute-adaptive inference*: allocating computational effort according to the given input and task (Schwartz et al., 2020).

Adaptive computation can be introduced through architectural design, training for variable-depth execution, or post-hoc adaptation of pretrained backbones. Natively adaptive designs include Looped Transformers that vary effective depth through repeated block execution (Giannou et al., 2023; Geiping et al., 2025) and Mixture-of-Experts that route tokens to specialized sub-networks to decouple massive capacity from active per-token compute (Shazeer et al., 2017; Fedus et al., 2022; Du et al., 2022). Beyond architectural designs, other work trains auxiliary adapters alongside the backbone without altering the core architecture to enable dynamic early-termination, such as entropy-based exits (Xin et al., 2020; Liu et al., 2020) and layer-specific prediction heads (Zhou et al., 2020).

The cost of retraining large adaptive models motivates a second line of work that trains only auxiliary modules. These approaches retrofit existing pretrained models to reduce inference depth by introducing lightweight decision mechanisms such as confidence-based stopping criteria (Schuster et al., 2022; Küken et al., 2025), frozen-backbone intermediate decoders (Küken et al., 2025), and learned post-hoc routing policies (He et al., 2025; Luo et al., 2025). (See Appdx. A for details.)

In this work, we investigate *post hoc depth-adaptive early exit* for pretrained tabular outlier detection models. These models process every dataset via in-context learning at a fixed depth irrespective of the input. We seek to terminate inference dynamically at dataset-specific intermediate layers, reserving deeper processing for datasets that benefit from it.

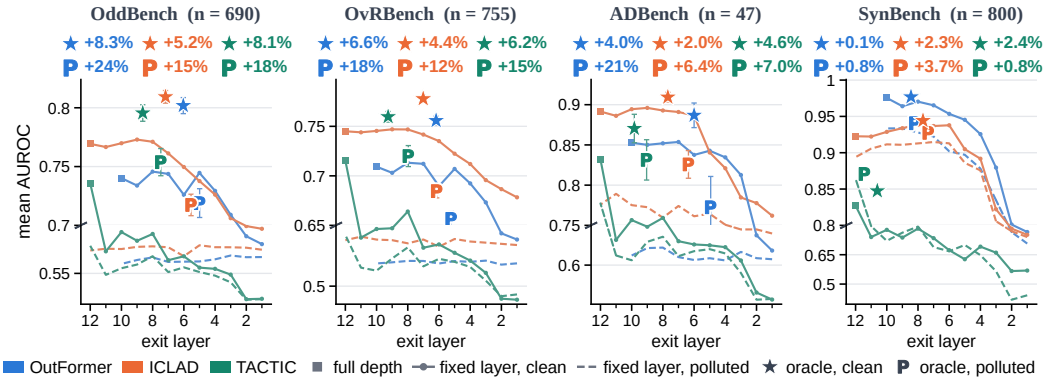

Figure 1: OD performance when exiting at a fixed layer under clean (–) and polluted (- -) context; context pollution hinders performance across different pretrained models and benchmarks. Per-dataset oracle early-exit boosts performance for both clean (★) and especially polluted (**P**) context.

While adaptive inference traditionally targets efficiency, our investigation uncovers a surprising opportunity: reducing depth can improve both latency *and* detection performance. We trace a key part of these gains to *context pollution*, which causes outlier masking: deeper processing strengthens the influence of nearby context samples, which can harm predictions when those neighbors are themselves outliers. Early exit helps mitigate masking by limiting the accumulation of context pollution in query representations, making early-exit gains under pollution more pronounced (see Figure 1).

Motivated by these dual benefits, we pretrain a plug-in router, called ROUT, to select a dataset-specific exit layer, leaving the base foundation model and its prediction head entirely unchanged. A core challenge is that while context outliers amplify early-exit benefits, inference-time datasets lack ground truth labels. To bridge this gap, we leverage query outlier labels as privileged information (Vapnik & Izmailov, 2015) exclusively during ROUT's training, requiring no labels at inference. Our main contributions are summarized as follows.

- **Post-hoc Early Exit for Pretrained Outlier Detection.** We present the first study of depth-adaptive early exit for pretrained outlier detection (OD) models, addressing dataset-specific exit selection without labeled outliers in target datasets. Our approach preserves the pretrained backbone and prediction head, requiring only a lightweight routing module.
- **"Pollution in the Deep": Why Early Exit Can Improve Detection.** We uncover a dual benefit of early exit: reducing computation can also improve detection performance. We identify context pollution as a key mechanism, showing that deeper layers increasingly rely on nearby in-context samples whose influence can become detrimental when those neighbors are outliers. Early exit alleviates this effect: across three distinct pretrained OD backbones, optimal intermediate layer predictions outperform their full-depth counterparts on diverse real-world benchmarks by 4.7–7.3% on average, while also reducing computation.
- **Learning to Route under Context Pollution without Inference-time Labels.** We introduce ROUT, a dataset-adaptive router that combines sequential stopping and retrospective exit selection with annealed query labels as privileged information during synthetic pretraining, preserving the backbone and prediction head and requiring no inference-time labels.
- **Evaluation across Backbones, Benchmarks, and Pollution Levels.** Experiments across three pretrained backbones, OUTFORMER (Ding et al., 2026b), TACTIC (Marszałek et al., 2026), and ICLAD (Wei & Armanfard, 2026), and three large real-world benchmarks and a synthetic benchmark (Ding et al., 2026a; Han et al., 2022) establish that higher accuracy and lower latency at inference are not mutually exclusive objectives for pretrained outlier detection models, where strategic early-exit by ROUT yields up to 45% of the oracle gain and up to 1.8× inference speedups on clean context, with larger gains as context pollution increases.

2 Background and Findings

Preliminaries. Pretrained tabular foundation models (TFMs) address the learning problem via *in-context* learning (Xie et al., 2022). A given dataset $\mathcal{D}$, comprising samples in $\mathbb{R}^d$, is partitioned into two parts: a set of context points $\mathcal{D}_{\text{ctx}} \subset \mathcal{D}$ representing the training data, and a query set $\mathcal{D}_{\text{qry}} = \mathcal{D} \setminus \mathcal{D}_{\text{ctx}}$ serving as the unlabeled test set. In supervised TFMs for standard classification

or regression, $\mathcal{D}_{\text{ctx}}$ is labeled, with feature vectors concatenated with their respective target labels, whereas outlier detection TFMs are pretrained using both unlabeled context and query sets.

TFMs directly model the posterior predictive distribution $p(y \mid \mathbf{x}, \mathcal{D}_{\text{ctx}})$ over the unlabeled query set $\mathcal{D}_{\text{qry}}$ given context set $\mathcal{D}_{\text{ctx}}$. They are typically optimized on synthetic datasets $\mathcal{D} \sim \pi(\mathcal{D})$ drawn from a data-generating distribution $\pi(\cdot)$, specifying prior probabilities over datasets. Training is supervised using cross-entropy loss to predict the labels of query points given the context. Pretrained outlier detection models specifically leverage ground-truth outlier/inlier labels from the synthetic priors for supervision, while context points are provided without labels. The training objective is

$$\theta^\star = \arg\min_\theta \mathbb{E}_{\mathcal{D}\sim\pi(\mathcal{D})}\Big[\sum_{\mathbf{x}\in\mathcal{D}_{\text{qry}}} \ell\left(y(\mathbf{x}), \hat{p}_\theta(y = \text{outlier} \mid \mathbf{x}, \mathcal{D}_{\text{ctx}})\right) \Big] . \tag{1}$$

During zero-shot inference, a pretrained TFM labels outliers in input datasets in a single forward pass by conditioning directly on an unlabeled context set without any updates to the model weights.

Outlier Detection TFMs. Pretrained tabular outlier detection (OD) models differ along two primary axes: (1) their synthetic data priors over inlier distributions and outlier-generating mechanisms, and (2) their training supervision regimes. FOMO-0D (Shen et al., 2025), the seminal TFM for zero-shot OD, is pretrained on Gaussian mixture models (GMMs) featuring subspace outliers with inflated variance across random subsets of features, operating exclusively under the **one-class, clean context** setting with inlier-only context points. Its successor, OUTFORMER (Ding et al., 2026b), broadens this paradigm by training on a mixture of heterogeneous priors, including GMMs, structural causal models (SCMs), and copulas, while maintaining an inlier-only clean context.

Similarly, TACTIC (Marszałek et al., 2026) leverages a hybrid prior of SCMs and GMMs, modeling rare-class outliers from SCMs alongside local, global, and clustered outliers derived from variance-inflated or mean-shifted GMMs. While TACTIC-C adopts the standard one-class setting, TACTIC-N addresses the **unsupervised, polluted context** scenario by allowing unlabeled outliers within the context to enhance real-world robustness. Finally, ICLAD (Wei & Armanfard, 2026) employs SCM-based datasets incorporating rare-class outliers and additive-noise subspace outliers, while expanding supervision to encompass one-class, unsupervised, as well as semi-supervised context where a small subset of context points receive explicit outlier labels while the rest stay unlabeled.

Early Exiting OD-TFMs – Setup. *Backbones.* We study three OD-TFMs with distinct training regimes (OUTFORMER, TACTIC, and ICLAD) to analyze their early-exit behavior across two context settings: *inlier-only* and *polluted*. Context points are provided without labels, i.e., the semi-supervised regime is excluded. Each backbone is kept entirely frozen, including the parameters of its final classification head. For prediction, representations of query points from intermediate layers are routed to the fixed head to quantify detection performance. The dataset-specific optimal layer yielding peak performance is designated as the *oracle layer*.

Datasets. Our study spans (1) SynBench with 800 synthetic datasets generated from OUTFORMER's priors (GMMs, SCMs, copulas); (2) ADBench (Han et al., 2022), comprising its 47 classical tabular OD datasets; and (3) OddBench and (4) OvRBench, both sourced from MacrOData (Ding et al., 2026a), OddBench featuring 690 datasets with real-world anomalies (system faults, errors, malware, etc.) and OvRBench consisting of 755 repurposed classification datasets, one class designated as inliers and the rest subsampled to form outliers.

Context Pollution. While SynBench supplies perfectly clean, simulated inlier-only context data via its standard splits, real-world datasets may provide unsupervised random splits, or may harbor inherent label noise, which we refer to as "hidden pollution", reflecting the difficulty of acquiring purely inlier data in practice.

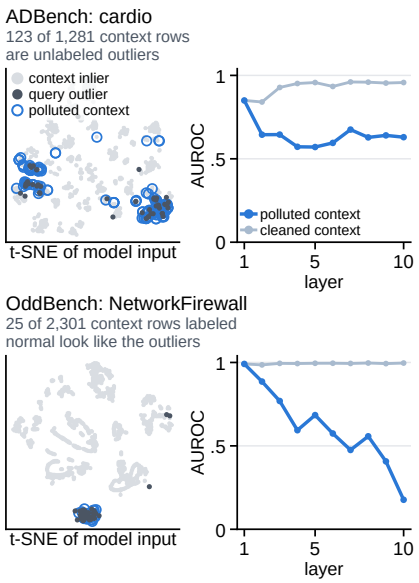

Figure 2: Hidden pollution in two official real-world splits: (left) t-SNE of the model input, polluting context rows circled; (right) layer-wise AUROC (OUTFORMER) with and without them.

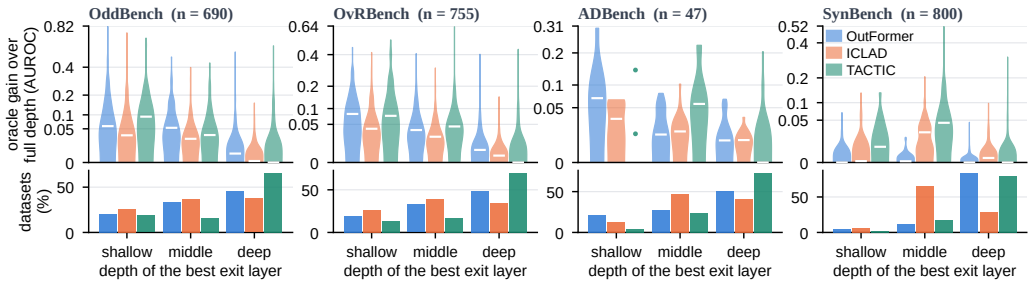

Figure 3: Oracle gain over full depth, grouped by which third of the depth the best exit layer falls in (square-root scaled y-axis); bars give the share of datasets per group.

Figure 2 illustrates these issues on two real-world datasets. We study context pollution by matching the proportion of outliers in context and query sets to the natural rate in the underlying dataset.

