FROM UNFAMILIAR TO FAMILIAR: DETECTING PRE-TRAINING DATA VIA GRADIENT DEVIATIONS IN LARGE LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Pre-training data detection for LLMs is essential for addressing copyright concerns and mitigating benchmark contamination. Existing methods mainly focus on the likelihood-based statistical features or heuristic signals before and after fine-tuning, but the former are susceptible to word frequency bias in corpora, and the latter strongly depend on the similarity of fine-tuning data. From an optimization perspective, we observe that during training, samples transition from unfamiliar to familiar in a manner reflected by systematic differences in gradient behavior. Familiar samples exhibit smaller update magnitudes, distinct update locations, and more concentrated updates. Based on this insight, we propose GDS, a method that identifies pre-training data by probing Gradient Deviation Scores of target samples. Specifically, we first represent each sample using gradient profiles that capture the magnitude, location, and concentration of parameter updates across FFN and Attention modules, revealing consistent distinctions between member and non-member data. These features are then fed into a lightweight classifier to perform binary membership inference. Experiments on five public datasets show that GDS achieves state-of-the-art performance with improved cross-dataset transferability over strong baselines. Further analyses reveal distinct gradient distributions between members and non-members, validating our methodology.

1 INTRODUCTION

Large language models (LLMs) performance scales with the size and quality of pre-training data (Kaplan et al., 2020). As pre-training corpora expand to trillions of tokens and become increasingly proprietary and non-transparent, they raise significant concerns, such as unauthorized copyright, biased or harmful content, and contamination of evaluation benchmarks (Balloccu et al., 2024; Grynbaum & Mac, 2023). These challenges motivate a specialized task of membership inference attacks, named pre-training data detection: determining whether a given target sample was included in a model's pre-training corpus (Carlini et al., 2021).

Recent studies (Carlini et al., 2021; Li; Shi et al., 2024; Zhang et al., 2025) primarily use unsupervised likelihood-based statistical features to detect pre-training samples. For example, the MinK method (Shi et al., 2024; Zhang et al., 2025) examines the k% of tokens with the lowest predicted probabilities, assuming non-member texts contain more low-likelihood tokens. However, these approaches are susceptible to word frequency bias in pre-training corpora, specifically for rare words or short texts scenarios. To improve detection accuracy, other researchers (Zhang et al.; Choi et al.) propose some supervised methods to analyze heuristic signals before and after fine-tuning, leveraging the fact that fine-tuning impacts member and non-member samples differently. KDS (Choi et al.) shows that non-member datasets exhibit much larger embedding changes, but it only focuses on dataset-level detection. In contrast, FSD (Zhang et al.) observes a substantially larger loss reduction for non-member samples. Nevertheless, both methods rely on target-like probing data, limiting transfer under dataset shift. Therefore, achieving strong in-domain detection while retaining cross-dataset transferability remains an important challenge for pre-training data detection.

Optimization theory (Ruder, 2016; Frankle et al.; Wang et al., 2024) suggests that samples exhibit shifting gradient behavior as they become familiar, causing pre-training members and non-members

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

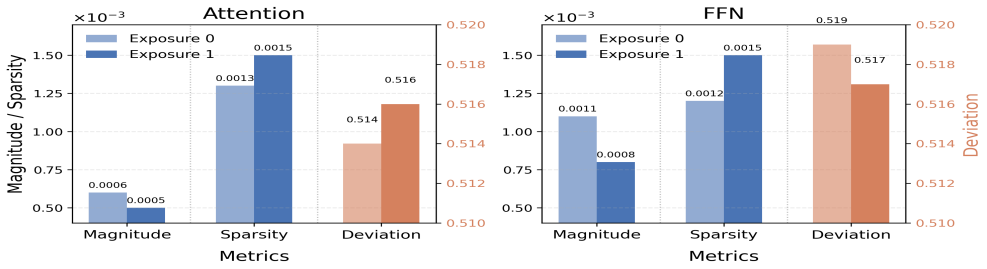


Figure 1: Gradient-difference statistics for target data in LLaMA-7B under zero (unfamiliar) and one (familiar) exposure. Magnitude measures overall parameter update magnitude, sparsity the proportion of updates below 1e-6, and deviation the distance of core updates from the LoRA center.

to induce distinct gradients in target LLMs. Motivated by this, we compare LLM's gradient behaviors on familiar(exposure=1) and unfamiliar(exposure=0) data [1], and observe clear differences in magnitude, concentration, and location, as illustrated in Figure 1 and Section 3. (1) **Decaying update magnitude.** Parameter updates are attenuated as data exposure grows, i.e., from 0.0011 to 0.0008 in FFN, indicating that familiar data leads to smaller update magnitudes as the model becomes less sensitive to repeatedly seen examples. (2) **Increasing update sparsity.** During training, the proportion of parameters with updates below the threshold(1e-6) increases, indicating that updates become more concentrated in a small subset of neurons. (3) **Gradual stabilization of update positional deviation.** The regions of parameter updates gradually shift and stabilize toward a consistent positional distribution, indicating the update pattern becomes increasingly structured.

In this paper, we propose GDS, a novel pre-training data detection method that infers member data by probing Gradient Deviation Scores of target samples. Specifically, we collect per-sample gradients across layers during backpropagation, encoding them as eight-dimensional features that capture the magnitude, location, and concentration of updates. Compared to non-members, member samples show smaller matrix-level and row-wise update magnitudes, lower FFN but higher attention eccentricity, lower variability in matrix-level and row-wise updates, a higher share of top gradients, and sparser neurons. Then, we extract discriminative statistics from the signals to build fixed-dimensional features, which are fed to a lightweight MLP for binary membership classification.

Extensive experiments on five benchmarks show that GDS consistently outperforms strong baselines across datasets and models, with improved cross-dataset transferability over strong supervised methods. Interpretability analyses reveal member/non-member gradient distributions, while semi-supervised experiments support practical detection scenarios. [2] Our contributions are as follows:

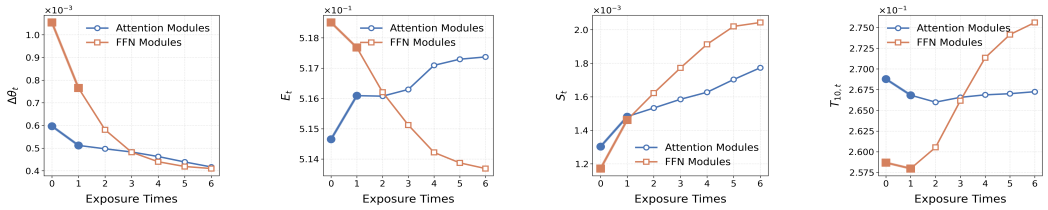
- We analyze exposure-wise parameter update dynamics as LLMs transition from unfamiliarity to familiarity with data, providing a new perspective for pre-training data detection.
- We propose gradient deviation features capturing magnitude, location, and concentration, and develop a gradient deviation score (GDS) for pre-training data detection.
- Extensive experiments validate GDS's effectiveness, robustness, interpretability, and practical utility for semi-supervised data detection.

2 Related Work

Unsupervised Likelihood-Based Methods. Early works distinguish member and non-member samples by capturing static statistical metrics in token-level likelihood distributions. Global-likelihood–based methods, such as PPL (Li), Zlib, and Lowercase (Carlini et al., 2021), perform discrimination by exploiting global probability features, compression entropy, and case-conversion ratios. MinK% (Shi et al., 2024) and MinK%++ (Zhang et al., 2025) mitigate global dependence by focusing on low-probability outlier tokens or target–candidate probability comparisons. DC-PDD (Zhang et al., 2024) approximates the word-frequency distribution of pre-training corpora

[1] Due to unavailable publish time, original LLaMA-7B is treated as unfamiliar model to unseen BookMIA data, while 1-epoch pre-training on them defines the familiar model.

[2] https://anonymous.4open.science/r/iclr-code-D0CA.



(a) $\Delta\theta_t$ (b) E_t (c) S_t (d) $T_{10,t}$

Figure 2: Dynamic curves of four parameter update characteristics defined in the Motivation section. Ordinate shows values of predefined characteristics, with orange and blue representing Attention and FFN modules. **Bold** denotes an obvious difference between exposure 0 and 1, which is also the most common pattern of difference between familiar and unfamiliar data.

using public datasets, but these references lack fidelity due to limited domain coverage. These unsupervised methods are effective but they often hit a performance ceiling in complex scenarios.

Supervised Fine-tuning or Activate Methods. Recent studies (Zhang et al.; Choi et al.) adopt lightweight fine-tuning or param activate to actively amplify asymmetric differences between member and non-member samples. KDS (Choi et al.) evaluates dataset contamination by comparing kernel similarity matrices of embeddings before and after fine-tuning, assuming non-member datasets undergo larger embedding changes than member datasets. FSD (Zhang et al.) detects membership by comparing loss reduction before and after fine-tuning, assuming non-member samples experience a larger drift than member samples. Another activate-based method NAPDD (Tang et al., 2025) builds reference activation statistics for member/non-member samples during forward propagation and detects membership via overlap of activated neuron indices. However, it only considers static positional information, ignoring activation strength and training dynamics.

3 Motivation

Inspired by optimization theory (Ruder, 2016; Bottou, 2010), we investigate how LLMs shift from unfamiliarity to familiarity with data and how this process relates to the dynamics of model parameters during training. To this end, we fine-tune LLaMA-7B with LoRA on the unseen split of BookMIA via the standard next-token prediction objective (consistent with pre-training), and track the resulting LoRA updates over 7 epochs. Exposure-wise analysis in Figure 2 shows the evolution of gradient features. We observe that models familiar with test data (exposure > 0)exhibit smaller, sparser, and more stable gradient updates, with this disparity becoming evident even after a single training epoch, which motivates GDS for membership detection.