Early-Exit Results & Observations. Figure 1 illustrates detection performance across layers when each model exits at a *fixed layer* $l \in \{\texttt{max\_depth}, \ldots, 1\}$ under both clean and polluted (at original rate) context, where `max_depth` is 10 for OUTFORMER and 12 for both TACTIC and ICLAD. Context pollution consistently degrades detection performance across all models. Average performance across datasets remains fairly stable and comparable to the final-layer performance before it starts a considerable decline. Surprisingly, when models exit at the dataset-specific *oracle layer*, average performance actually *improves* relative to the final layer for all models, as depicted with ★ (7.3% for OUTFORMER, 7.0% for TACTIC, and 4.7% for ICLAD across real-world benchmarks). Performance gains are even *more pronounced under polluted context*, as depicted with **P** (20.9% for OUTFORMER, 16.3% for TACTIC, and 13.2% for ICLAD across real-world benchmarks). This lift in gains from clean to polluted is significant across models and benchmarks, as Table 1 shows.

Table 1: Oracle performance gains (%): Clean → Polluted; p-values in parentheses.

Backbone	OddBench (690)	OvRBench (755)	ADBench (47)
OutFormer	$+8.3 \rightarrow +24.0$ (3×10^{-48})	$+6.6 \rightarrow +18.1$ (2×10^{-56})	$+4.0 \rightarrow +21.4$ (2×10^{-8})
TACTIC	$+8.1 \rightarrow +18.5$ (3×10^{-48})	$+6.2 \rightarrow +15.0$ (2×10^{-44})	$+4.6 \rightarrow +7.0$ (0.086)
ICLAD	$+5.2 \rightarrow +15.4$ (4×10^{-60})	$+4.4 \rightarrow +11.6$ (2×10^{-48})	$+2.0 \rightarrow +6.4$ (10^{-5})

Figure 3 (top) shows the performance gain distribution under clean context by oracle depth, grouped as shallow, middle and deep. While relative gains are marginal on the synthetic SynBench with perfectly clean context, they become *more pronounced on real-world* datasets (0.1% vs 7.3% for OUTFORMER, 2.4% vs 7.0% for TACTIC, and 2.3% vs 4.7% for ICLAD).

Figure 3 (bottom) displays the share of datasets whose oracle layer falls in each depth group (per-layer histograms in Figure 8). Oracle layers vary notably for all models across all benchmarks, underscoring the potential gains that early exit offers for OD-TFMs. A notable observation for OUTFORMER is that the majority of SynBench datasets accumulate at the final layer as their oracle layer, reflecting the model's pretraining on matching underlying priors for final-layer optimization. In contrast, the distribution disperses considerably across real-world benchmarks.

3 THE PARADOX OF EARLY EXIT: UNDERSTANDING PERFORMANCE GAINS

Before translating these findings into a practical early-exit strategy for OD-TFMs, we seek to understand the apparent paradox: intermediate representations can yield substantially better predictions through a head trained exclusively on final-layer outputs, without updating any parameters.

The decline in detection performance and increase in oracle early-exit gains as we move from purely clean synthetic data to real-world benchmarks, and further to polluted contexts, suggest that in-context outliers play a crucial role. Real-world gains under nominally clean contexts may likewise stem from "hidden pollution", i.e., outlier-like samples within designated inlier sets.

In-context learning has been interpreted through several perspectives, including implicit Bayesian inference (Xie et al., 2022), implicit gradient descent (Dai et al., 2023), and specialized circuits such as induction heads (Olsson et al., 2022). Particularly relevant to our observations is the retrieval-based perspective proposed for tabular generalization, whereby models identify and aggregate relevant context examples (Shaheen et al., 2026). This view is consistent with connections between attention, kernel smoothing, and non-parametric regression (Ilin et al., 2026; Santos et al., 2026).

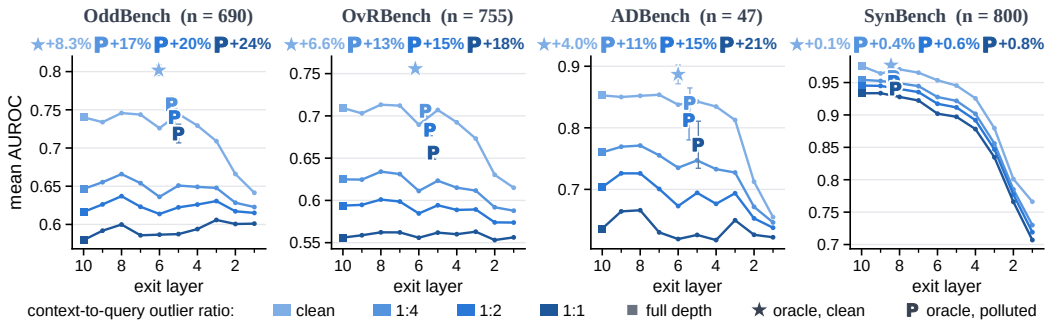

Figure 4: Fixed-layer and oracle performance of OUTFORMER as held-out context pollution increases; ★ depicts the oracle on clean context and **P** the oracle under pollution.

Under this interpretation, nearby context outliers may provide misleading evidence for a query outlier, inducing "**outlier masking**": outliers in training data make similar query outliers appear less anomalous. We hypothesize that repeated context aggregation across layers amplifies this masking effect, progressively suppressing the anomaly scores of affected queries. Early exit may thus mitigate outlier masking by limiting the accumulation of context pollution in query representations.

We probe this hypothesis in two pollution setups. In **held-out pollution**, we start from clean context and progressively inject known held-out outliers, adjusting the context-to-query outlier ratio across 1:4, 1:2, and 1:1 (natural rate). Figure 4 shows for OUTFORMER that increasing pollution degrades detection performance while amplifying the gains from oracle early-exit, consistent with earlier observations (similar results for ICLAD and TACTIC are in Figs. 9 and 10). We also inject **near-duplicate pollution**, where we plant "sibling" outliers into the context by creating perturbed copies of the query outliers. The results strongly align with prior observations. (See Figs. 11, 12, and 13.)

Next, we quantify **sibling influence** at each exit layer using gradient-based saliency (Kokhlikyan et al., 2020): the gradient norm of a query's outlier score with respect to its sibling's input features. We normalize this norm for scale differences across datasets (details in Appdx. B.1). Figure 5 presents results for OUTFORMER (see also Figs. 14 and 15). Notably, targeted pollution increasingly impairs detection (left). Furthermore, sibling influence, and thus the impact of pollution, increases with depth (middle); in fact, the performance loss closely tracks this influence across layers (right).

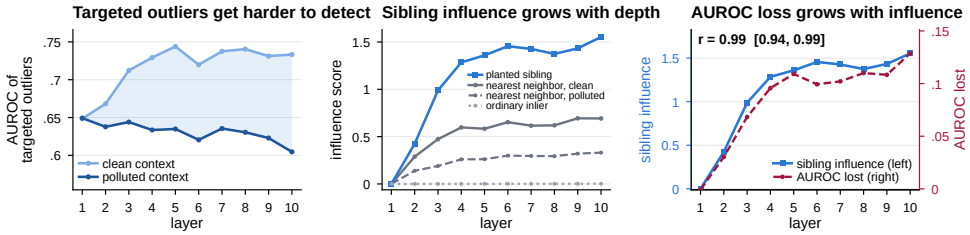

Figure 5: Effect of one planted sibling per targeted outlier (OUTFORMER): **(left)** Detection of targeted outliers declines with depth; **(middle)** Outlier scores increasingly rely on the sibling with depth; **(right)** Sibling influence and AUROC loss rise together, showing a strong correlation of 0.99.

4 LAYER-ADAPTIVE ROUTING: LEARNING WHEN TO EXIT

Many depth-adaptive methods use backbone activations to route computation layer by layer (Elbayad et al., 2020; Heakl et al., 2026; Küken et al., 2025). They do not separately choose among predictions from layers already computed. For OD-TFMs, this matters: deeper layers can *impair* detection; thus, getting stuck in a later layer without **retrospective recovery** can be detrimental. To enable this, ROUT introduces **layer-wise attention** to separate computation depth from output selection, comparing all computed layers to decide when to stop and which prediction to return.

Architecture. At its core, ROUT uses layer-wise context and query representations to select an exit layer. As Figure 6 shows, it comprises two attention modules: Sample Attention attends across samples at layer k, while Layer Attention attends across each sample's representations from layers 1 through k. Following learned attention pooling (Lee et al., 2019), an MLP predicts a distribution $\mathbf{p}_k$ over all L candidate exit layers. We stop at the first layer where the probability mass on the layers already computed reaches τ, i.e., $K = \min\{k : \sum_{j \le k} p_{k,j} \ge \tau\}$. At that point, ROUT

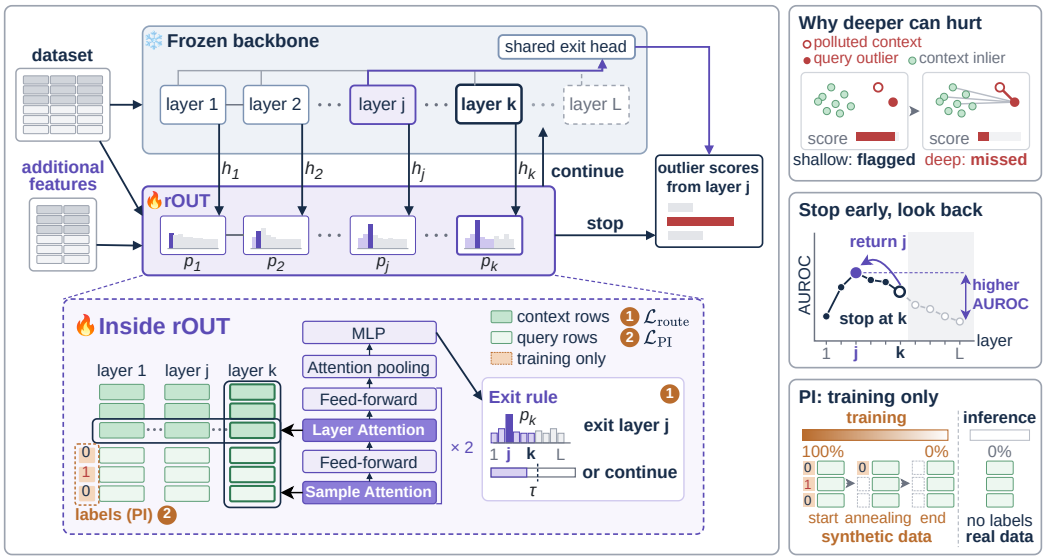

Figure 6: **Left:** ROUT reads the frozen backbone layer by layer, stops at layer k, and returns the best computed layer $j \leq k$. **Right:** deeper layers can lean on polluted context that masks query outliers; thus, ROUT stops early and looks back; query labels are training-only privileged information.

returns the backbone's predictions from the most probable layer among those already computed, $\hat{j} = \arg\max_{j \leq K} p_{K,j}$. Following the validation-based threshold selection of Elbayad et al. (2020), we select a single τ per backbone on synthetic validation datasets to maximize $\mathbb{E}_{\text{val}}[\text{AUROC}_{\hat{j}} - \lambda K]$, where λ controls the depth penalty. We apply this threshold unchanged to every test dataset.

Training Objective. We train ROUT to balance the quality of the returned predictions against the depth computed. For each training dataset, let A_j be the backbone's AUROC at exit layer j, computed with the query outlier labels, and $R_j = \max_i A_i - A_j$ the regret of selecting layer j. The depth penalty λ controls the trade-off between AUROC and computation: without it, the router tends to defer its decision until the final few layers, resulting in little computational saving (see Table 15).

Sequential stopping loss. We formulate sequential stopping probabilistically and optimize the expected loss over stopping depths (Banino et al., 2021). Conditioned on reaching layer k, the stopping probability (as a function of trainable router parameters θ) is $\pi_k(\theta) = \sum_{j \leq k} p_{k,j}(\theta)$, giving $w_k(\theta) = \pi_k(\theta) \prod_{i<k}(1-\pi_i(\theta))$ as the probability of stopping at k, with $\sum_{k=1}^{L} w_k(\theta) = 1$. Then,

$$\mathcal{L}_{\text{seq}}(\theta) = \sum_{k=1}^{L} w_k(\theta) \Big(\sum_{j \leq k} \frac{p_{k,j}(\theta)}{\pi_k(\theta)} R_j + \lambda k \Big), \tag{2}$$

where the first term measures the expected regret of the layer returned upon stopping at k, which may be an earlier layer $j < k$, while the second penalizes the computation required to reach k. Training with a cross-entropy loss that targets the best layer would penalize all other exits equally, whereas the regret penalizes each exit in proportion to its AUROC gap from the best layer.