3.1 Decay of Update Magnitude

Model training aims to minimize the loss function $L(\theta)$, with update magnitude determined by the gradient norm $\|\nabla L(\theta)\|$. According to optimization convergence theory (Ruder, 2016), as the parameters θ_t approach the optimum θ^*, the gradient norm of the loss function vanishes, i.e., $\lim_{\theta\to\theta^*} \|\nabla L(\theta_t)\| = 0$. This causes the parameter update magnitude to decrease synchronously. Therefore, we use $\Delta\theta_t$ as the core indicator of the **average update magnitude** across all trainable parameters at iteration t, where N denotes the total number of parameters:

$$\Delta\theta_t = \frac{1}{N}\sum_{i=1}^{N} |\theta_{t,i} - \theta_{t-1,i}| \,. \tag{1}$$

As training progresses, both the loss $L(\theta_t)$ and gradient norm $\|\nabla L(\theta_t)\|$ decrease, reducing the update magnitude $\Delta\theta_t$. Eventually, $L(\theta_t)$ stabilizes near its minimum, parameter updates $\Delta\theta_t$ become negligible. As shown in Figure 2a, the mean update magnitude decreases monotonically with training, with the largest changes occurring in the first two stages, consistent with loss convergence.

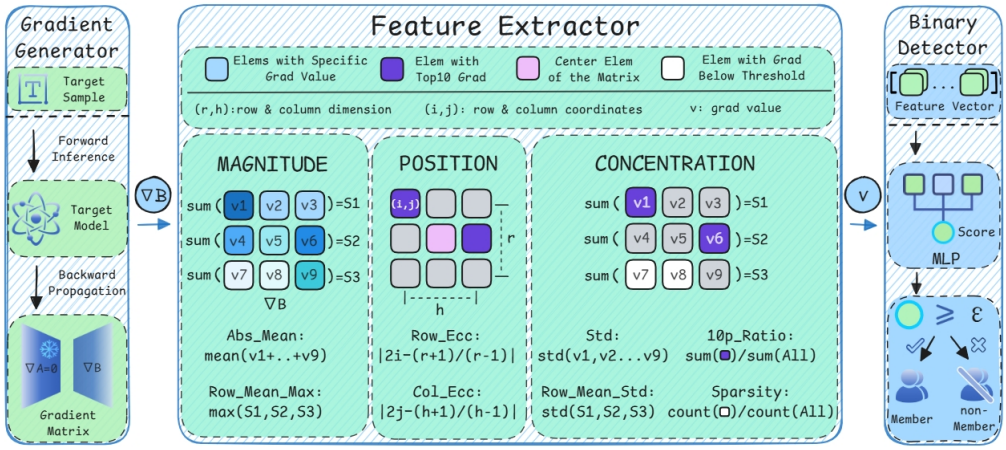


Figure 3: Overview of Gradient Deviation Scores method.

3.2 Gradually Stabilizing of Update Locations

Studies (Liu et al., 2024; Tang et al., 2025) have shown that neural activations occur at different locations when processing member and non-member samples. Inspired by this, we adopt the **parameter update eccentricity** as the core indicator, which quantifies the deviation of the top-10% parameter update positions from the global parameter centroid. We first define the global centroid as follows, where θ_i is the **coordinate** of the i-th parameter: $\theta_c = \frac{1}{N}\sum_{i=1}^{N}\theta_i$.

The update eccentricity at iteration t is then defined as follows, where $\mathrm{card}(\cdot)$ denotes the number of elements in a set, $U_{10,t} = \{\|\Delta\theta_{t,i}\|_1 \in \mathrm{Top10\%}(\|\Delta\theta_t\|_1)\}$, $\|\cdot\|_1$ denotes the ℓ_1 norm and $\|\cdot\|_2$ denotes the ℓ_2 norm:

$$E_t = \frac{1}{\mathrm{card}(U_{10,t})}\sum_{\theta_i \in U_{10,t}} \|\theta_i - \theta_c\|_2 . \tag{2}$$

In the initial training stage, activation patterns are unstable, leading to scattered $U_{10,t}$ and thus random large values of E_t. As training proceeds, the model's updates concentrate on a stable subset of parameters: θ_t stabilizes toward the optimum θ^*, and $U_{10,t}$ converges to a consistent positional region, resulting in E_t gradually stabilizing at a low level. Figure 2b shows that the eccentricity of parameter update evolves to a relatively stable value, consistent with the evolution of the activation pattern, suggesting that training drives updates toward a stable positional subset rather than scattered across the parameter space.

3.3 Increasing Update Sparsity

The Hessian spectrum (Gur-Ari et al., 2018; Frankle et al.) characterizes the curvature of the loss landscape, which governs the direction and efficiency of parameter updates. For loss $L(\theta)$, its Hessian matrix $\mathcal{H}(\theta) = \nabla^2 L(\theta)$ admits the eigenvalue decomposition, where U and Λ denote the orthogonal eigenvectors and eigenvalues, respectively. During training, the Hessian spectrum evolves from nearly uniform eigenvalues to a structure dominated by a few large values with most approaching zero:$\mathcal{H}(\theta) = U\Lambda U^T$.

This reshaping yields concentrated parameter updates, focusing on directions with large eigenvalues while leaving most others nearly unchanged, motivating our two joint indicators. S_t measures the fraction of parameters with negligible updates ($< 10^{-6}$) at iteration t, and $T_{10,t}$ denotes the fraction of total update magnitude contributed by the top 10% largest updates.

$$S_t = \frac{\mathrm{card}\left(i \mid |\Delta\theta_{t,i}| < 10^{-6}\right)}{N}, T_{10,t} = \frac{\sum_{|\Delta\theta_{t,i}| \in U_{10,t}} |\Delta\theta_{t,i}|}{||\Delta\theta_t||_1}. \tag{3}$$

As training progresses, parameter updates increasingly concentrate along core directions. Consequently, S_t gradually increases, while $T_{10,t}$ remains relatively large and stable(0.26–0.27), indicating that update energy is consistently dominated by a small subset of parameters. Figures 2c and 2d

confirm this growing sparsity and stable Top 10% contribution. Further details on the variation trends of the three metrics are provided in the Appendix B.

4 METHOD

4.1 TASK DEFINITION

Given a target text x and a target LLM f_θ pre-trained on a corpus D, the goal of the pre-training data detection task is to construct a detector $h(x, f_\theta)$ to determine whether x belongs to D. This task can be formalized as a binary classification problem: $h(x, f_\theta) \to \{0, 1\}$ where "1" indicates $x \in D$ as member and "0" indicates $x \notin D$ as non-member.

As shown in Figure 3, GDS comprises three stages: gradient matrix acquisition, feature vector extraction and lightweight Multilayer Perceptron(MLP) training. For each training sample, we perform inference and backpropagation to obtain LoRA gradients across l layers and m sub-modules, yielding $l \times m$ gradient matrices $\mathbb{G}$. From these matrices, we extract eight-dimensional features to form a gradient feature vector $\mathbb{V}^{l\times m\times 8}$, which is used to train an MLP classifier for binary membership prediction. During inference, the same feature extraction process is applied to a target sample x^*, and the resulting feature vector is fed into the trained MLP to produce the final classification result.

4.2 GRADIENT MATRIX ACQUISITION

Given the target model f_θ and training sample x, we first initialize f_θ with LoRA, yielding $l \times m$ LoRA matrices $\mathbb{L}$. We then feed a single sample x into f_θ for forward inference and backpropagation, obtaining LoRA gradient matrices $\mathbb{G} = \nabla_{\mathbb{L}}\mathcal{L}_{\text{CLM}}(f_\theta(x))$, where $\mathcal{L}_{\text{CLM}}$ denotes the causal language modeling loss of the pre-trained LLM. Since the LoRA_B matrices are fully initialized to zero, a single round of gradient propagation leads to zero gradients for the LoRA_A matrices. Thus, we only collect the gradient matrices of LoRA_B for subsequent processing.

4.3 FEATURE VECTOR EXTRACTION

Given gradients $\mathbb{G}$, we extract features based on the three parameter update trends(Section 3).

4.3.1 MAGNITUDE

We propose two magnitude indicators to measure the overall scale of gradient updates in the LoRA parameter space.

(1) Absolute Mean: The mean of the absolute values of all elements in the gradient matrix, which characterizes the overall parameter update strength, where $\mathbb{G}_{i,j}$ denotes the element in the i-th row and j-th column of the gradient matrix with r rows and h columns: $\text{Abs_Mean} = \frac{1}{r\cdot h}\sum_{i=1}^{r}\sum_{j=1}^{h}|\mathbb{G}_{i,j}|$.

(2) Max Row Mean: The mean of each row in the LoRA matrix and take the maximum value, which characterizes the most sample-responsive local optimal response dimension with the largest parameter update magnitude: $\text{Row_Mean_Max} = \max_{1\le i\le r}\left(\frac{1}{h}\sum_{j=1}^{h}|\mathbb{G}_{i,j}|\right)$.

4.3.2 POSITION

Screen the top 10% of gradient elements by absolute value from $\mathbb{G}$ and calculate their offset from this matrix center, capturing offset characteristics of core gradient positions for **(1) Row Eccentricity** and **(2) Column Eccentricity**, respectively, where $S = \{(i, j) \mid |\mathbb{G}_{i,j}| \in \text{Top10\%}(|\mathbb{G}|)\}$ denotes the set of indices for the top 10% of elements in the gradient matrix $\mathbb{G}$ ranked by absolute gradient value. Top gradients near the center correspond to an eccentricity of 0, while those near the edge correspond to a score of 1:

$$\text{Row_Ecc} = \frac{1}{\text{card}(S)}\sum_{(i,j)\in S}\left|\frac{2i-(r+1)}{r-1}\right|, \text{Col_Ecc} = \frac{1}{\text{card}(S)}\sum_{(i,j)\in S}\left|\frac{2j-(h+1)}{h-1}\right|. \quad (4)$$

4.3.3 CONCENTRATION

We propose four indicators to measure the concentration of gradient distributions.