Layer-wise selection loss. To provide a direct exit-selection signal at every depth, we additionally supervise $\mathbf{p}_k(\theta)$ independently of the stopping probabilities:

$$\mathcal{L}_{\text{layer}}(\theta) = \frac{1}{L} \sum_{k=1}^{L} \sum_{j=1}^{L} p_{k,j}(\theta) \big(R_j + \lambda j \big). \tag{3}$$

$\mathcal{L}_{\text{layer}}$ incentivizes $\mathbf{p}_k$ to assign high probability to layers with a favorable accuracy–computation trade-off. Here each candidate is penalized for its own depth j; in contrast, $\mathcal{L}_{\text{seq}}$ penalizes the depth k actually computed, which exceeds j when the router returns an earlier layer.

The two terms form the routing objective, $\mathcal{L}_{\text{route}} = \mathcal{L}_{\text{seq}} + \alpha \mathcal{L}_{\text{layer}}$, with weight α on layer-wise selection. We additionally use entropy regularization to prevent the selection distributions from collapsing onto a single layer (Appdx. B.3).

Outlier Labels as Privileged Information (PI). We pretrain ROUT on synthetic datasets drawn from diverse data priors, with ground-truth outlier labels and varying levels of context pollution.

Query outlier labels are available during pretraining, whereas inference-time datasets are fully unlabeled. Following the learning using privileged information (LUPI) framework (Vapnik & Vashist, 2009; Vapnik & Izmailov, 2015), we use these labels both as supervision and as privileged inputs.
PI as supervision. $\mathcal{L}_{\text{route}}$ uses query labels through *overall* layer-wise AUROC. We also use them to directly supervise the router's representations through an auxiliary objective $\mathcal{L}_{\text{PI}}$, including binary cross-entropy for predicting *individual* query rows' outlier labels. This supervision helps the router distinguish inliers from outliers when selecting an exit. We minimize the combined loss

$$\min_{\theta} \quad \mathcal{L}_{\text{route}}(\theta) + \mathcal{L}_{\text{PI}}(\theta) \; . \tag{4}$$

The auxiliary terms and their weights are detailed in Appdx. B.3.

PI as annealed input. Besides outlier label supervision, we expose outlier structure through privileged inputs: learned embeddings of the revealed query labels and label-derived summaries of the query outliers. Training begins with all query labels and their summaries available, then progressively withdraws both, using cosine schedules (Appdx. B.3) so that ROUT learns to route without them, as PI input is only available during training. When a query label is withheld, its embedding is replaced by a shared embedding indicating that the label is unavailable. The final training phase uses no privileged inputs, matching the fully unlabeled inference setting.

Additional Features. Alongside the backbone representations, the router receives, for each row of the dataset, its raw input features, the backbone's layer-wise outlier scores, and 29 geometric features (Appdx. B.2). The geometric features describe neighborhood distances, local density, and each row's position relative to the context distribution. Raw inputs and scores from intermediate prediction heads add little computational overhead, and we restrict the types and number of geometric features to keep feature extraction overhead low.

Pretraining Priors. Finally, motivated by ICLAD's extensive priors, we extend OUTFORMER's data priors in two ways. First, we add unsupervised detection tasks with polluted contexts, including near-duplicate pollution. Second, we complement its mixed priors with class-structured outliers, by relabeling classes of TabPFN's classification datasets (Table 9).

5 EXPERIMENTS

Datasets and evaluation settings. We evaluate ROUT on ADBench (Han et al., 2022), OddBench, and OvRBench (Ding et al., 2026a), three real-world outlier detection benchmarks comprising 47, 690, and 755 datasets, respectively, and on SynBench (Ding et al., 2026a), a synthetic benchmark comprising 800 datasets (Table 8). We evaluate under clean context as well as the held-out and near-duplicate pollution settings described in Section 3.

Baselines. Besides **Full depth**, we consider **Half depth** as a simple heuristic baseline. We also compare to the tabular early-stopping approach of Küken et al. (2025), which we call **LA-Entropy**; it trains separate layer-specific prediction heads (decoders) per model post hoc and exits when the entropy of the predictive distribution falls below a threshold τ, returning the corresponding predictions. Following the original formulation, we use a single τ shared across all exit heads. We tune τ on synthetic validation datasets and keep it fixed across all test datasets. (See Figure 16 for the full threshold sweep.) We also report *Oracle*, depicting peak performance at the dataset-specific optimal exit layer, and *Best fixed layer*, selecting the same layer for all datasets within a benchmark to maximize average AUROC. Both use ground-truth test AUROCs and are reported as references.

Metrics. We report mean AUROC and the mean number of computed layers, counting all layers evaluated before the exit decision, even if ROUT selects an earlier layer for prediction. We report the relative AUROC gain of $m \in \{\text{ROUT}, \text{oracle}\}$ over full depth, $\frac{\bar{A}_m - \bar{A}_{\text{full}}}{\bar{A}_{\text{full}}} \times 100\%$, where $\bar{A}$ is mean AUROC across a benchmark's datasets. In Table 2, parentheses beside ROUT's AUROC give the share of the full-depth-to-oracle gap it closes, $\frac{\bar{A}_{\text{ROUT}} - \bar{A}_{\text{full}}}{\bar{A}_{\text{oracle}} - \bar{A}_{\text{full}}} \times 100\%$. For each benchmark, we report total computation in PFLOPs and total wall-clock inference time in seconds across all datasets on one H100 GPU, including backbone computation, router calls, and additional-feature computation.

Implementation details. We use the same router architecture for all three backbones: two blocks of Sample and Layer Attention with hidden size 256 and four attention heads, followed by learned attention pooling, totaling 4.6M parameters per router. The router processes up to 256 context and 1,024 query rows per dataset, while the backbone scores all query rows. We train on over 200,000 synthetic datasets and use around 6,000 held-out datasets for validation. Training uses AdamW and

Table 2: rOUT against fixed-depth baselines and per-layer optimized LA-Entropy$^{\triangle}$ (Küken et al., 2025) across benchmarks and backbones. **Bold** is the best deployable method (*Oracle* and *Best fixed layer* are for reference). In parentheses is the share of Full-depth-to-*Oracle* gap that rOUT closes.

		OddBench		OvRBench		ADBench		SynBench	
Backbone	Method	AUROC	Layers	AUROC	Layers	AUROC	Layers	AUROC	Layers
OutFormer (L=10)	*Oracle*	0.802	6.0	0.756	6.2	0.887	6.0	0.977	8.4
	Best fixed layer	0.746	8.0	0.713	8.0	0.854	7.0	0.975	10.0
	Full depth	0.740	10.0	0.709	10.0	0.853	10.0	**0.975**	10.0
	Half depth	0.745	5.0	0.707	5.0	0.843	5.0	0.945	5.0
	LA-Entropy	0.709	2.5	0.680	2.9	0.770	2.5	0.915	2.6
	rOUT (ours)	**0.761** (34%)	5.5	**0.730** (45%)	5.4	**0.860** (21%)	5.5	0.967	5.7
ICLAD (L=12)	*Oracle*	0.809	7.2	0.778	7.0	0.909	7.7	0.944	7.7
	Best fixed layer	0.773	9.0	0.747	9.0	0.896	9.0	0.939	8.0
	Full depth	0.769	12.0	0.745	12.0	0.891	12.0	0.923	12.0
	Half depth	0.750	6.0	0.735	6.0	0.887	6.0	0.938	6.0
	LA-Entropy	0.704	1.6	0.696	2.0	0.786	2.1	0.856	1.8
	rOUT (ours)	**0.776** (18%)	6.8	**0.755** (30%)	6.0	**0.891** (0%)	6.0	**0.938** (71%)	4.4
TACTIC (L=12)	*Oracle*	0.796	8.6	0.760	9.3	0.870	9.8	0.847	10.6
	Best fixed layer	0.736	12.0	0.715	12.0	0.832	12.0	0.827	12.0
	Full depth	0.736	12.0	0.715	12.0	0.832	12.0	0.827	12.0
	Half depth	0.603	6.0	0.603	6.0	0.678	6.0	0.673	6.0
	LA-Entropy	0.693	2.1	0.682	2.4	0.780	2.2	**0.876***	1.9
	rOUT (ours)	**0.747** (18%)	10.2	**0.729** (31%)	10.0	**0.849** (45%)	10.1	0.838 (55%)	9.0

$^{\triangle}$ Even a full sweep of τ for LA-Entropy does not improve over rOUT on real-world benchmarks (Figure 16).
* LA-Entropy can exceed *Oracle* as it retrains a separate head per layer, while *Oracle* uses the final-layer head like rOUT.

the privileged-input schedule described in Section 4. At inference, we average the routing logits of three independently trained seeds per backbone. The stopping threshold is selected on synthetic validation data. Full training and hyperparameter settings are provided in Appdx. B.3.

5.1 Results

Adaptive exits deliver better detection with less computation. Table 2 shows that rOUT matches or improves Full-depth AUROC in 11 of 12 settings while computing 15–63% fewer layers, and recovers up to 45% of the *Oracle* improvement on real-world benchmarks (also see Figure 17).

LA-Entropy exits much earlier, at 1.6–2.9 layers, but sacrifices detection performance: despite training a separate prediction head per layer, it underperforms even Half-depth inference for OutFormer and ICLAD on every benchmark, and Full-depth TACTIC on all real-world benchmarks.

Beyond Best fixed layer. rOUT delivers higher average performance than the *Best fixed layer* (selected using test labels) in 8 of 9 real-world settings, while reducing its number of computed layers by approximately 15–33%, highlighting the value of dataset-adaptive exit selection.

Cross-backbone gains on SynBench. On SynBench, generated from OutFormer's priors, rOUT also captures early-exit gains for ICLAD and TACTIC, closing 71% and 55% of their Full-depth-to-*Oracle* gaps while computing on average just 4.4 and 9.0 of 12 layers, respectively.

Table 3: Total compute and inference time (s) for each benchmark on one H100 GPU, full depth vs. rOUT (router calls and extra features included).

		OddBench				OvRBench				ADBench			
Backbone	Method	PFLOPs	Time (s)	**Speedup** (↑)	AUROC **gain** (↑)	PFLOPs	Time (s)	**Speedup** (↑)	AUROC **gain** (↑)	PFLOPs	Time (s)	**Speedup** (↑)	AUROC **gain** (↑)
OutFormer (L=10)	Full depth	2.29	111	–	–	3.07	146	–	–	0.21	9.9	–	–
	rOUT	1.21	69.0	1.61×	+2.8%	1.48	82.4	1.77×	+3.0%	0.11	6.0	1.64×	+0.9%
ICLAD (L=12)	Full depth	2.04	70.6	–	–	2.64	89.9	–	–	0.18	6.4	–	–
	rOUT	1.25	53.1	1.33×	+0.8%	1.36	56.7	1.59×	+1.4%	0.087	3.8	1.67×	+0.0%
TACTIC (L=12)	Full depth	2.71	183	–	–	3.64	246	–	–	0.25	17.0	–	–
	rOUT	2.52	178	1.03×	+1.5%	3.19	226	1.09×	+1.9%	0.19	13.9	1.22×	+2.1%

Table 3 shows that rOUT accelerates inference across all three backbones and benchmarks, achieving 1.61–1.77× speedups for OutFormer and 1.33–1.67× for ICLAD, delivering up to 3.0% relative AUROC gain. For TACTIC, rOUT retains more layers when beneficial for detection, achieving up to 1.22× acceleration alongside a 2.1% relative AUROC gain. These savings account for backbone computation, router calls, and additional-feature computation (breakdown in Table 11).

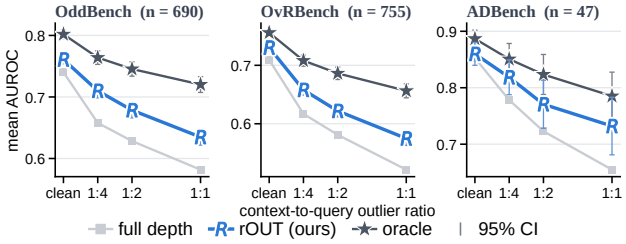

Figure 7: ROUT under **near-duplicate pollution** improves over Full depth while exiting early (OUTFORMER).

Context pollution amplifies the gains from adaptive exits. Table 4 shows that ROUT recovers the greater early-exit opportunities under pollution (Figure 18): on all three real-world benchmarks, its relative AUROC gains over Full-depth OUTFORMER are larger in every tested held-out pollution setting than with clean context (similar results in Table 12 for other backbones). On ADBench, the gain rises from 0.9% with clean context to 5.3%, 8.6%, and 10.1% at context-to-query outlier ratios of 1:4, 1:2, and 1:1, respectively. Gains also reach 8.0% on OddBench and 6.0% on OvRBench, compared with 2.8% and 3.0% with clean context. ROUT outperforms both Half depth and LA-Entropy across all tested pollution settings, demonstrating the value of adaptive exit selection when the context itself contains hidden outliers. Figure 7 mirrors these results under near-duplicate pollution, with even larger gains in this setting. (See Table 13 for full results.)

Table 4: ROUT under **held-out pollution** of the context (OUTFORMER). Columns give the context-to-query outlier ratio; at a 1:1 ratio, the context pollution rate is close to the dataset's natural outlier rate (median 0.8–1.2×). The row under ROUT gives its relative AUROC gain over full depth.

Held-out Pollution	OddBench				OvRBench				ADBench			
Method	Clean	1:4	1:2	1:1	Clean	1:4	1:2	1:1	Clean	1:4	1:2	1:1
Oracle	.802	.757	.741	.719	.756	.706	.684	.657	.887	.841	.813	.773
Best fixed layer	.746	.666	.637	.606	.713	.634	.601	.563	.854	.771	.726	.666
Full depth	.740	.646	.616	.580	.709	.625	.594	.556	.853	.760	.704	.636
Half depth	.745	.651	.622	.587	.707	.623	.594	.562	.843	.747	.694	.626
LA-Entropy	.709	.657	.639	.612	.680	.627	.606	.579	.770	.723	.706	.689
rOUT (ours)	**.761**	**.694**	**.666**	**.617**	**.730**	**.663**	**.629**	**.586**	**.860**	**.800**	**.764**	**.701**
— *AUROC gain*	+2.8%	+7.4%	+8.0%	+6.5%	+3.0%	+6.0%	+5.9%	+5.4%	+0.9%	+5.3%	+8.6%	+10.1%

Privileged information and richer representations strengthen routing. Table 5 presents component-wise ablations of ROUT on OUTFORMER. For a fair comparison, each variant's threshold τ is tuned to match the full router's mean layer budget. The full router outperforms all ablated variants significantly across all three real-world benchmarks. Removing PI causes the largest drop (up to 2.4% AUROC), followed by omitting either routing loss (up to 2.0%), additional features (up to 1.7%), and layer attention (up to 1.2%). These consistent performance drops confirm that each component contributes to exit selection. Together, they enable ROUT to significantly outperform Full depth across all benchmarks by up to 3.0%. We further ablate the training data recipes, where different sources benefit different evaluation settings (Table 14). Ablating the depth penalty ($\lambda = 0$) costs OUTFORMER 3.3–4.3 more computed layers at nearly unchanged AUROC (Table 15).

Table 5: Ablations of ROUT: relative AUROC change (%) from the full router at matched depth; Full depth is the backbone without routing; one-sided Wilcoxon $^{*}p<.05$, $^{**}p<.01$, $^{***}p<.001$.

OUTFORMER	Odd	OvR	AD
rOUT (full)	.761	.730	.860
w/o $\mathcal{L}_{\text{layer}}$	-1.5^{***}	-2.0^{***}	-0.5
w/o $\mathcal{L}_{\text{seq}}$	-0.6^{***}	-0.6^{**}	-0.4^{*}
w/o PI	-2.0^{***}	-2.4^{***}	-1.2^{*}
w/o layer attention	-0.8^{**}	-0.7^{***}	-1.2^{*}
w/o additional feat.	-1.7^{***}	-1.0^{***}	-1.5^{*}
Full depth	-2.7^{***}	-2.9^{***}	-0.9^{*}

6 CONCLUSION

We present the first study of early exit for pretrained tabular outlier detection, revealing that it improves not only efficiency but also performance. We identify context pollution as a key mechanism: deeper processing strengthens the influence of context outliers that mask query outliers, which early exit mitigates. Building on this, we introduce ROUT, which learns sequential stopping and retrospective layer selection using annealed access to privileged query labels during pretraining. Across three backbones and real-world benchmarks, ROUT achieves up to $1.8\times$ speedup while recovering up to 45% of Oracle gains on clean contexts, with gains growing to 59% under context pollution.

REFERENCES

Andrea Banino, Jan Balaguer, and Charles Blundell. PonderNet: Learning to ponder. In *8th ICML Workshop on Automated Machine Learning*, 2021. URL https://arxiv.org/abs/2107.05407.

Yilong Chen, Xueying Ding, and Leman Akoglu. VIP-COP: Context optimization for tabular foundation models. *arXiv preprint arXiv:2605.12904*, 2026.

Damai Dai, Yutao Sun, Li Dong, Yaru Hao, Shuming Ma, Zhifang Sui, and Furu Wei. Why can GPT learn in-context? language models secretly perform gradient descent as meta-optimizers. In *Findings of the Association for Computational Linguistics: ACL 2023*, pp. 4005–4019, 2023. doi: 10.18653/v1/2023.findings-acl.247.

Xueying Ding, Simon Klüttermann, Haomin Wen, Yilong Chen, and Leman Akoglu. MacrOData: New benchmarks of thousands of datasets for tabular outlier detection. In *Proceedings of the 32nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '26)*, pp. 8777–8788, 2026a. doi: 10.1145/3770855.3817520. URL https://doi.org/10.1145/3770855.3817520.

Xueying Ding, Haomin Wen, Simon Klüttermann, and Leman Akoglu. From zero to hero: Advancing zero-shot foundation models for tabular outlier detection. In *Proceedings of the 43rd International Conference on Machine Learning (ICML)*. PMLR, 2026b.

Nan Du, Yanping Huang, Andrew M Dai, Simon Tong, Dmitry Lepikhin, Yuanzhong Xu, Maxim Krikun, Yanqi Zhou, Adams Wei Yu, Orhan Firat, Barret Zoph, Liam Fedus, Maarten Bosma, Zongwei Zhou, Tao Wang, Yu Emma Wang, Kellie Webster, Marie Pellat, Kevin Robinson, Kathleen Meier-Hellstern, Toju Duke, Lucas Dixon, Kun Zhang, Quoc V Le, Yonghui Wu, Zhifeng Chen, and Claire Cui. GLaM: Efficient scaling of language models with mixture-of-experts. In Kamal Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato (eds.), *Proceedings of the Thirty-Ninth International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 5547–5569. PMLR, 17–23 Jul 2022.

Maha Elbayad, Jiatao Gu, Edouard Grave, and Michael Auli. Depth-adaptive transformer. In *International Conference on Learning Representations*, 2020.

Mostafa Elhoushi, Akshat Shrivastava, Diana Liskovich, Basil Hosmer, Bram Wasti, Liangzhen Lai, Anas Mahmoud, Bilge Acun, Saurabh Agarwal, Ahmed Roman, Ahmed A Aly, Beidi Chen, and Carole-Jean Wu. 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)*, pp. 12622–12642, 2024.

William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23(120):1–39, 2022.

Benjamin Feuer, Robin Tibor Schirrmeister, Valeriia Cherepanova, Chinmay Hegde, Frank Hutter, Micah Goldblum, Niv Cohen, and Colin White. TuneTables: Context optimization for scalable prior-data fitted networks. In *Advances in Neural Information Processing Systems*, volume 37, pp. 83430–83464, 2024.

Jonas Geiping, Sean McLeish, Neel Jain, John Kirchenbauer, Siddharth Singh, Brian R. Bartoldson, Bhavya Kailkhura, Abhinav Bhatele, and Tom Goldstein. Scaling up test-time compute with latent reasoning: A recurrent depth approach. In *Advances in Neural Information Processing Systems*, volume 38, 2025. URL https://arxiv.org/abs/2502.05171.

Angeliki Giannou, Shashank Rajput, Jy-Yong Sohn, Kangwook Lee, Jason D. Lee, and Dimitris Papailiopoulos. Looped transformers as programmable computers. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 11398–11442. PMLR, 2023.

Léo Grinsztajn, Klemens Flöge, Oscar Key, Felix Birkel, Philipp Jund, Brendan Roof, Mihir Manium, Shi Bin Hoo, Magnus Bühler, Anurag Garg, Dominik Safaric, Jake Robertson, Benjamin Jäger, Simone Alessi, Adrian Hayler, Vladyslav Moroshan, Lennart Purucker, Philipp Singer, Alan Arazi, Julien Siems, Jan Hendrik Metzen, Georg Grab, Nick Erickson, Siyuan Guo, Eliott Kalfon, Simon Bing, David Salinas, Clara Cornu, Lilly Charlotte Wehrhahn, Diana Kriuchkova, Kursat Kaya, Lydia Sidhoum, Marie Salmon, Jerry Chen, Madelon Hulsebos, Yann LeCun, Samuel Müller, Bernhard Schölkopf, Sauraj Gambhir, Noah Hollmann, and Frank Hutter. TabPFN-3: Technical report. *arXiv preprint arXiv:2605.13986*, 2026. URL https://arxiv.org/abs/2605.13986.

Songqiao Han, Xiyang Hu, Hailiang Huang, Minqi Jiang, and Yue Zhao. ADBench: Anomaly detection benchmark. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 35, pp. 32142–32159, 2022.

Shwai He, Tao Ge, Guoheng Sun, Bowei Tian, Xiaoyang Wang, and Dong Yu. Router-tuning: A simple and effective approach for dynamic depth. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 1925–1938, 2025.

Ahmed Heakl, Martin Gubri, Salman Khan, Sangdoo Yun, and Seong Joon Oh. Dr.LLM: Dynamic layer routing in LLMs. In *International Conference on Learning Representations*, 2026.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. An empirical analysis of compute-optimal large language model training. In *Advances in Neural Information Processing Systems*, volume 35, pp. 30016–30030, 2022. doi: 10.52202/068431-2176.

Noah Hollmann, Samuel Müller, Katharina Eggensperger, and Frank Hutter. TabPFN: A transformer that solves small tabular classification problems in a second. In *International Conference on Learning Representations (ICLR)*, 2023.

Noah Hollmann, Samuel Müller, Lennart Purucker, Arjun Krishnakumar, Max Körfer, Shi Bin Hoo, Robin Tibor Schirrmeister, and Frank Hutter. Accurate predictions on small data with a tabular foundation model. *Nature*, 637(8045):319–326, 2025.

Vasily Ilin, Peter Sushko, and Ranjay Krishna. DiScoFormer: Plug-in density and score estimation with transformers. In *Proceedings of the 43rd International Conference on Machine Learning (ICML)*, 2026. URL https://arxiv.org/abs/2511.05924.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

Narine Kokhlikyan, Vivek Miglani, Miguel Martin, Edward Wang, Bilal Alsallakh, Jonathan Reynolds, Alexander Melnikov, Natalia Kliushkina, Carlos Araya, Siqi Yan, and Orion Reblitz-Richardson. Captum: A unified and generic model interpretability library for PyTorch, 2020.

Weihao Kong and Abhimanyu Das. Introducing TabFM: A zero-shot foundation model for tabular data. Google Research Blog, June 2026.

Jaris Küken, Lennart Purucker, and Frank Hutter. Early stopping tabular in-context learning. In *ICML 2025 Workshop on Foundation Models for Structured Data*, 2025.

Juho Lee, Yoonho Lee, Jungtaek Kim, Adam Kosiorek, Seungjin Choi, and Yee Whye Teh. Set transformer: A framework for attention-based permutation-invariant neural networks. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 3744–3753. PMLR, 2019. URL https://proceedings.mlr.press/v97/lee19d.html.

Ziyue Li, Yang Li, and Tianyi Zhou. Skip a layer or loop it? learning program-of-layers in LLMs. In *Proceedings of the 43rd International Conference on Machine Learning (ICML)*, 2026. URL https://arxiv.org/abs/2606.06574.

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*, pp. 6035–6044, 2020. doi: 10.18653/v1/2020.acl-main.537.

Yijin Liu, Fandong Meng, Jie Zhou, Yufeng Chen, and Jinan Xu. Faster depth-adaptive transformers. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pp. 13424–13432, 2021.

Xuan Luo, Weizhi Wang, and Xifeng Yan. Adaptive layer-skipping in pre-trained LLMs. In *Conference on Language Modeling*, 2025.

Junwei Ma, Valentin Thomas, Guangwei Yu, and Anthony Caterini. In-context data distillation with TabPFN. In *ICLR 2024 Workshop on Mathematical and Empirical Understanding of Foundation Models (ME-FoMo)*, 2024. URL https://arxiv.org/abs/2402.06971.

Junwei Ma, Valentin Thomas, Rasa Hosseinzadeh, Alex Labach, Hamidreza Kamkari, Jesse C. Cresswell, Keyvan Golestan, Guangwei Yu, Anthony L. Caterini, and Maksims Volkovs. TabDPT: Scaling tabular foundation models on real data. In *Advances in Neural Information Processing Systems*, 2025. URL https://arxiv.org/abs/2410.18164.

Patryk Marszałek, Tomasz Kuśmierczyk, and Marek Śmieja. TACTIC for navigating the unknown: Tabular anomaly detection via in-context inference. *arXiv preprint arXiv:2603.14171*, 2026.