(1) Top-10% Ratio: The ratio of the sum of the top 10% largest gradient magnitudes to the total update magnitude of the gradient matrix, which quantifies the contribution ratio of core gradients:

$$\text{10p_Ratio} = \frac{\sum_{(i,j)\in S} |\mathbb{G}_{i,j}|}{\|\mathbb{G}\|_1}. \tag{5}$$

(2) Sparsity: The proportion of gradient elements with absolute values less than 10^{-6}, quantifying the sparsity degree of gradient distributions:

$$\text{Sparsity} = \frac{\text{card}\left(\{(i,j) \mid |\mathbb{G}_{i,j}| < 10^{-6}\}\right)}{r \cdot h}. \tag{6}$$

(3) Standard Deviation: The standard deviation of the elements in the gradient matrix, which quantifies the dispersion degree of parameter updates:

$$\text{Std} = \sqrt{\frac{1}{r \cdot h} \sum_{i=1}^{r} \sum_{j=1}^{h} (|\mathbb{G}_{i,j}| - \text{Abs_mean})^2}. \tag{7}$$

(4) Row Mean Standard Deviation: The mean of each row in the LoRA matrix and take the standard deviation, which quantifies the consistency of update strength across all rows, where $\bar{\mathbb{G}}_i = \frac{1}{h}\sum_{j=1}^{h} |\mathbb{G}_{i,j}|$ denotes the i-th row's absolute mean, $\mu_{\bar{\mathbb{G}}} = \frac{1}{r}\sum_{i=1}^{r} \bar{\mathbb{G}}_i$ is the mean of row means:

$$\text{Row_Mean_Std} = \sqrt{\frac{1}{r} \sum_{i=1}^{r} \left(\bar{\mathbb{G}}_i - \mu_{\bar{\mathbb{G}}}\right)^2}. \tag{8}$$

We compute the aforementioned eight feature values from all gradient matrices to derive a feature vector $\mathbb{V}^{l\times m\times 8}$ per sample, which is then used for subsequent MLP training and inference.

4.4 LIGHT MLP TRAINING

Collecting the feature vectors $\mathbb{V}_i$ from all training samples into a set $\mathbb{V}_s$, we feed them into a lightweight MLP for training. The target output corresponds to the binary label $\{0, 1\}$ for each sample, where 0 denotes non-member and 1 denotes member. The training loss is defined as follows, where N denotes the number of training samples, $y_i \in \{0, 1\}$ is the ground-truth label of the i-th sample, and $\hat{y}_i$ is the predicted probability output by the MLP:

$$\mathcal{L} = -\frac{1}{N} \sum_{i=1}^{N} \left[y_i \log \hat{y}_i + (1 - y_i) \log(1 - \hat{y}_i) \right]. \tag{9}$$

5 EXPERIMENTS

5.1 EXPERIMENTAL SETUP

Datasets Following prior work, we evaluate on five prevalent datasets: WikiMIA, BookMIA (Shi et al., 2024), ArxivTection, BookTection (Duarte et al.), and Mimir (Duan et al.), a widely recognized challenging benchmark for pre-training data detection.

Target Models We evaluate five open-source LLMs with diverse architectures: Neo-2.7B (Black et al., 2021), GPT-J-6B (Wang & Komatsuzaki, 2021), OPT-6.7B (Zhang et al., 2022), Pythia-6.9B (Biderman et al., 2023), and LLaMA-7B (Touvron et al., 2023) from Hugging Face.

Comparison Methods We select seven representative state-of-the-art baselines. Three unsupervised methods include PPL (Carlini et al., 2021), MinK (Shi et al., 2024), and MinK++ (Zhang

Table 1: Comparison (AUROC/TPR@5%FPR) on 4 Datasets across 5 target models. Target models are denoted as 2.7B/6B/6.7B/6.9B/7B, corresponding to Neo-2.7B/GPT-J-6b/OPT-6.7b/Pythia-6.9b/LLaMA-7B respectively. Results are averaged over three experimental runs.

Method	WikiMIA					ArxivTection				
	2.7B	6B	6.7B	6.9B	7B	2.7B	6B	6.7B	6.9B	7B
Unsupervised										
PPL	0.61/0.13	0.64/0.13	0.60/0.12	0.63/0.13	0.69/0.14	0.62/0.13	0.64/0.13	0.60/0.12	0.63/0.13	0.70/0.14
MinK	0.65/0.17	0.67/0.19	0.63/0.15	0.67/0.19	0.73/0.18	0.65/0.17	0.68/0.19	0.63/0.15	0.67/0.19	0.72/0.18
MinK++	0.67/0.15	0.69/0.19	0.65/0.11	0.70/0.18	0.82/0.22	0.67/0.15	0.69/0.19	0.65/0.10	0.70/0.17	0.83/0.22
Supervised										
NLL	0.67/0.16	0.69/0.21	0.62/0.16	0.69/0.21	0.69/0.16	0.85/0.43	0.87/0.44	0.76/0.23	0.86/0.45	0.80/0.31
KDS	0.79/0.29	0.66/0.17	0.78/0.22	0.65/0.17	0.63/0.17	0.85/0.56	0.67/0.15	0.91/0.54	0.64/0.16	0.73/0.17
NAPDD	0.82/0.38	0.83/0.46	0.84/0.46	0.81/0.39	0.83/0.46	0.82/0.44	0.96/0.79	0.91/0.52	**0.95**/0.76	0.94/0.69
FSD	**0.92/0.77**	**0.95/0.78**	0.90/0.63	0.90/**0.66**	0.92/0.41	0.91/0.65	0.96/0.79	0.89/0.63	**0.95**/0.66	0.94/0.81
Ours	0.90/0.60	0.93/0.66	**0.94/0.67**	**0.92**/0.63	**0.96/0.84**	**0.94/0.73**	**0.97/0.86**	**0.94/0.75**	**0.95/0.83**	**0.97/0.85**

Method	BookTection					BookMIA				
	2.7B	6B	6.7B	6.9B	7B	2.7B	6B	6.7B	6.9B	7B
Unsupervised										
PPL	0.69/0.15	0.74/0.25	0.64/0.13	0.73/0.25	0.71/0.25	0.29/0.02	0.22/0.13	0.23/0.01	0.59/0.19	0.56/0.21
MinK	0.71/0.18	0.75/0.27	0.67/0.15	0.74/0.26	0.71/0.25	0.45/0.05	0.60/0.20	0.40/0.03	0.62/0.20	0.60/0.21
MinK++	0.64/0.13	0.67/0.18	0.61/0.11	0.66/0.19	0.63/0.15	0.54/0.13	0.64/0.29	0.49/0.09	0.60/0.23	0.58/0.20
Supervised										
NLL	0.76/0.25	0.82/0.34	0.71/0.18	0.78/0.30	0.76/0.30	0.90/0.61	0.88/0.55	0.92/0.66	0.80/0.42	0.79/0.43
KDS	0.92/0.63	**0.97**/0.85	**0.98**/0.88	0.81/0.38	0.97/0.86	0.77/0.27	0.87/0.66	**0.99**/0.95	0.67/0.17	0.89/0.51
NAPDD	0.91/0.66	0.90/0.72	0.96/0.80	0.91/0.70	0.93/0.78	0.98/0.93	0.98/0.94	0.97/0.93	0.97/0.92	**0.99**/0.94
FSD	0.92/0.53	0.91/0.52	0.96/0.77	0.93/0.59	0.92/0.55	0.98/0.93	0.97/0.89	0.98/0.96	0.98/0.93	0.98/0.91
Ours	**0.96/0.84**	**0.97/0.88**	**0.98/0.92**	**0.96/0.83**	**0.98/0.92**	**0.99/0.98**	**0.99/0.99**	**0.99/0.99**	**0.99/0.98**	**0.99/0.99**

et al., 2025). The supervised method includes KDS, NAPDD, FSD; NLL denotes the use of three unsupervised metrics as features for MLP classification. More details are shown in Appendix A.

Evaluation Metrics Two metrics evaluate the binary pre-training data detection task: **AUROC** measures overall discriminative performance, and **TPR@5%FPR** captures practical effectiveness by assessing detection rate under a 5% false positive constraint.

Implementation Details We use the PEFT library (Mangrulkar et al., 2022) for LoRA-based (Hu et al., 2022) gradient feature extraction without parameter updates. All supervised methods are trained on 30% of the data, with the remaining 70% used for inference evaluation under FSD settings. We adopt random data splitting and repeat each experiment three times, taking the average of the results. More Details can be seen in Appendix A.8.

5.2 Main Results

Table 1 reports results on four datasets. On WikiMIA with LLaMA-7B, GDS reaches an AUROC of 96%, outperforming the best baseline FSD by 4.3% and substantially surpassing score-based methods such as MinK++. GDS performs best in most settings, with particularly large TPR@5%FPR gains on BookTection and BookMIA, i.e., on BookTection with LLaMA-7B, the improvement is nearly 67.3% over FSD. GDS also remains the most stable across models, with AUROC variation mostly within 0.04, indicating robust performance across diverse model architectures. Table 2 reports results on four Mimir subsets. Despite performance drops caused by similar data distributions, GDS delivers the best average gains across models, with notable advantages on PubMed Central, DM Mathematics, and GitHub. With LLaMA, these three subsets reach AUROCs of 0.82, 0.95, and 0.91. The results for all subsets of Mimir can be found in the Appendix D.1.

5.3 Ablation Study

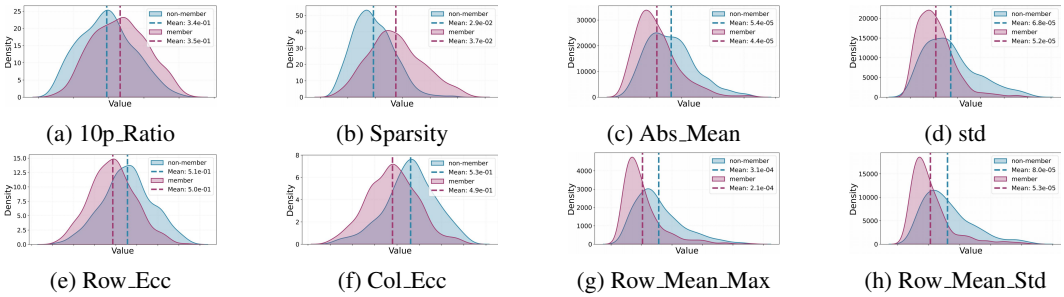
We perform two ablation studies: (1) removing each feature category individually, and (2) using features from only the Attention or FFN module. Table 3(a) shows that all categories help, with Position Offset contributing most; the full feature set performs best, confirming their complementarity

Table 2: Comparison (AUROC) on 4 Subsets of Mimir across 5 target models. Target models are denoted as 2.7B/6B/6.7B/6.9B/7B, corresponding to Neo-2.7B/GPT-J-6b/OPT-6.7b/Pythia-6.9b/LLaMA-7B respectively. Results are averaged over three experimental runs.

Method	GitHub					PubMed Central					ArXiv					DM Mathematics				
	2.7B	6B	6.7B	6.9B	7B	2.7B	6B	6.7B	6.9B	7B	2.7B	6B	6.7B	6.9B	7B	2.7B	6B	6.7B	6.9B	7B
Unsupervised																				
PPL	0.78	0.79	0.78	0.90	0.83	0.78	0.79	0.78	0.78	0.72	**0.78**	**0.79**	0.65	**0.78**	0.70	0.78	0.79	0.65	0.91	0.31
MinK	0.88	0.89	0.76	0.90	0.83	0.81	0.80	0.74	0.79	0.72	**0.78**	**0.79**	0.65	**0.78**	0.70	0.93	0.93	0.92	0.92	0.32
MinK++	0.84	0.86	0.56	0.86	0.73	0.71	0.72	0.60	0.70	0.59	0.60	0.64	0.53	0.70	0.57	0.77	0.79	0.67	0.75	0.22
Supervised																				
NLL	0.89	0.90	0.80	0.90	0.85	0.80	0.81	0.74	**0.80**	0.73	0.74	0.77	0.71	0.75	0.71	0.91	0.92	0.93	0.93	0.92
KDS	0.74	0.76	0.70	0.77	0.70	0.67	0.66	0.71	0.68	0.67	0.69	0.69	0.69	0.68	0.65	0.79	0.80	0.85	0.80	0.86
NAPDD	0.89	0.88	0.89	0.91	0.90	0.74	0.79	0.76	0.77	0.75	**0.78**	0.77	0.77	0.76	**0.79**	0.91	0.91	0.90	0.92	0.90
FSD	0.77	0.80	0.62	0.77	0.72	0.71	0.79	0.63	0.56	0.63	0.72	0.78	0.55	0.70	0.52	0.58	0.85	0.60	0.74	0.53
Ours	**0.90**	**0.92**	**0.88**	**0.92**	**0.91**	**0.84**	**0.85**	**0.84**	0.79	**0.82**	**0.78**	**0.79**	**0.78**	**0.78**	0.77	**0.95**	**0.95**	**0.94**	**0.95**	**0.95**

Table 3: Ablation results using AUROC and TPR@5%FPR on WikiMIA and ArxivTection. (a) Removing each feature category individually; (b) using features from only Attention or FFN module. "-" denotes feature removal.

	(a) Feature Category Ablation			(b) Module Ablation		
Dataset	-Magnitude	-Concentrate	-Position	ATT-Only	FFN-Only	Origin
WikiMIA	0.94/0.72	0.94/0.67	0.93/0.65	0.94 / 0.68	0.93 / 0.64	0.96 / 0.84
ArxivTection	0.96/0.82	0.95/0.79	0.95/0.77	0.95 / 0.76	0.93 / 0.72	0.97 / 0.85



(a) 10p_Ratio (b) Sparsity (c) Abs_Mean (d) std

(e) Row_Ecc (f) Col_Ecc (g) Row_Mean_Max (h) Row_Mean_Std

Figure 4: Distribution differences of eight gradient features between member(red) and non-member(blue) samples. Dashed lines indicate distribution means. The x-axis shows feature values, and the y-axis shows probability density.

(see Appendix C). Table 3(b) shows that Attention features are more informative than FFN features, but both single-module variants lag behind the full model, indicating that Attention and FFN capture complementary gradient patterns and are best used together.

5.4 Analysis

Feature Distribution Figure 4 illustrates the eight features with the largest member/non-member discrepancies across all layers and sub-modules. Member samples show lower Abs_Mean and Row_Mean_Max, higher Sparsity, lower Std and Row_Mean_Std, and higher 10p_Ratio, indicating smaller, more concentrated, and more stable gradients. They also have lower Row_Ecc and Col_Ecc, suggesting that the top-gradient positions of member samples occupy a more concentrated and less edge-dispersed region, rather than being uniformly spread across rows and columns.

Table 4: Additional analysis results. (a) Cross-dataset transfer performance (AUROC); (b) Temporal feature perturbation on WikiMIA (AUROC).

(a) Cross-dataset

Dataset	NLL	KDS	NAPDD	FSD	Ours
Wiki (Arxiv)	0.67	0.60	**0.72**	0.52	0.67
Arxiv (Wiki)	0.61	0.56	0.59	0.58	**0.70**
Mix	0.76	0.70	0.86	0.92	**0.95**

(b) Temporal perturbation

	NLL	KDS	NAPDD	FSD	Ours
Origin	0.69	0.63	0.83	0.92	**0.96**
Deletion	0.64	0.62	0.76	0.76	**0.84**

Table 5: Semi-supervised performance (AUROC / TPR@5%FPR) on WikiMIA.

	NLL	KDS	NAPDD	MinK++	FSD	GDS
LLaMA-7B	0.59/0.13	0.59/0.08	0.85/0.38	0.82/0.22	0.79/0.134	**0.86**/0.40
Pythia-6.9B	0.63/0.12	0.71/0.16	0.78/0.31	0.70/0.18	0.76/0.18	**0.80**/0.34
GPT-J-6B	0.65/0.16	0.74/0.20	0.77/0.24	0.69/0.19	0.75/0.14	**0.79**/0.26
OPT-6.7B	0.58/0.15	0.78/0.25	0.75/0.33	0.65/0.11	0.70/0.11	**0.81**/0.37
GPT-Neo-2.7B	0.58/0.15	0.69/0.15	0.76/0.23	0.67/0.15	0.76/0.17	**0.78**/0.26

Transferability Although GDS uses a unified feature extraction pipeline, the current detector remains dataset-dependent. We therefore evaluate direct cross-dataset transfer between WikiMIA and ArxivTection, as well as a unified classifier trained on their mixture. As shown in Table 4, GDS achieves AUROCs of 0.67 and 0.70 under direct transfer, outperforming FSD (0.52 and 0.58), but still exhibiting a substantial drop relative to in-domain evaluation. In contrast, training a unified classifier on mixed-domain data yields an AUROC of 0.95. These results suggest that the gradient features transfer better than those used by the supervised fine-tuning baseline, while the learned decision boundary remains sensitive to dataset shift.

WikiMIA Modification Dataset partitioning can introduce distribution shifts or dataset-specific artifacts. For example, WikiMIA is split by publication time, with all non-member samples labeled as 2023. To ensure our method does not exploit such artifacts, we remove all timestamp tokens and reevaluate all methods. As shown in Table 4, although token removal degrades performance, our method suffers a much smaller drop than the strongest baseline, FSD, and still achieves SOTA results. Furthermore, GDS remains effective in low-resource settings, sensitivity analysis, computational efficiency and full-parameter results(see Appendix D.2, D.3, D.4 and D.5, respectively).

Semi-Supervised Setting To test GDS in label-free real-world settings, we combine it with the SOTA unsupervised method MinK++. MinK++ first filters WikiMIA samples using thresholds of -1.4 and -4.6, producing 310 pseudo-labeled samples for training GDS and FSD. The results in Table 5 show semi-supervised GDS outperforms unsupervised MinK++ and semi-supervised FSD, demonstrating excellent noise resistance and practical value. Notably, GDS has a distinct advantage in TPR@5%FPR, reflecting strong positive sample recall under extremely low false positive tolerance. This indicates GDS can more accurately and robustly identify pre-trained data, suppress non-member high-score interference, select purer member samples in high-confidence regions, and thus has stronger deployment value in low-false-alarm, high-reliability scenarios.

6 Conclusion

We revisit pre-training data detection from an optimization and training-dynamics perspective and introduce GDS for pre-training membership inference. By analyzing model training dynamics, we show that member and non-member samples exhibit stable and interpretable gradient differences in magnitude, location, and concentration across datasets and architectures. Experiments across five datasets and five model architectures demonstrate that GDS consistently outperforms competitive baselines and remains robust across different backbone models and dataset transfer. Future work will explore more stable training dynamics to achieve better cross-data transferability.