Catherine Olsson, Nelson Elhage, Neel Nanda, Nicholas Joseph, Nova DasSarma, Tom Henighan, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, Dawn Drain, Deep Ganguli, Zac Hatfield-Dodds, Danny Hernandez, Scott Johnston, Andy Jones, Jackson Kernion, Liane Lovitt, Kamal Ndousse, Dario Amodei, Tom Brown, Jack Clark, Jared Kaplan, Sam McCandlish, and Chris Olah. In-context learning and induction heads. *arXiv preprint arXiv:2209.11895*, 2022. doi: 10.48550/arxiv.2209.11895.

Jingang Qu, David Holzmüller, Gaël Varoquaux, and Marine Le Morvan. TabICL: A tabular foundation model for in-context learning on large data. In *International Conference on Machine Learning*, 2025. URL https://arxiv.org/abs/2502.05564.

Jingang Qu, David Holzmüller, Gaël Varoquaux, and Marine Le Morvan. TabICLv2: A better, faster, scalable, and open tabular foundation model. In *International Conference on Machine Learning*, 2026. URL https://arxiv.org/abs/2602.11139.

Kiran Ramnath, Kang Zhou, Sheng Guan, Soumya Smruti Mishra, Xuan Qi, Zhengyuan Shen, Shuai Wang, Sangmin Woo, Sullam Jeoung, Yawei Wang, Haozhu Wang, Han Ding, Yuzhe Lu, Zhichao Xu, Yun Zhou, Balasubramaniam Srinivasan, Qiaojing Yan, Yueyan Chen, Haibo Ding, Panpan Xu, and Lin Lee Cheong. A systematic survey of automatic prompt optimization techniques. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 33078–33110, 2025. doi: 10.18653/v1/2025.emnlp-main.1681.

Saul Santos, Nuno Gonçalves, Daniel C. McNamee, Marcos Treviso, and André F. T. Martins. Sparse attention as compact kernel regression. *arXiv preprint arXiv:2601.22766*, 2026.

Tal Schuster, Adam Fisch, Jai Gupta, Mostafa Dehghani, Dara Bahri, Vinh Q Tran, Yi Tay, and Donald Metzler. Confident adaptive language modeling. In *Advances in Neural Information Processing Systems*, volume 35, pp. 17456–17472, 2022.

Roy Schwartz, Gabriel Stanovsky, Swabha Swayamdipta, Jesse Dodge, and Noah A. Smith. The right tool for the job: Matching model and instance complexities. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 6640–6651, 2020. doi: 10.18653/v1/2020.acl-main.593.

Nour Shaheen, Junwei Ma, Alex Labach, Frank Hutter, Valentin Thomas, and Anthony L. Caterini. Understanding the surprising generalization properties of tabular foundation models. *arXiv preprint arXiv:2608.17957*, 2026.

Weiqiao Shan, Long Meng, Tong Zheng, Yingfeng Luo, Bei Li, Junxin Wang, Tong Xiao, and Jingbo Zhu. Early exit is a natural capability in transformer-based models: An empirical study on early exit without joint optimization, 2024.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc V Le, Geoffrey E Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *International Conference on Learning Representations (ICLR)*, 2017.

Yuchen Shen, Haomin Wen, and Leman Akoglu. FoMo-0D: A foundation model for zero-shot tabular outlier detection. *Transactions on Machine Learning Research*, 2025. ISSN 2835-8856.

Oscar Skean, Md Rifat Arefin, Dan Zhao, Niket Nikul Patel, Jalal Naghiyev, Yann LeCun, and Ravid Shwartz-Ziv. Layer by layer: Uncovering hidden representations in language models. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 55854–55875, 2025.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling LLM test-time compute optimally can be more effective than scaling parameters for reasoning. In *International Conference on Learning Representations (ICLR)*, pp. 10131–10165, 2025.

Qingyun Sun, Haonan Yuan, Yi Huang, Ziwei Zhang, Xingcheng Fu, Ruijie Wang, Haoyi Zhou, Jianxin Li, Jia Wu, and Philip S. Yu. A survey on foundation models for structured data: Tabular, time series, and graphs. *IEEE Transactions on Knowledge and Data Engineering*, 2026.

Vladimir Vapnik and Rauf Izmailov. Learning using privileged information: Similarity control and knowledge transfer. *Journal of Machine Learning Research*, 16(61):2023–2049, 2015.

Vladimir Vapnik and Akshay Vashist. A new learning paradigm: Learning using privileged information. *Neural Networks*, 22(5-6):544–557, 2009.

Jack Yi Wei and Narges Armanfard. ICLAD: In-context learning for unified tabular anomaly detection across supervision regimes. *arXiv preprint arXiv:2603.19497*, 2026.

Sang Michael Xie, Aditi Raghunathan, Percy Liang, and Tengyu Ma. An explanation of in-context learning as implicit bayesian inference. In *International Conference on Learning Representations (ICLR)*, 2022.

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*, pp. 2246–2251, 2020. doi: 10.18653/v1/2020.acl-main.204.

Xiyuan Zhang, Danielle C. Maddix, Junming Yin, Nick Erickson, Abdul Fatir Ansari, Boran Han, Shuai Zhang, Leman Akoglu, Christos Faloutsos, Michael W. Mahoney, Cuixiong Hu, Huzefa Rangwala, George Karypis, and Bernie Wang. Mitra: Mixed synthetic priors for enhancing tabular foundation models. In *Advances in Neural Information Processing Systems*, 2025.

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

APPENDIX

A EXTENDED RELATED WORK

Tabular Foundation Models (TFMs). Recent advances in tabular machine learning have shifted from dataset-specific training toward amortized in-context inference, establishing Tabular Foundation Models (TFMs) that leverage In-Context Learning (ICL) by training Transformers on diverse synthetic priors toward zero-shot inference on unseen tabular datasets in a single forward pass without gradient-based fine-tuning (Sun et al., 2026; Hollmann et al., 2023). For supervised classification and regression tasks, building on the pioneering TabPFN (Hollmann et al., 2023; 2025), recent advances include scalable inference in TabPFN-3 (Grinsztajn et al., 2026), TabICL/TabICLv2 (Qu et al., 2025; 2026), and TabFM (Kong & Das, 2026), diverse synthetic priors in Mitra (Zhang et al., 2025), and pretraining on real-world tables in TabDPT (Ma et al., 2025).

Beyond supervised tasks, recent research has successfully adapted the TFM paradigm to unsupervised and zero-shot tabular outlier detection (OD). FOMO-0D introduced the first pretrained outlier detection model by synthetic pretraining with Gaussian mixture priors, enabling zero-shot detection without additional training or hyperparameter tuning (Shen et al., 2025). Its successor OUTFORMER expanded this direction using diverse synthetic mixtures, including Gaussian mixture, structural causal, and copula-based priors, and introduced a self-evolving curriculum that adaptively allocates training effort across tasks (Ding et al., 2026b).

More recently, TACTIC (Marszałek et al., 2026) combines outlier-centric synthetic priors with contaminated contexts during pretraining to improve robustness to outliers in the context, while ICLAD unifies tabular outlier detection across unsupervised, one-class, and semi-supervised regimes via meta-learned task conditioning (Wei & Armanfard, 2026).

Compute-Adaptive Inference. To enable models to dynamically allocate compute, adaptive computation and early-exit mechanisms have emerged. These approaches can be broadly categorized into: (1) designing or training models for adaptive computation via recurrent execution, token-level sparsity, early exit, or layer skipping; and (2) retrofitting existing pretrained backbones through auxiliary modules or post-hoc routing policies.

At the architectural level, **natively adaptive designs** include Looped Transformers that vary inference-time compute by repeatedly applying shared Transformer blocks, with the number of iterations controlling effective depth (Giannou et al., 2023; Geiping et al., 2025), as well as Mixture-of-Experts (MoE) that adapt by employing a gating network to dynamically route each input token to only a small subset of specialized expert sub-networks, effectively decoupling massive parameter capacity from active per-token computational cost (Shazeer et al., 2017; Fedus et al., 2022; Du et al., 2022). Beyond structural designs, other backbones are often trained or fine-tuned with dynamic compute mechanisms through intermediate supervision and auxiliary prediction heads. Depth-Adaptive Transformer supervises intermediate predictions and learns halting policies (Elbayad et al., 2020), while Faster Depth-Adaptive Transformers estimate token-level depth requirements using mutual information or reconstruction loss (Liu et al., 2021). DeeBERT and FastBERT attach intermediate classifiers and use prediction uncertainty to determine when to exit (Xin et al., 2020; Liu et al., 2020), whereas PABEE stops when predictions stabilize across successive layers (Zhou et al., 2020). CALM combines intermediate-layer supervision with calibrated confidence thresholds for token-level early exiting (Schuster et al., 2022). LayerSkip incorporates layer dropout and early-exit supervision with a shared output head during pretraining, continual pretraining, or fine-tuning (Elhoushi et al., 2024).

To avoid costly backbone retraining to be adaptive, **post-hoc retrofitting** methods instead learn lightweight auxiliary modules. Evidence of early-exit capability without joint backbone optimization (Shan et al., 2024), together with the transferability of intermediate-layer representations (Skean et al., 2025), supports this direction. Router-Tuning trains lightweight routing modules (He et al., 2025), while FlexiDepth additionally trains adapters to compensate for representation changes caused by skipped layers (Luo et al., 2025). Dr.LLM and PoLaR extend the execution choices to skipping and repeating pretrained layers, learning routers or execution-program predictors from offline search while keeping the backbone frozen (Heakl et al., 2026; Li et al., 2026). For TFMs, Early Stopping Tabular In-Context Learning augments a frozen TabPFN backbone with separately trained intermediate decoders and entropy-based stopping (Küken et al., 2025).

Our work follows this post-hoc direction, learning a dataset-specific exit router for pretrained tabular outlier detectors while reusing the original prediction head at every exit and keeping all backbone and head parameters fixed.

Test-Time Optimization. While compute-adaptive inference targets time savings by minimizing compute dynamically based on input difficulty, test-time optimization typically aims to improve prediction performance by expending additional inference compute.

Test-time optimization encompasses strategies that dynamically refine inputs, prompts, or support sets during inference to maximize prediction quality. In Large Language Models (LLMs), test-time compute scaling and search-based prompt optimization allocate extra inference steps or refine prompt structures to elicit higher reasoning capabilities (Ramnath et al., 2025; Snell et al., 2025). In Tabular Foundation Models (TFMs), where raw input tables often exceed context limits or contain noisy features, context optimization methods refine support contexts to align inference inputs with

pretraining bounds. Soft prompt tuning techniques like TuneTables and In-Context Distillation optimize compact pseudo-contexts via gradient updates (Feuer et al., 2024; Ma et al., 2024), while hard context selection methods like VIP-COP perform black-box, sample- and feature-level attribution to isolate the most influential predictors (Chen et al., 2026).

While traditional test-time optimization trades increased latency for accuracy gains, our work demonstrates that performance enhancements and inference time reduction can be jointly realized in pretrained outlier detection models. By orchestrating compute-adaptive execution, our approach bypasses the conventional efficiency-performance trade-off.

B Technical Details

B.1 Planted-Sibling Experiment

Setup. In each real-world benchmark, we take datasets in a fixed random order and keep the first 100 that admit 50 targeted outliers (22 on ADBench). The targets are random query outliers at pairwise distances above 2δ, with δ defined below. Each sibling $\mathbf{x}_s = \mathbf{x}_q + 0.5\, r_q \mathbf{u}$, where r_q is the distance from $\mathbf{x}_q$ to its nearest context row and $\mathbf{u}$ is a random unit direction, replaces a random context inlier outside the ten nearest context rows of every target. All three backbones share this setup and the statistics below (Figure 5 for OutFormer; Figures 14 and 15 for ICLAD and TACTIC).

Influence score. We compute input gradients with automatic differentiation, following the saliency formulation implemented in Captum (Kokhlikyan et al., 2020). For a targeted query outlier $\mathbf{x}_q$ and its planted sibling $\mathbf{x}_s$, we define the influence score at exit layer k as

$$R_k(\mathbf{x}_q, \mathbf{x}_s) = \frac{\delta}{\sigma_k} \left\| \nabla_{\mathbf{x}_s} z_k(\mathbf{x}_q) \right\|_2, \tag{5}$$

where z_k is the outlier-class logit minus the inlier-class logit. The distance scale δ is the median Euclidean distance from query inliers to their nearest point in the clean context, measured in the preprocessed input space. The score scale σ_k is the standard deviation of z_k over all queries at layer k, computed separately for each context condition. These factors account for differences in input distances across datasets and score scales across layers and conditions. Under a first-order approximation, R_k is the maximum absolute change in the query score for a sibling perturbation of norm δ, expressed in units of σ_k. We take the median over the 50 targeted outliers within each dataset, then the median across datasets. We apply the same normalization to the other context rows shown in Figure 5; the nearest-neighbor comparison tracks the same row selected from the clean context before and after sibling planting.