AI USE STATEMENT

In this work, we used generative AI tools for limited code assistance. We have not used generative AI tools for generating research ideas, designing the methodology, conducting the core experiments, analyzing or interpreting results, or writing the core scientific claims, and the remaining required disclosure tasks are not applicable to this work. Additionally, we used generative AI tools for manuscript content checking , such as checking grammar, clarity, terminology consistency, and internal consistency. We have reviewed all AI-assisted work. Specifically, the LLM-assisted code was tested and verified by the authors for correctness, and the LLM-assisted content checks were manually reviewed by the authors, with any suggested edits accepted only after author validation. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work investigates pre-training data detection, a form of membership inference that determines whether a given sample was included in an LLM's pre-training corpus. While our proposed GDS method improves detection accuracy and generalization without fine-tuning, it also raises important ethical concerns. Pre-training data detection methods could be misused to probe proprietary or confidential training corpora, potentially revealing sensitive information about data sources or collection practices. Although GDS does not reconstruct or expose training samples directly, it contributes to capabilities that may be exploited if applied irresponsibly. Our primary motivation is to support transparency, accountability, and safety. As pre-training datasets become increasingly large and opaque, tools like GDS can help researchers, auditors, and regulators verify data usage claims, identify potential copyright violations, and detect benchmark contamination. In this sense, GDS is intended as a diagnostic and auditing mechanism rather than a data extraction attack. We release code and models to promote reproducibility while encouraging responsible deployment. Overall, we hope this work advances trustworthy and transparent large-scale model development while highlighting the need for careful governance of pre-training data.

REPRODUCIBILITY STATEMENT

All implementations of the methods described in this paper will be made publicly available at `https://anonymous.4open.science/r/iclr-code-D0CA/`. The repository includes the complete implementation pipeline for every component in the Method section, covering the backpropagation-based collection of the eight key GDS detection features, the processing of all sample features and the construction of the training dataset, as well as the MLP training and inference procedures used for member sample detection. We provide the full source code so that all experimental results reported in this work can be reproduced.

REFERENCES

Simone Balloccu, Patrícia Schmidtová, Mateusz Lango, and Ondřej Dušek. Leak, cheat, repeat: Data contamination and evaluation malpractices in closed-source llms. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 67–93, 2024.

Stella Biderman, Hailey Schoelkopf, Quentin Gregory Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, et al. Pythia: A suite for analyzing large language models across training and scaling. In *International Conference on Machine Learning*, pp. 2397–2430. PMLR, 2023.

Sid Black, Gao Leo, Phil Wang, Connor Leahy, and Stella Biderman. Gpt-neo: Large scale autoregressive language modeling with mesh-tensorflow. *Zenodo*, 2021.

Léon Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010: 19th International Conference on Computational StatisticsParis France, August 22-27, 2010 Keynote, Invited and Contributed Papers*, pp. 177–186. Springer, 2010.

Nicholas Carlini, Florian Tramer, Eric Wallace, Matthew Jagielski, Ariel Herbert-Voss, Katherine Lee, Adam Roberts, Tom Brown, Dawn Song, Ulfar Erlingsson, et al. Extracting training data from large language models. In *30th USENIX security symposium (USENIX Security 21)*, pp. 2633–2650, 2021.

Hyeong Kyu Choi, Maxim Khanov, Hongxin Wei, and Yixuan Li. How contaminated is your benchmark? measuring dataset leakage in large language models with kernel divergence. In *Forty-second International Conference on Machine Learning*.

Michael Duan, Anshuman Suri, Niloofar Mireshghallah, Sewon Min, Weijia Shi, Luke Zettlemoyer, Yulia Tsvetkov, Yejin Choi, David Evans, and Hannaneh Hajishirzi. Do membership inference attacks work on large language models? In *First Conference on Language Modeling*.

André Vicente Duarte, Xuandong Zhao, Arlindo L Oliveira, and Lei Li. De-cop: Detecting copyrighted content in language models training data. In *Forty-first International Conference on Machine Learning*.

Jonathan Frankle, David J Schwab, and Ari S Morcos. The early phase of neural network training. In *International Conference on Learning Representations*.

Michael M Grynbaum and Ryan Mac. The times sues openai and microsoft over ai use of copyrighted work. *The New York Times*, 27(1), 2023.

Guy Gur-Ari, Daniel A Roberts, and Ethan Dyer. Gradient descent happens in a tiny subspace. *arXiv preprint arXiv:1812.04754*, 2018.

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

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.

Yucheng Li. Estimating contamination via perplexity: Quantifying memorisation in language model evaluation. In *The Future of Machine Learning Data Practices and Repositories at ICLR 2025*.

Zhenhua Liu, Tong Zhu, Chuanyuan Tan, Bing Liu, Haonan Lu, and Wenliang Chen. Probing language models for pre-training data detection. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1576–1587, 2024.

Sourab Mangrulkar, Sylvain Gugger, Lysandre Debut, Younes Belkada, Sayak Paul, and Benjamin Bossan. Peft: State-of-the-art parameter-efficient fine-tuning methods. 2022.

Sebastian Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016.

Weijia Shi, Anirudh Ajith, Mengzhou Xia, Yangsibo Huang, Daogao Liu, Terra Blevins, Danqi Chen, and Luke Zettlemoyer. Detecting pretraining data from large language models. In *12th International Conference on Learning Representations, ICLR 2024*, 2024.

Hongyi Tang, Zhihao Zhu, and Yi Yang. Identifying pre-training data in llms: A neuron activation-based detection framework. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 18738–18751, 2025.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

Ben Wang and Aran Komatsuzaki. Gpt-j-6b: A 6 billion parameter autoregressive language model, 2021.

Hongyu Wang, Shuming Ma, Ruiping Wang, and Furu Wei. Q-sparse: All large language models can be fully sparsely-activated. *arXiv preprint arXiv:2407.10969*, 2024.

Hengxiang Zhang, Songxin Zhang, Bingyi Jing, and Hongxin Wei. Fine-tuning can help detect pretraining data from large language models. In *The Thirteenth International Conference on Learning Representations*.

Jingyang Zhang, Jingwei Sun, Eric Yeats, Yang Ouyang, Martin Kuo, Jianyi Zhang, Hao Frank Yang, and Hai Li. Min-k%++: Improved baseline for pre-training data detection from large language models. In *The Thirteenth International Conference on Learning Representations*, 2025.

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, et al. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*, 2022.

Weichao Zhang, Ruqing Zhang, Jiafeng Guo, Maarten Rijke, Yixing Fan, and Xueqi Cheng. Pretraining data detection for large language models: A divergence-based calibration method. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 5263–5274, 2024.

A Details of Baseline

Shared Notations: We first define the universal notations used across all baseline methods for consistency:

- x: Input text sample for membership inference detection.
- $\{x_1, x_2, \ldots, x_N\}$: Token sequence of input text sample x, where N denotes the total number of tokens in the text.
- $x_{<t}$: Prefix token sequence of the t-th token x_t, i.e., $x_{<t} = \{x_1, \ldots, x_{t-1}\}$.
- θ: Parameters of the target pre-trained language model.
- f_θ: Target pre-trained language model parameterized by θ.
- $p(\cdot \mid \cdot; \theta)$: Conditional probability distribution parameterized by the model parameters θ.
- $k\%$: Percentage threshold for selecting low-probability outlier tokens (used in MinK% and MinK%++).
- $y \in \{0, 1\}$: Ground-truth membership label, where 1 denotes a member sample and 0 denotes a non-member sample.
- $\hat{y} \in [0, 1]$: Predicted membership probability output by a supervised classifier.
- M: Number of training samples used by supervised baselines.

A.1 PPL

Perplexity (PPL) is a classic global likelihood-based metric that measures the model's uncertainty in predicting a given text, with lower PPL indicating higher text familiarity (more likely to be a member sample). First, the text is tokenized into a sequence of tokens $\{x_1, x_2, \ldots, x_N\}$. Then, the log-likelihood of each token is calculated conditional on the preceding tokens, and the final PPL is derived by normalizing the sum of log-likelihoods by the token sequence length and performing an exponential transformation.

$$\text{PPL} = \exp\left(-\frac{1}{N}\sum_{i=1}^{N}\log p(x_i|x_1, ..., x_{i-1}; \theta)\right) \tag{10}$$

A.2 MinK%

MinK% is a local likelihood-based method that mitigates global word frequency interference by focusing on low-probability outlier tokens. For an input text token sequence $x = \{x_1, x_2, \ldots, x_N\}$

(where N denotes the total number of tokens), it first calculates the conditional log-likelihood of each token x_i given its preceding context:

$$\log p(x_i \mid x_1, x_2, \ldots, x_{i-1}; \theta)$$

Subsequently, it sorts all token conditional probabilities in ascending order, selects the bottom $k\%$ tokens (i.e., the $k\%$ tokens with the smallest conditional probabilities) to form the low-probability set $T_{\text{min}(k\%)} = \text{MinK\%}(x)$. The final MinK% score is defined as the average conditional log-likelihood of tokens in $T_{\text{min}(k\%)}$:

$$\begin{aligned} &\text{MinK\% Score}(x) \\ &= \frac{1}{|T_{\text{min}(k\%)}|} \sum_{x_i \in T_{\text{min}(k\%)}} \log p(x_i \mid x_1, \ldots, x_{i-1}; \theta) \end{aligned} \tag{11}$$

where $|T_{\text{min}(k\%)}| = \lfloor k\% \times N \rfloor$ denotes the cardinality of the low-probability token set (rounded down to the nearest integer). During detection, a threshold is set on MinK% Score(x): text with a score below the threshold is classified as a member sample (since member samples tend to contain more low-probability outlier tokens), and vice versa.

A.3 MINK%++

MinK%++ is an optimized variant of MinK% that enhances detection robustness by integrating vocabulary-level probability statistics, addressing the text-length dependency and short-text fluctuation issues of the original method. For an input text token sequence $x = \{x_1, x_2, \ldots, x_N\}$, it first computes the normalized conditional log-likelihood for each token x_t (given prefix $x_{<t} = \{x_1, \ldots, x_{t-1}\}$) by normalizing with the vocabulary-wide log-probability statistics of the prefix $x_{<t}$.

Core Token-Level Score Calculation

For each token x_t, the token-level MinK%++ score is defined as:

$$\begin{aligned} &\text{MinK\%++}_{\text{token}}(x_{<t}, x_t) \\ &= \frac{\log p(x_t \mid x_{<t}; \theta) - \mu_{x_{<t}}}{\sigma_{x_{<t}}} \end{aligned} \tag{12}$$

where $\mu_{x_{<t}} = \mathbb{E}_{z \sim p(\cdot \mid x_{<t}; \theta)} [\log p(z \mid x_{<t}; \theta)]$: Expectation of the next-token log-likelihood over the model's vocabulary, given prefix $x_{<t}$; $\sigma_{x_{<t}} = \sqrt{\mathbb{E}_{z \sim p(\cdot \mid x_{<t}; \theta)} \left[\left(\log p(z \mid x_{<t}; \theta) - \mu_{x_{<t}} \right)^2 \right]}$: Standard deviation of the next-token log-likelihood over the vocabulary, given prefix $x_{<t}$.

Both $\mu_{x_{<t}}$ and $\sigma_{x_{<t}}$ can be computed analytically using the model's output logits (since $p(\cdot \mid x_{<t}; \theta)$ follows a categorical distribution), requiring no additional computational overhead beyond standard LLM inference.

Final Sequence-Level Score Calculation

Similar to MinK%, MinK%++ selects the bottom $k\%$ tokens with the smallest MinK%++$_{\text{token}}$ scores to form the low-probability set $T^{+}_{\text{min}(k\%)}$. The final sequence-level score is the average of the token-level scores in this set:

$$\begin{aligned} &\text{MinK\%++ Score}(x) \\ &= \frac{1}{|T^{+}_{\text{min}(k\%)}|} \sum_{(x_{<t}, x_t) \in T^{+}_{\text{min}(k\%)}} \text{MinK\%++}_{\text{token}}(x_{<t}, x_t) \end{aligned} \tag{13}$$

where $|T^{+}_{\text{min}(k\%)}| = \lfloor k\% \times N \rfloor$ denotes the cardinality of the low-probability token set. During detection, a threshold is applied: text with a lower MinK%++ score is more likely to be a member sample, as the normalized score better distinguishes low-probability outlier tokens of member samples from non-member ones.

A.4 NLL

NLL is a supervised feature-fusion baseline built on three unsupervised likelihood-based detectors: PPL, MinK%, and MinK%++. For an input sample x, it first computes the corresponding detection scores using the target model parameterized by θ, and concatenates them into a fixed-dimensional feature vector:

$$\mathbf{z}(x) = [\text{PPL}(x), \text{MinK\%}(x), \text{MinK\%++}(x)]\,. \tag{14}$$

Here, $\mathbf{z}(x)$ denotes the fused feature vector of sample x. The vector is then fed into a lightweight MLP classifier to predict the membership probability:

$$\hat{y} = \text{MLP}(\mathbf{z}(x)), \quad \hat{y} \in [0, 1], \tag{15}$$

where $\hat{y}$ is the predicted probability that x is a member sample. Let M denote the number of training samples, and let $y_i \in \{0, 1\}$ be the ground-truth membership label of the i-th training sample, with 1 indicating a member and 0 indicating a non-member. The MLP is trained with the standard binary cross-entropy loss:

$$\mathcal{L}_{\text{NLL}} = -\frac{1}{M}\sum_{i=1}^{M}\left[y_i \log \hat{y}_i + (1 - y_i)\log(1 - \hat{y}_i)\right], \tag{16}$$

where $\hat{y}_i$ is the predicted probability for the i-th sample. Thus, NLL serves as a supervised fusion baseline that combines global and local likelihood signals instead of relying on a single detection score.

A.5 NAPDD

NAPDD is a supervised detection framework based on neuron activation patterns. For an input sample x, it performs a forward pass through the target model parameterized by θ, and collects activations from attention and FFN sub-modules across different layers:

$$\mathbf{a}^{l,m}(x) = \text{Act}^{l,m}(x; \theta), \tag{17}$$

where $\mathbf{a}^{l,m}(x)$ denotes the activation of the m-th sub-module in the l-th layer, and $\text{Act}^{l,m}(\cdot; \theta)$ is the corresponding activation function. NAPDD then summarizes these activations into an activation-pattern feature vector:

$$\phi(x) = \mathcal{F}\left(\{\mathbf{a}^{l,m}(x)\}_{l,m}\right), \tag{18}$$

where $\mathcal{F}(\cdot)$ is a feature extraction function that captures statistics such as activation magnitude, sparsity, top-k neuron positions, and layer/module-wise activation distributions, and $\phi(x)$ is the resulting activation-pattern representation. Since member and non-member samples tend to trigger different neuron activation patterns, a supervised classifier $h(\phi(x)) \to \{0, 1\}$ is trained with binary membership labels to distinguish them. Here $h(\cdot)$ denotes the NAPDD classification head, and its output 1 indicates a member sample while 0 indicates a non-member.

A.6 KDS

KDS is a supervised detection baseline that measures distributional discrepancy with kernel divergence. For an input sample x, KDS compares the behavior of the target model parameterized by θ with that of a reference model parameterized by θ_{ref}, in terms of output distributions or hidden representations. Here θ_{ref} denotes the parameters of the reference model. It computes a kernel-based divergence score, e.g., a maximum mean discrepancy (MMD)-style score:

$$\text{KDS}(x) = D_\kappa\left(p_\theta(\cdot \mid x), p_{\theta_{\text{ref}}}(\cdot \mid x)\right), \tag{19}$$

where D_κ denotes a kernel-based divergence, $\kappa(\cdot, \cdot)$ is the kernel function, and $p_\theta(\cdot \mid x)$ and $p_{\theta_{\text{ref}}}(\cdot \mid x)$ are the output distributions of the target and reference models given x, respectively. A larger KDS value indicates that the target model's behavior on x deviates more from the reference model, which can serve as evidence of data leakage or membership. The KDS scores are then used as discriminative features to train a supervised classifier $h(\text{KDS}(x)) \to \{0, 1\}$ for final member/non-member prediction. Here $h(\cdot)$ denotes the KDS classification head, and its output 1 indicates a member sample while 0 indicates a non-member.

A.7 FSD

Fine-tuned Score Deviation (FSD) is a fine-tuning enhanced method that leverages score discrepancies between pre-trained and fine-tuned models for member sample detection. Its core workflow includes two key steps: first, constructing a domain-matched non-member fine-tuning dataset (using post-model-release unseen data), then performing self-supervised next-token prediction fine-tuning on the pre-trained model. Finally, it calculates the score difference between the pre-trained and fine-tuned models to distinguish member and non-member samples. The self-supervised fine-tuning loss is defined as:

$$\mathcal{L}_{\text{fine-tuning}}(x) = -\frac{1}{n}\sum_{i=1}^{n} \log f_\theta(x_i \mid x_1, \ldots, x_{i-1}) \tag{20}$$

The final FSD score is the deviation between the sample scores obtained from the pre-trained model f_θ and the fine-tuned model $f_{\theta'}$ (with parameters θ'):

$$\text{FSD}(x; f_\theta, f_{\theta'}) = S(x; f_\theta) - S(x; f_{\theta'}) \tag{21}$$

where $S(\cdot)$ represents a base scoring function (e.g., Perplexity, MinK%). A large FSD score indicates the sample is likely a non-member (fine-tuning reduces non-member perplexity significantly), while a small score suggests a member sample (fine-tuning has negligible impact on member perplexity). A validation-set optimized threshold ε is used for final classification.

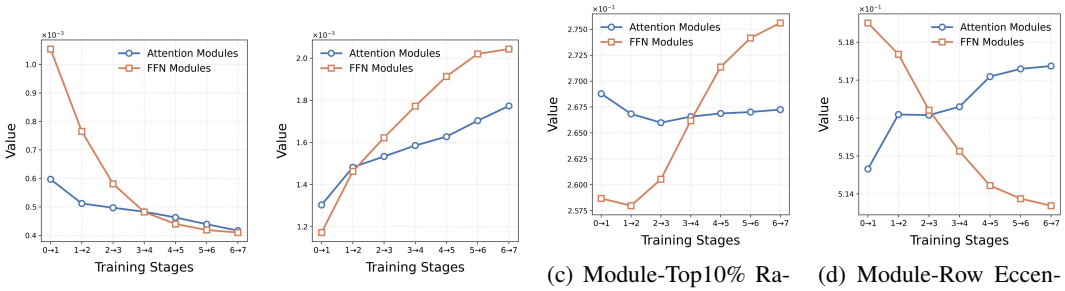
A.8 Experimental Settings

We use the PEFT library (Mangrulkar et al., 2022) for LoRA-based (Hu et al., 2022) gradient feature extraction without parameter updates. The base model is initialized from pre-trained checkpoints, and for LoRA configuration, we adopt default settings with rank $r = 16$, LoRA scaling factor $\alpha = 32$, and dropout set to 0 since we only focus on gradients from a single backpropagation step and no parameter updates are required. Target modules include all sub-modules of attention and FFN, including query, key, value, output, gate, up, and down projections. For different datasets, we select the tokenizer sequence length based on the overall length distribution: 256 for the WikiMIA dataset and 512 for all other datasets. When training the MLP classifier, the dataset is uniformly split into training and test sets at a ratio of 3:7. The MLP classifier is configured with two hidden layers of dimensions 128 and 64, respectively, and uses the default Adam optimizer with a learning rate of 0.001.