For the detection curves in Figure 5, we compare the targeted outlier scores before and after sibling planting against a fixed reference: the scores of all query inliers under clean context at the same layer. This comparison isolates changes in the targeted outlier scores. We average these AUROC values across datasets; the plotted loss is the clean value minus the value after sibling planting. The reported Pearson correlation is computed across layers between this mean AUROC loss and the median sibling influence.

B.2 rOUT Inputs

Backbone representations. The router reads a random subset of up to 256 context and 1,024 query rows per dataset, which keeps its cost independent of dataset size. Each layer's representation is projected onto 64 principal components fitted on the training data and stored in int8 during training. In pilot experiments, routers trained on these compressed representations performed on par with routers trained on the full-precision ones, and halving the number of query rows read by the router lowered performance only slightly. Besides these representations, the router receives three groups of per-row features.

Raw features. Each row enters with its preprocessed input values, zero-padded to 100 dimensions.

Layer-wise outlier scores. We apply the backbone's scoring head at every layer, rank the resulting scores within each dataset and layer, and map the ranks through the inverse standard normal CDF. Context rows, which the backbone does not score, receive zeros.

Geometric features. We add 29 label-free statistics of each row's neighborhood (Table 6). They are measured relative to the context of the same dataset, which makes them comparable across datasets.

Table 6: Geometric features of each row. Distances are Euclidean in the preprocessed input space; context rows are scored leave-one-out.

Feature	Count	Definition
Neighbor distances	21	For each $k \in \{1, 2, 5, 10, 20, 50, 100\}$: log mean distance to the k nearest context rows, its percentile among context rows, and log mean distance to the k nearest other query rows
Distance to center	2	Standardized distance from the context mean, as a log value and as a percentile
Local density	4	For each $k \in \{5, 20\}$: local outlier factor and reverse-neighbor count
Boundary and duplicate	2	Fraction of features at the boundary of the quantile transform; whether an exact copy exists in the context

B.3 rOUT Training Objective

We first complete the routing objective, then define the two privileged-supervision terms, and finally describe the privileged-input schedule. The routing objective adds an entropy bonus, weighted by η, that keeps the selection distributions from collapsing onto a single layer; with H denoting entropy, the complete routing and privileged-supervision objectives are

$$\mathcal{L}_{\text{route}} = \mathcal{L}_{\text{seq}} + \alpha \mathcal{L}_{\text{layer}} - \frac{\eta}{L} \sum_{k=1}^{L} H(\mathbf{p}_k), \tag{6}$$

$$\mathcal{L}_{\text{PI}} = \beta \mathcal{L}_{\text{row}} + \gamma \mathcal{L}_{\text{sum}}. \tag{7}$$

The two privileged-supervision terms supervise the router at two granularities. The row-level term $\mathcal{L}_{\text{row}}$ asks the router to predict each query row's outlier label at every depth, so that its representations encode which rows are outliers. The distribution-level term $\mathcal{L}_{\text{sum}}$ asks the rows that the router weights as outliers to match the true outliers as a group, in the mean and spread of their backbone representations at every layer up to the current depth. It thus teaches the router where the outlier population lies in each layer's representation space, the information that exit selection relies on. Removing privileged information altogether costs up to 2.4% AUROC (Table 5).

Let $q_{k,i}$ be the router's predicted outlier probability for query row i at router depth k, $y_i \in \{0, 1\}$ its outlier label, and $\mathcal{S}$ the supervised query rows: all rows in the first training stage and the unrevealed rows in the second. The two privileged-supervision terms are

$$\mathcal{L}_{\text{row}} = -\frac{1}{L} \sum_{k=1}^{L} \frac{1}{|\mathcal{S}|} \sum_{i \in \mathcal{S}} \big[y_i \log q_{k,i} + (1 - y_i) \log(1 - q_{k,i}) \big], \tag{8}$$

$$\mathcal{L}_{\text{sum}} = \frac{1}{L} \sum_{k=1}^{L} \frac{1}{k} \sum_{l \le k} \frac{1}{2D} \Big(\Big\| \frac{\hat{\boldsymbol{\mu}}_{k,l} - \boldsymbol{\mu}_l}{\mathbf{s}_l^{\mu}} \Big\|^2 + \Big\| \frac{\hat{\boldsymbol{\sigma}}_{k,l} - \boldsymbol{\sigma}_l}{\mathbf{s}_l^{\sigma}} \Big\|^2 \Big), \tag{9}$$

where $\hat{\boldsymbol{\mu}}_{k,l}$ and $\hat{\boldsymbol{\sigma}}_{k,l}$ are the mean and standard deviation of the layer-l backbone representations of the query rows, weighted by $q_{k,i} / \sum_{i'} q_{k,i'}$, $\boldsymbol{\mu}_l$ and $\boldsymbol{\sigma}_l$ are the same statistics over the true query outliers, $\mathbf{s}_l^{\mu}, \mathbf{s}_l^{\sigma}$ are fixed per-component scales, $D = 64$ is the number of principal components, and division is elementwise. Only layers up to the router depth k enter $\mathcal{L}_{\text{sum}}$. Each term is zero for datasets without supervised rows or a valid outlier summary. We use $\alpha = 1$, $\beta = 0.05$, $\gamma = 1.5$, and $\lambda = 0.0025$ for all backbones, with $\eta = 0.02$ for OutFormer and ICLAD and $\eta = 0.06$ for TACTIC. The full objective is $\mathcal{L} = \mathcal{L}_{\text{route}} + \mathcal{L}_{\text{PI}}$.

Privileged-input schedule. Training has two stages (Table 7). The first stage (12 epochs) reveals all query labels and the outlier summary on every dataset. The second stage (8 epochs) warm-starts from the first and withdraws both inputs with cosine schedules: the probability that a dataset receives a privileged input decays from 0.75 to 0 along a half-cosine, and a dataset that receives query labels reveals a fraction of its query rows drawn from $U[0, 1]$. The summary is withdrawn over the first 30% of the second stage and the query labels over the next 40%, so the last 30% trains entirely without privileged inputs, as at deployment.

Training cost. On one H100 GPU, training a router takes 4.3 hours for OUTFORMER and 5.0 hours for ICLAD and TACTIC (Table 7), far less than the three days that pretraining OUTFORMER takes on four L40S GPUs (Ding et al., 2026b).

Table 7: Training recipe of ROUT.

	Stage 1 query labels shown	Stage 2 privileged inputs withdrawn
Epochs	12	8
Batch size (datasets)	64	32
Peak learning rate	5×10^{-4}	2×10^{-4}
Warmup steps	300	100
Training time (hours, one H100)		
OUTFORMER	2.5	1.8
ICLAD	3.0	2.0
TACTIC	3.0	2.0
Shared by both stages		
Architecture	two blocks of Sample and Layer Attention, $d = 256$, 4 heads, attention pooling	
Input	backbone representations and the per-row features of Appendix B.2	
Optimizer	AdamW, weight decay 0.02, gradient clip 1.0; the rate holds at the peak for 60% of the steps after warmup, then cosine to 10% of it	
At inference	stop at the first layer whose probability passes τ; τ and the epoch are chosen on synthetic validation data	

C EXPERIMENT DETAILS

C.1 SETUP

Evaluation benchmarks. We evaluate on three real-world benchmarks and one synthetic benchmark (Table 8): OddBench and OvRBench with 690 and 755 datasets, the 47 classical tabular datasets of ADBench, and SynBench with 800 datasets generated from the OUTFORMER priors. Our main protocol fills the context with nominal inliers only, which isolates the effect of the pollution we add explicitly. On ADBench, the context holds a random 70% of the inliers and the query set holds the remaining inliers and all outliers; for AUROC evaluation, datasets are upsampled with replacement to at least 1,000 rows and subsampled to at most 10,000, whereas computation and inference time (Tables 3 and 11) are measured on the full datasets. All backbones score each dataset in a single forward pass without context ensembling (the "w.o. Ens" setting of Ding et al. (2026b)), and TACTIC uses its polluted-context checkpoint except in Figure 15. For consistency with the synthetic data priors, which generate at most 100 features (Table 10), each backbone receives at most 100 features, and TACTIC at most 50, its input limit; for datasets with more features, we randomly select features up to this limit. Averages across benchmarks weight each benchmark by its number of datasets.

Pollution settings. Both settings replace context inliers with unlabeled query outliers or their siblings, so the context size is unchanged, and the ratios 1:4, 1:2, and 1:1 give the number of polluting context rows relative to the number of query outliers. In *held-out pollution*, we randomly split the query outliers of each dataset into two halves: one half stays in the query set at every ratio, and the other half is a pool from which 25%, 50%, or 100% is moved into the context. The query inliers are subsampled by the same factor, which keeps the query outlier rate unchanged; the clean columns of Tables 4 and 12 therefore use the full query set, and the polluted columns the retained half. In *near-duplicate pollution*, the query set is unchanged. We give 25%, 50%, or 100% of the query outliers one sibling each: in the space standardized by the context mean and standard deviation, the outlier is moved in a random direction by ρ times its distance to the nearest context row, with $\rho \sim U(0.1, 1)$, and the result is clipped to the range of the context.

Router pretraining corpus. ROUT is trained on 211,495 synthetic datasets, with 6,053 more for validation and 6,128 held out (Table 9). They come from two families of priors. The OUTFORMER priors generate outliers through five mechanisms: Gaussian mixtures, two copula variants, and two

structural causal model variants. The TabPFN priors produce classification datasets, which we turn into outlier detection tasks by relabeling one or several classes as outliers. We keep such a dataset only if its query outliers are separable from its query inliers, that is, a supervised kNN classifier with $k = 20$, evaluated leave-one-out on the labeled query rows, reaches an AUROC of at least 0.6. Each family contributes datasets with clean and with polluted context. Table 10 lists how each dataset is sampled, including its dimensionality, context size, and outlier rate and, for polluted context, the fraction of context rows replaced by outliers and the share of near-duplicates among them.

Table 8: Benchmarks used in this study.

			Median per dataset		
Benchmark	Tables	Datasets	rows	columns	anomalies
OddBench	real	690	7,504	6	9.2%
OvRBench	real	755	5,610	10	18.7%
ADBench	real	47	4,207	19	18.2%
SynBench	synthetic	800	3,202	49	25.8%

Table 9: Synthetic training corpus of ROUT.

Prior family	Mechanisms	Context	Train	Val	Held out
OutFormer priors	GMM, Copula-Dependence, Copula-Probability, SCM-Structure, SCM-Measurement	clean	17,298	1,000	999
		polluted	51,878	1,000	1,000
TabPFN priors	class relabeling: One-vs-Rest, One-vs-One, Many-vs-Rest	clean	71,376	2,037	2,081
		polluted	70,943	2,016	2,048
		Total	211,495	6,053	6,128

Table 10: Sampling parameters of the training corpus.

Hyperparameter	Values	Description
Each dataset		
d	$\mathrm{LogUniform}(2, 100)$, rounded	Dimensionality of data
N	5,000	Rows per dataset, context and query together
n_{ctx}	$U\{500, 750, \ldots, 4{,}750\}$	Context rows, a uniform draw from the grid with step 250; inliers only in the clean pools
n_{query}	$N - n_{\mathrm{ctx}}$	Query rows, 250 to 4,500, all of them scored
r	$\mathrm{Beta}(1, 4)$, capped at 0.5	Outlier rate among the query rows
Polluted context, in half the pools		
ε	$U(0, 0.40)$	Fraction of context rows replaced by outliers
s	0 or $U(0.1, 0.5)$, each with probability $\frac{1}{2}$	Fraction of the replaced rows that are near-duplicates
ρ	$U(0.1, 1.0)$	Near-duplicate offset, in units of the source outlier's distance to its nearest clean context row

C.2 Other Experiment Results

Oracle exit layers. For every backbone, the oracle exit layer varies widely across datasets. Figure 8 shows its per-layer histograms on each benchmark, which underlie the depth groups in Figure 3.

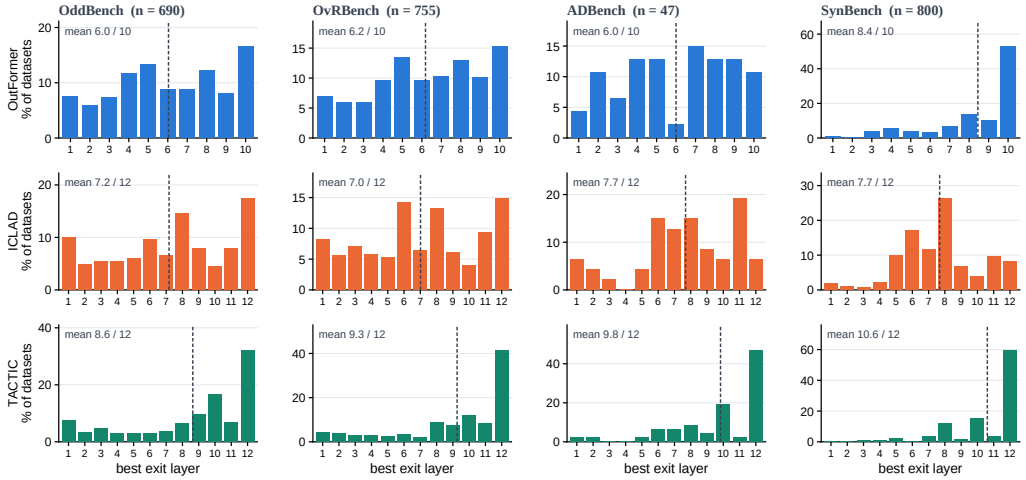

Figure 8: Distribution of the oracle exit layer over datasets for every backbone and benchmark; the dashed line marks the mean.

Context pollution across backbones. The pollution effect in Figure 4 holds for all three backbones: on the real-world benchmarks, the oracle's gain over full depth grows with the pollution ratio under both pollution modes. Figures 9 and 10 show held-out pollution for ICLAD and TACTIC, and Figures 11–13 show near-duplicate pollution for all three backbones.

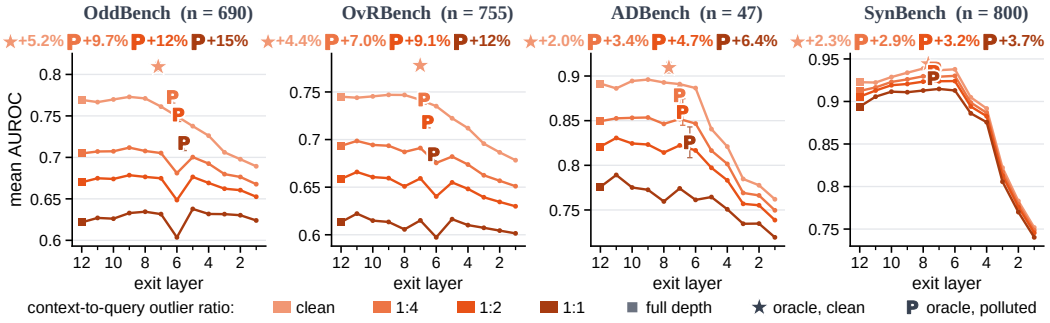

Figure 9: Per-layer performance of ICLAD under **held-out pollution**.

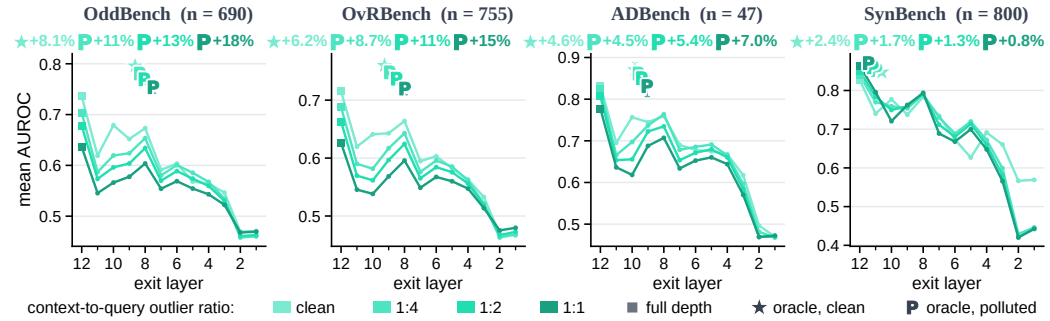

Figure 10: Per-layer performance of TACTIC under **held-out pollution**.

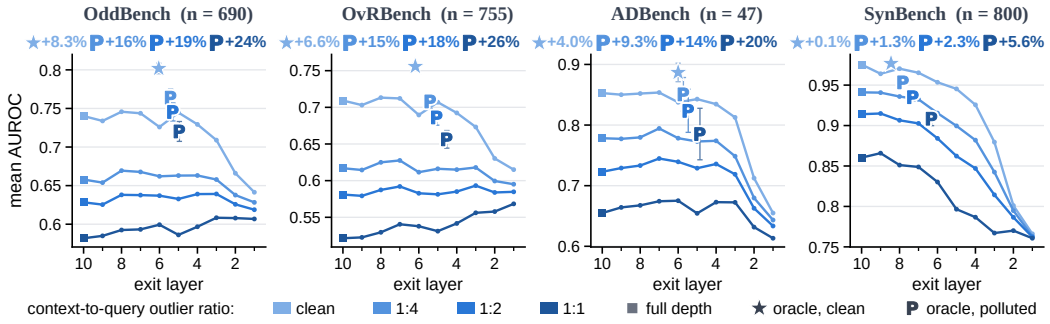

Figure 11: Per-layer performance of OUTFORMER under **near-duplicate pollution**.

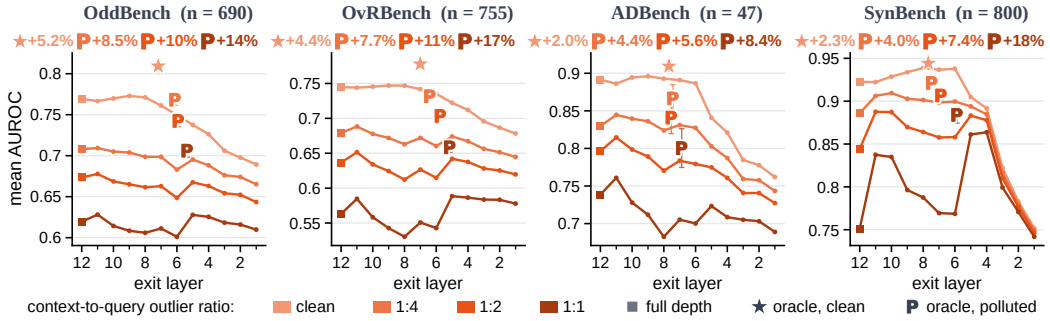

Figure 12: Per-layer performance of ICLAD under **near-duplicate pollution**.

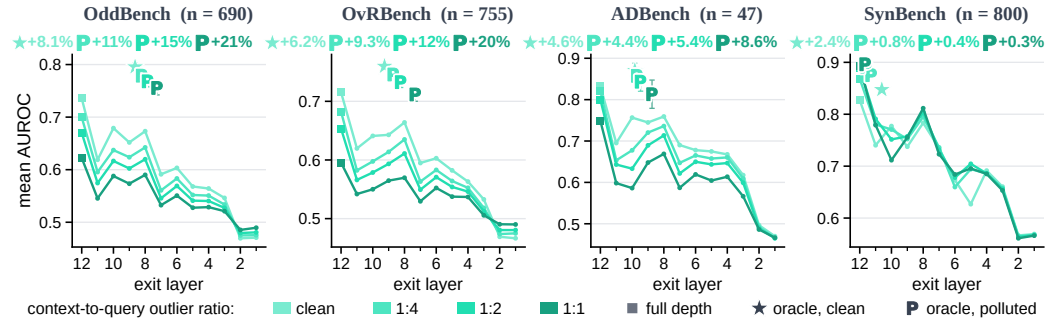

Figure 13: Per-layer performance of TACTIC under **near-duplicate pollution**.

Planted siblings across backbones. ICLAD and TACTIC follow the same mechanism as OUTFORMER, only at different depths (Figures 14 and 15). Sibling influence begins to rise after layer 5 in ICLAD but only in the last few layers of TACTIC, and in both backbones it tracks the AUROC loss of the targeted outliers across layers (Pearson $r = 0.93$ and 0.55, respectively).

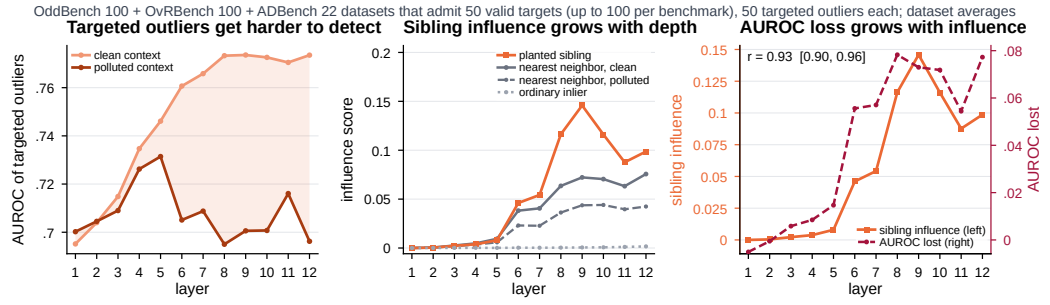

Figure 14: One planted sibling per targeted outlier (ICLAD).

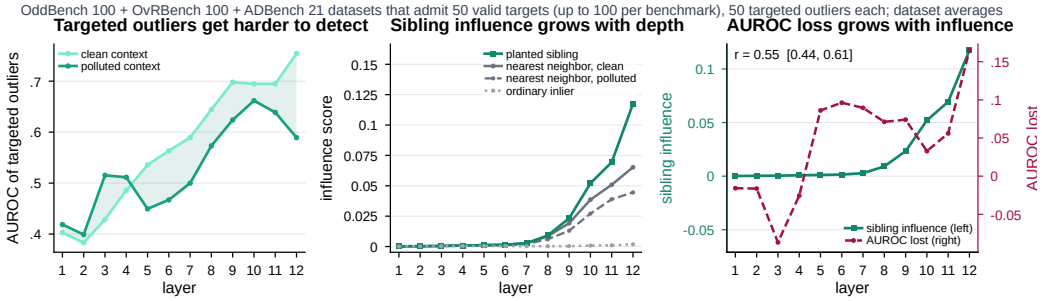

Figure 15: One planted sibling per targeted outlier (TACTIC, clean-context checkpoint).

Baseline and main results. On the real-world benchmarks, ROUT lies above the full LA-Entropy threshold sweep for every backbone (Figure 16). Figure 17 shows where ROUT sits between Full depth and the *Oracle* on each benchmark. Table 11 breaks the totals of Table 3 down into backbone computation, router calls, and additional features; the latter two account for at most 6.1% of ROUT's PFLOPs, so nearly all of the savings come from skipping backbone layers.

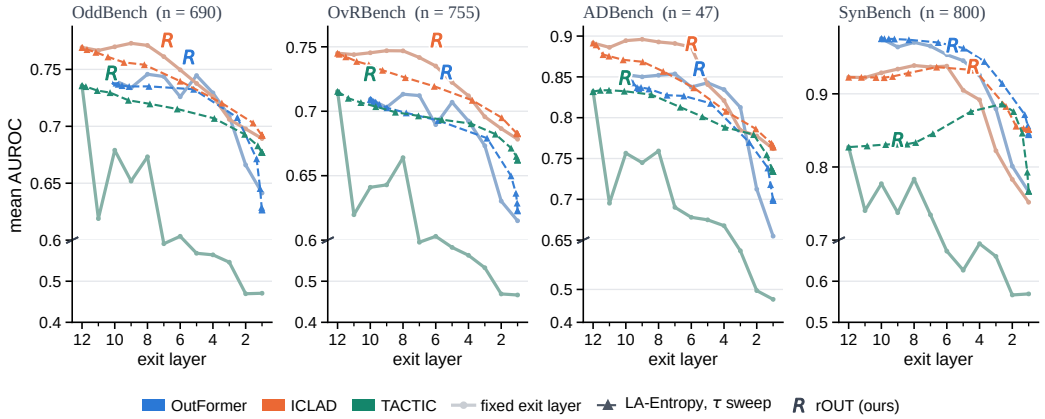

Figure 16: LA-Entropy over its full threshold sweep, with exit heads averaged over three seeds, compared with fixed exit layers and ROUT; its layer-specific heads lift TACTIC's early layers well above its fixed-layer curve.