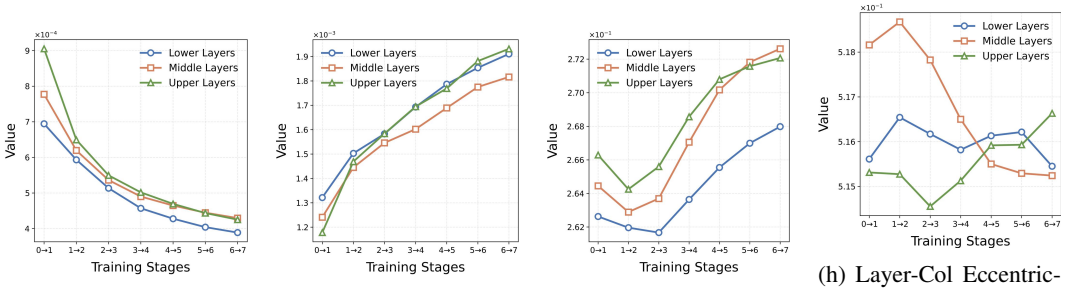
B Motivation Experiment Supplementary Details

To verify the parameter update law driven by the training process, we conduct LoRA fine-tuning with 7 epochs on LLaMA-7B using the unseen set of BookMIA dataset, counting LoRA matrix parameter updates per epoch to extract dynamic change curves of target features. Experiments are performed under three learning rates ($1e-5$, $3e-5$, $5e-5$). We present the variation trends of the four feature categories across different modules and layers under each learning rate.

We present the variation trends of the four feature categories across different modules and layers under each learning rate (Figs. 5–7). Across all learning rates, $\Delta\Theta_t$ and S_t exhibit clear epoch-wise patterns: $\Delta\Theta_t$ continuously decreases with training epochs, while S_t gradually increases. For E_t, distinct opposite trends are observed between the ATT and MLP modules—the update positions of the ATT module tend to be centralized, whereas those of the MLP module tend to be marginalized. Despite this divergence, E_t values of both modules converge to the same stable range across the three learning rates. In contrast, $T_{10,t}$ shows a different dynamic: under large learning rates, it first increases and then decreases, yet remains stably around 0.27 throughout training. Notably, the inflection point of $T_{10,t}$ decline coincides with the stage when E_t stabilizes. This aligns with our core hypothesis: during training, the model identifies core parameters correlated with the input data and performs concentrated updates on them; once this process is completed, the overall parameter update dynamics tend to stabilize. Regarding different hierarchical levels, only E_t shows noticeable discrepancies, while all other features follow the same evolutionary laws.

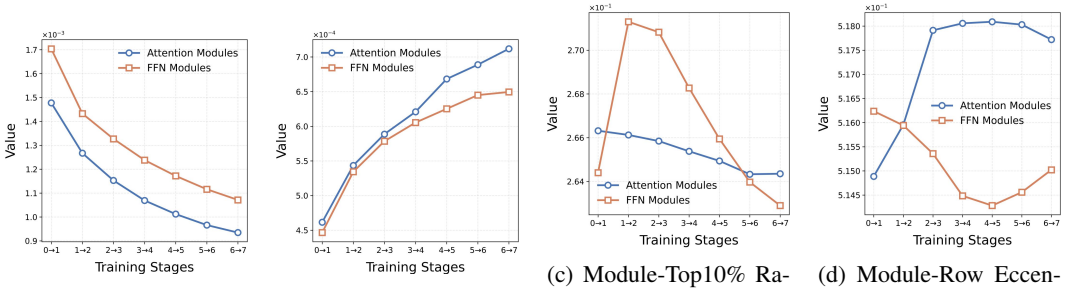


(a) Module-Mean Update (b) Module-Sparsity (c) Module-Top10% Ratio (d) Module-Row Eccentricity

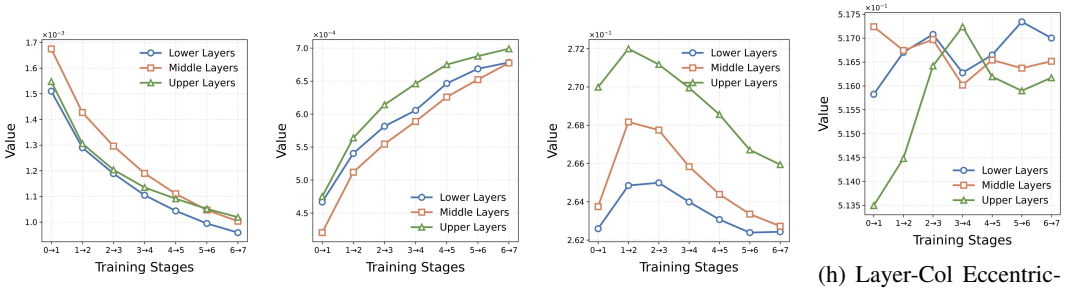


(e) Layer-Mean Update (f) Layer-Sparsity (g) Layer-Top10% Ratio (h) Layer-Col Eccentricity

Figure 5: Feature Dynamic Changes During LoRA Fine-Tuning (Learning Rate $1e-5$). Top row: Features in ATT/MLP modules; Bottom row: Features in low/middle/high layers.

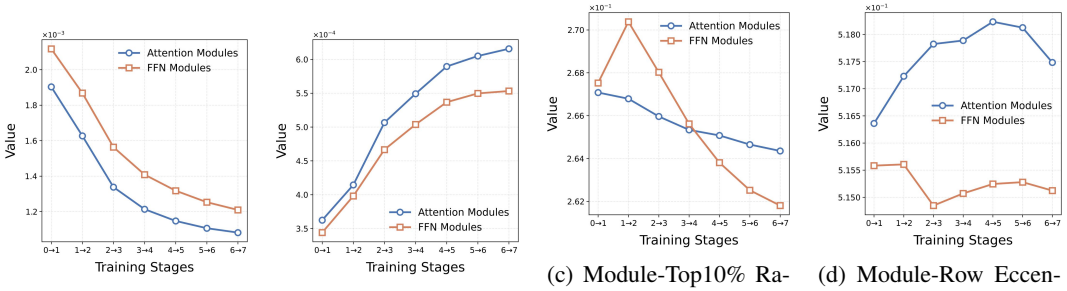


(a) Module-Mean Update (b) Module-Sparsity (c) Module-Top10% Ratio (d) Module-Row Eccentricity

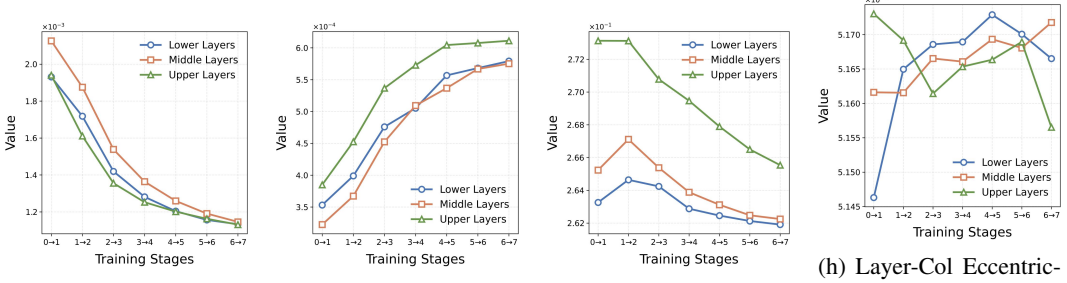


(e) Layer-Mean Update (f) Layer-Sparsity (g) Layer-Top10% Ratio (h) Layer-Col Eccentricity

Figure 6: Feature Dynamic Changes During LoRA Fine-Tuning (Learning Rate $3e-5$).



(a) Module-Mean Update (b) Module-Sparsity (c) Module-Top10% Ratio (d) Module-Row Eccentricity



(e) Layer-Mean Update (f) Layer-Sparsity (g) Layer-Top10% Ratio (h) Layer-Col Eccentricity

Figure 7: Feature Dynamic Changes During LoRA Fine-Tuning (Learning Rate $5e-5$).

C Ablation Experiments on Sub-features and Sub-modules

To further explore the discriminative power of fine-grained components in our method, we conduct two additional ablation experiments: sub-feature category ablation (8 sub-features) and sub-module ablation (7 sub-modules). All experiments follow the same configuration as the main text (LLaMA-7B model, WikiMIA/ArxivTection datasets), with AUROC and TPR@5% FPR as evaluation metrics.

C.1 Sub-feature Category Ablation Experiment

The three core feature categories are decomposed into 8 sub-features, as follows:

- Magnitude Distribution Features: Abs_Mean, Row_Mean_Max
- Core Concentration Features: 10p_Ratio, Sparsity, Std, Row_Mean_Std
- Position Offset Features: Row_Ecc, Col_Ecc

Ablation is performed by removing one sub-feature at a time (retaining the other 7), and the results are shown in Table 6.

Table 6: Ablation Experiment Results of Sub-features (AUROC / TPR@5% FPR Scores).