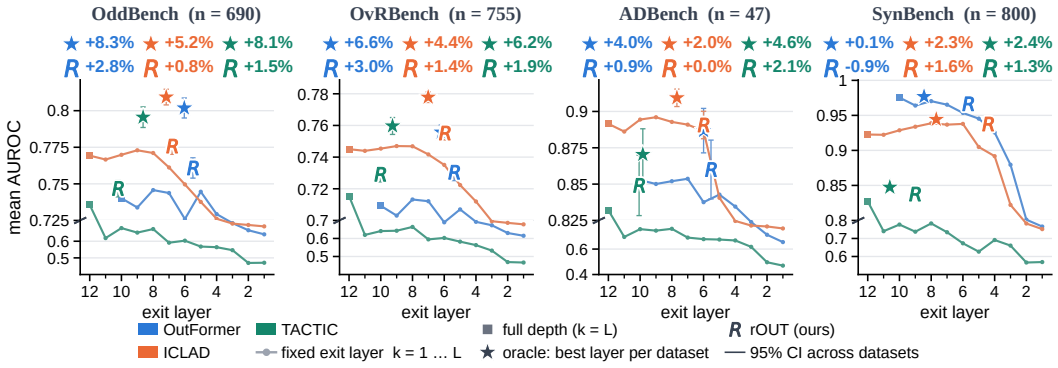

Figure 17: Mean AUROC of fixed exit layers, Full depth, the *Oracle*, and ROUT for every backbone and benchmark (Table 2).

Table 11: Full cost breakdown behind Table 3: totals over each benchmark on one H100.

Backbone		OddBench		OvRBench		ADBench	
		PFLOPs	Time (s)	PFLOPs	Time (s)	PFLOPs	Time (s)
OutFormer (L=10)	Full depth	2.29	111	3.07	146	0.21	9.9
	rOUT total	1.21	69.0	1.48	82.4	0.11	6.0
	backbone to exit	1.15	57.8	1.42	70.5	0.11	5.1
	router calls	0.051	7.6	0.055	8.1	0.003	0.5
	extra features	0.006	3.6	0.006	3.7	0.000	0.4
	Saving	47.2%	37.9%	51.7%	43.6%	47.9%	39.1%
ICLAD (L=12)	Full depth	2.04	70.6	2.64	89.9	0.18	6.4
	rOUT total	1.25	53.1	1.36	56.7	0.087	3.8
	backbone to exit	1.17	41.2	1.29	45.3	0.083	3.0
	router calls	0.070	8.1	0.065	7.7	0.003	0.5
	extra features	0.006	3.7	0.006	3.8	0.000	0.3
	Saving	38.9%	24.8%	48.3%	36.9%	51.1%	40.2%
TACTIC (L=12)	Full depth	2.71	183	3.64	246	0.25	17.0
	rOUT total	2.52	178	3.19	226	0.19	13.9
	backbone to exit	2.41	163	3.06	208	0.18	12.6
	router calls	0.11	12.3	0.12	14.7	0.006	0.9
	extra features	0.006	3.5	0.006	3.9	0.000	0.4
	Saving	6.7%	2.5%	12.4%	8.0%	23.9%	18.2%

rOUT **under context pollution.** Under both pollution modes, rOUT gains the most over full depth on OUTFORMER, less on ICLAD, and the least on TACTIC, which already exits close to full depth. Tables 12 and 13 give the full results for all three backbones, and Figure 18 is the held-out counterpart of Figure 7.

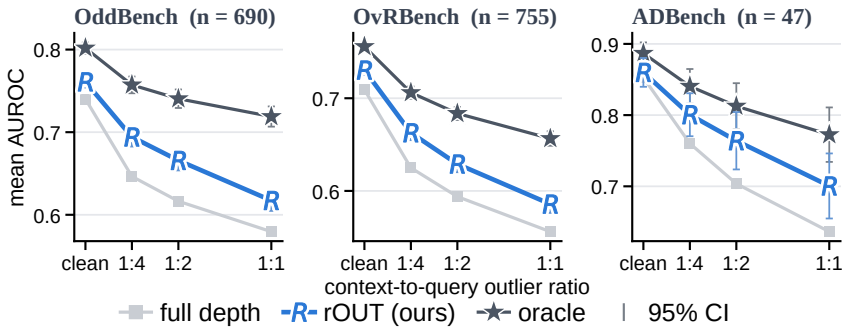

Figure 18: rOUT under **held-out pollution** improves over Full depth while exiting early (OUTFORMER).

Table 12: **Held-out pollution** on all three backbones. Columns give the context-to-query outlier ratio; the row under ROUT gives its relative AUROC gain over full depth.

Held-out Pollution		OddBench				OvRBench				ADBench			
Backbone	Method	Clean	1:4	1:2	1:1	Clean	1:4	1:2	1:1	Clean	1:4	1:2	1:1
OutFormer (L=10)	*Oracle*	.802	.757	.741	.719	.756	.706	.684	.657	.887	.841	.813	.773
	Best fixed layer	.746	.666	.637	.606	.713	.634	.601	.563	.854	.771	.726	.666
	Full depth	.740	.646	.616	.580	.709	.625	.594	.556	.853	.760	.704	.636
	Half depth	.745	.651	.622	.587	.707	.623	.594	.562	.843	.747	.694	.626
	LA-Entropy	.709	.657	.639	.612	.680	.627	.606	.579	.770	.723	.706	.689
	rOUT (ours)	**.761**	**.694**	**.666**	**.617**	**.730**	**.663**	**.629**	**.586**	**.860**	**.800**	**.764**	**.701**
	— AUROC gain	+2.8%	+7.4%	+8.0%	+6.5%	+3.0%	+6.0%	+5.9%	+5.4%	+0.9%	+5.3%	+8.6%	+10.1%
ICLAD (L=12)	*Oracle*	.809	.773	.752	.717	.778	.742	.718	.685	.909	.878	.860	.826
	Best fixed layer	.773	.712	.679	.638	.747	.699	.666	.622	.896	.854	.831	.789
	Full depth	.769	.705	.670	.622	.745	.693	.659	.613	.891	**.849**	.821	.776
	Half depth	.750	.681	.649	.604	.735	.675	.640	.597	.887	.847	.817	.761
	LA-Entropy	.704	.676	.659	.629	.696	.662	.639	.607	.786	.768	.753	.732
	rOUT (ours)	**.776**	**.724**	**.693**	**.640**	**.755**	**.706**	**.671**	**.624**	**.891**	.846	**.831**	**.785**
	— AUROC gain	+0.8%	+2.7%	+3.4%	+2.9%	+1.4%	+1.8%	+1.9%	+1.7%	+0.0%	−0.4%	+1.2%	+1.2%
TACTIC (L=12)	*Oracle*	.796	.781	.769	.754	.760	.748	.737	.720	.870	.862	.850	.831
	Best fixed layer	.736	.703	.678	.636	.715	.688	.663	.626	.832	.825	.807	.777
	Full depth	.736	.703	.678	**.636**	.715	.688	.663	**.626**	.832	**.825**	**.807**	**.777**
	Half depth	.603	.601	.589	.569	.603	.596	.584	.567	.678	.685	.671	.653
	LA-Entropy	.693	.657	.641	.615	.682	.646	.627	.599	.780	.760	.747	.727
	rOUT (ours)	**.747**	**.712**	**.686**	.634	**.729**	**.698**	**.670**	.622	**.849**	.813	.802	.767
	— AUROC gain	+1.5%	+1.2%	+1.2%	−0.3%	+1.9%	+1.4%	+1.1%	−0.6%	+2.1%	−1.4%	−0.6%	−1.3%

Table 13: **Near-duplicate pollution** on all three backbones. Columns give the context-to-query outlier ratio; the row under ROUT gives its relative AUROC gain over full depth.

Near-duplicate Pollution		OddBench				OvRBench				ADBench			
Backbone	Method	Clean	1:4	1:2	1:1	Clean	1:4	1:2	1:1	Clean	1:4	1:2	1:1
OutFormer (L=10)	*Oracle*	.802	.764	.745	.720	.756	.708	.686	.656	.887	.851	.823	.785
	Best fixed layer	.746	.669	.639	.608	.713	.628	.593	.568	.854	.794	.745	.675
	Full depth	.740	.658	.628	.582	.709	.617	.581	.521	.853	.778	.723	.655
	Half depth	.745	.663	.633	.586	.707	.616	.581	.531	.843	.773	.729	.654
	LA-Entropy	.709	.658	.635	.600	.680	.623	.601	.565	.770	.737	.717	.688
	rOUT (ours)	**.761**	**.710**	**.678**	**.635**	**.730**	**.658**	**.622**	**.575**	**.860**	**.819**	**.771**	**.732**
	— AUROC gain	+2.8%	+7.9%	+8.0%	+9.1%	+3.0%	+6.6%	+7.0%	+10.3%	+0.9%	+5.2%	+6.6%	+11.8%
ICLAD (L=12)	*Oracle*	.809	.768	.742	.706	.778	.732	.704	.659	.909	.867	.841	.800
	Best fixed layer	.773	.709	.678	.628	.747	.688	.651	.588	.896	.845	.815	.761
	Full depth	.769	.708	.673	.619	.745	.679	.637	.563	.891	.830	.796	.738
	Half depth	.750	.683	.649	.601	.735	.660	.615	.543	.887	.827	.779	.700
	LA-Entropy	.704	.673	.651	.615	.696	.656	.629	.591	.786	.761	.742	.719
	rOUT (ours)	**.776**	**.724**	**.693**	**.637**	**.755**	**.693**	**.655**	**.594**	**.891**	**.844**	**.806**	**.756**
	— AUROC gain	+0.8%	+2.3%	+2.9%	+2.9%	+1.4%	+2.0%	+2.9%	+5.4%	+0.0%	+1.8%	+1.3%	+2.3%
TACTIC (L=12)	*Oracle*	.796	.779	.768	.756	.760	.746	.733	.712	.870	.857	.843	.813
	Best fixed layer	.736	.700	.670	.622	.715	.682	.653	.595	.832	.821	.800	.749
	Full depth	.736	.700	.670	.622	.715	.682	.653	.595	.832	.821	.800	.749
	Half depth	.603	.583	.569	.551	.603	.583	.571	.552	.678	.665	.650	.619
	LA-Entropy	.693	.670	.651	.619	.682	.656	.636	.599	.780	.766	.757	.724
	rOUT (ours)	**.747**	**.716**	**.686**	**.645**	**.729**	**.692**	**.667**	**.616**	**.849**	**.822**	**.811**	**.759**
	— AUROC gain	+1.5%	+2.3%	+2.4%	+3.7%	+1.9%	+1.5%	+2.0%	+3.4%	+2.1%	+0.2%	+1.3%	+1.4%

C.3 Additional Ablations

Table 14 extends the component ablations of Section 5 to the training data. Each variant drops one data source while keeping the total number of training datasets fixed, and is evaluated at the full router's mean computation depth. Each source matters most in the setting it resembles: without polluted contexts, AUROC drops by 4.4% on polluted OvRBench; without the TabPFN priors, whose relabeled classes mirror how OvRBench defines outliers, it drops by 2.6% on clean OvRBench; and without the OutFormer priors, it drops by 2.3% on SynBench. Since no removal improves any other setting by more than 0.5%, we train on all sources.

Table 14: **Training-data ablations** of rOUT on OutFormer: relative AUROC change from the full router at matched depth. Poll. OvR averages both pollution modes at 1:1; shading marks each source's largest loss; stars as in Table 5.

Backbone: OutFormer		OvR	Poll. OvR	Syn
	rOUT (full)	.730	.580	.967
Data recipe	w/o polluted ctx.	+0.4%	$-4.4\%^{***}$	0.0%
	w/o TabPFN priors	$-2.6\%^{***}$	−0.5%	+0.1%
	w/o OutFormer priors	0.0%	+0.5%	$-2.3\%^{***}$

Table 15 ablates the depth penalty. Because the regret R_j is measured in AUROC, λ acts as the price of one more layer: at $\lambda = 0.0025$, the router goes deeper only when it expects the AUROC gain to outweigh this price. Without the penalty, extra layers cost nothing and an earlier layer can still be selected retrospectively, so the router runs close to full depth, computing 3.3–4.3 more layers for OutFormer and up to 7.3 more for ICLAD, while gaining at most 0.006 AUROC.

Table 15: **Depth penalty ablations**: rOUT with depth cost $\lambda = 0.0025$ (default) vs. $\lambda = 0$: AUROC and mean exit layer.

		OddBench		OvRBench		ADBench		SynBench	
Backbone	λ	AUROC	Layers	AUROC	Layers	AUROC	Layers	AUROC	Layers
OutFormer	0.0025	.761	5.5	.730	5.4	.860	5.5	.967	5.7
(L=10)	0	.761	8.8	.731	8.7	.856	9.2	.973	10.0
ICLAD	0.0025	.776	6.8	.755	6.0	.891	6.0	.938	4.4
(L=12)	0	.777	11.1	.757	11.0	.896	11.3	.939	11.7
TACTIC	0.0025	.747	10.2	.729	10.0	.849	10.1	.838	9.0
(L=12)	0	.745	11.6	.732	11.6	.852	11.4	.840	11.6