Dataset	Sub-feature (Ablation Variant)								Full
	-Abs_Mean	-Std	-10p_Ratio	-Sparsity	-Row_Ecc	-Col_Ecc	-Row_Max	-Row_Std	
WikiMIA	0.95/0.71	0.95/0.72	0.95/0.70	0.94/0.67	0.95/0.70	0.93/0.62	0.95/0.73	0.95/0.72	**0.96/0.84**
ArxivTection	0.96/0.82	0.96/0.81	0.96/0.83	0.96/0.82	0.96/0.83	0.95/0.78	0.96/0.83	0.96/0.83	**0.97/0.85**

To further examine the contribution of each feature category as a whole, we conduct an additional category-level ablation experiment. Specifically, we remove one entire feature category at a time while retaining the other two categories, i.e., only the magnitude distribution features, only the core concentration features, or only the position offset features are used. Results are reported in Table 7.

Table 7: Category-level Ablation Experiment Results (AUROC / TPR@5% FPR Scores).

Dataset	Feature Category (Ablation Variant)		
	Magnitude Only	Concentration Only	Position Only
WikiMIA	0.8956/0.4428	0.9215/0.6418	0.9370/0.6484
ArxivTection	0.9164/0.6367	0.9466/0.7640	0.9381/0.7322

The sub-feature ablation results verify that all sub-features contribute to detection performance, with their combination achieving optimal AUROC and TPR scores across both datasets. Among these sub-features, Col_Ecc (a Position Offset Feature) is the most critical: its removal leads to the sharpest performance decline (AUROC drops to 0.93 on WikiMIA and 0.95 on ArxivTection, with TPR falling to 0.62 and 0.78 respectively). In contrast, removing Abs_Mean, Std, 10p_Ratio, Row_Ecc, Row_Max, or Row_Std induces only mild performance degradation, with AUROC remaining above 0.95 on both datasets, indicating their relatively moderate importance; Sparsity also shows a slight impact on WikiMIA, with its ablation resulting in an AUROC of 0.94. Overall, the fine-grained sub-feature design is rational, as each component provides complementary discriminative information for pre-training data detection.

C.2 Sub-module Ablation Experiment

The ATT and FFN modules are decomposed into 7 sub-modules based on Transformer architecture details:

- ATT-related sub-modules: Q-Proj, K-Proj, V-Proj, Out-Proj
- FFN-related sub-modules: Gate-Proj, Up-Proj, Down-Proj

Ablation is performed by removing one sub-module's features at a time, and the results are shown in Table 8.

Table 8: Ablation Experiment Results of Sub-modules (AUROC / TPR@5% FPR Scores).

Dataset	Sub-module (Ablation Variant)							Full
	-Q-Proj	-K-Proj	-V-Proj	-Out-Proj	-Gate-Proj	-Up-Proj	-Down-Proj	
WikiMIA	0.95/0.72	0.95/0.71	0.94/0.69	0.94/0.67	0.95/0.69	0.94/0.65	0.94/0.61	**0.96/0.84**
ArxivTection	0.96/0.80	0.96/0.82	0.95/0.79	0.96/0.80	0.96/0.82	0.95/0.78	0.96/0.82	**0.97/0.85**

The sub-module ablation results demonstrate that all sub-modules provide complementary gradient information, as single-sub-module ablation consistently leads to performance degradation while integrating all sub-modules' features achieves the optimal AUROC and TPR@5% FPR scores across both datasets. Though the performance differences between individual ablation variants are mild, ATT-related sub-modules show slightly stronger discriminative power than FFN-related ones, with K-Proj performing prominently among ATT sub-modules; in contrast, among FFN sub-modules, Down-Proj and Gate-Proj contribute more to detection performance than Up-Proj, as retaining the former two yields slightly higher AUROC and TPR scores. The performance gap between single-sub-module variants and the full model further confirms the necessity of multi-sub-module feature fusion, which enables the capture of comprehensive gradient patterns across the Transformer architecture. It can be seen from this that our method also achieves excellent performance with smaller LoRA adaptation parameters and lower computational complexity.

D Additional Experiments Analysis

D.1 Complete results for the seven subsets of Mimir

Table 10 reports the complete AUROC results of all methods on the seven subsets of Mimir across five target models. On the *Average* column, our method achieves the highest score for every target

Table 9: Detection performance under low-data scenario using 10% training data(AUROC / TPR@5% FPR Scores).

Model	WikiMIA		ArxivTection	
	GDS	MinK++	GDS	MinK++
LLaMA	0.90/0.52	0.82/0.22	0.91/0.65	0.83/0.22
Pythia	0.81/0.33	0.70/0.18	0.88/0.56	0.70/0.17
OPT	0.86/0.37	0.65/0.11	0.88/0.47	0.65/0.10
GPT-J	0.84/0.37	0.69/0.19	0.91/0.60	0.69/0.19
GPT-Neo	0.81/0.35	0.67/0.15	0.92/0.67	0.67/0.15

model (0.73–0.76), outperforming the strongest supervised baseline NAPDD (0.71–0.72) and the strongest unsupervised baseline MinK. Compared with **NLL**—a likelihood feature trained under the *same* supervised protocol and labeled budget—our method is consistently higher across all five models, indicating that the gain stems from gradient-based statistics rather than from supervision alone.

At the subset level, our method attains the highest average AUROC on five of the seven subsets: GitHub, Pile CC, PubMed Central, ArXiv, and DM Mathematics. It is particularly effective on subsets with stronger membership signals. On **DM Mathematics**, our method reaches 0.94–0.95, clearly exceeding the best baseline; on **PubMed Central**, it achieves 0.79–0.85 and outperforms the best baseline on most models; on **Pile CC**, it obtains the highest score for all five models (0.57–0.59). On the more difficult subsets, our method leads on Pile CC but is surpassed by likelihood-based baselines on HackerNews. We also note that the unsupervised MinK remains higher on Wikipedia and HackerNews (e.g., Wikipedia 0.66 vs. our 0.63 on the 2.7B model), which we attribute to its token-level memorization score being more discriminative on repetitive, highly structured text, whereas our per-sample gradient statistics are coarser. Overall, across all subset-model combinations, our method achieves the best result in the majority of cells, confirming that the improvement is consistent rather than driven by any single subset.

D.2 Low-Data Scenarios

To thoroughly verify the data efficiency and robustness of our GDS method in low-data scenarios with limited labeled samples, we conduct extensive experiments where only 10% of the training data is used for model training. Specifically, we evaluate GDS across five different language models and two benchmark datasets, with MinK++ as the baseline for performance comparison. The experimental results, measured by two key metrics (AUROC score and TPR@5% FPR), are presented in Table 9, which clearly demonstrates that GDS still maintains superior detection performance compared to the baseline even under the strict constraint of limited training data.

D.3 Sensitivity Analysis

We conduct sensitivity experiments on LLaMA-7B (WikiMIA) by varying LoRA rank, initialization seed, and target module under main experiment settings. Table 11 shows our method maintains stable performance: AUROC remains high across ranks and seeds; it maintains excellent performance even when only a small number of target modules are used.

D.4 Efficiency Analysis

We compare inference time and peak memory of PPL, FSD, and GDS on the first WikiMIA sample with LLaMA-7B in bfloat16. PPL takes 70 ms, FSD 760 ms, and GDS 1001 ms. Peak memory is about 13 GB for PPL and GDS, versus 16 GB for FSD. Although GDS is about 30% slower than FSD due to gradient backpropagation, it performs no parameter updates, so it avoids optimizer states and matches the memory use of unsupervised methods while cutting memory by nearly 18.75% to FSD. Our GDS offers a favorable trade-off between detection effectiveness and computational overhead: it achieves superior detection performance without parameter updates and with modest

Table 10: Comparison (AUROC) on 7 Subsets of Mimir across 5 target models. Target models are denoted as 2.7B/6B/6.7B/6.9B/7B, corresponding to Neo-2.7B/GPT-J-6b/OPT-6.7b/Pythia-6.9b/LLaMA-7B respectively. Results are averaged over three experimental runs.

Method	**Average**					GitHub					Pile CC					PubMed Central				
	2.7B	6B	6.7B	6.9B	7B	2.7B	6B	6.7B	6.9B	7B	2.7B	6B	6.7B	6.9B	7B	2.7B	6B	6.7B	6.9B	7B
Unsupervised																				
PPL	0.64	0.66	0.62	0.73	0.59	0.78	0.79	0.78	0.90	0.83	0.56	0.57	0.56	0.56	0.52	0.78	0.79	0.78	0.78	0.72
MinK	0.71	0.73	0.69	0.75	0.60	0.88	0.89	0.76	0.90	0.83	0.56	0.57	0.56	0.56	0.52	0.81	0.80	0.74	**0.79**	0.72
MinK++	0.60	0.63	0.56	0.67	0.53	0.84	0.86	0.56	0.86	0.73	0.54	0.55	0.54	0.56	0.51	0.71	0.72	0.60	0.70	0.59
Supervised																				
NLL	0.72	0.74	0.70	0.74	0.70	0.89	0.90	0.80	0.90	0.85	0.53	0.54	0.53	0.54	0.51	0.80	0.81	0.74	0.80	0.73
KDS	0.64	0.66	0.65	0.65	0.64	0.74	0.76	0.70	0.77	0.70	0.54	0.53	0.52	0.52	0.55	0.67	0.66	0.71	0.68	0.67
NAPDD	0.72	0.72	0.72	0.72	0.71	0.89	0.88	0.89	0.91	0.90	0.55	0.54	0.53	0.51	0.54	0.74	0.79	0.76	0.77	0.75
FSD	0.61	0.66	0.59	0.62	0.55	0.77	0.80	0.62	0.77	0.72	0.55	0.55	0.54	0.55	0.52	0.71	0.79	0.63	0.56	0.63
Ours	**0.73**	**0.75**	**0.74**	**0.76**	**0.75**	0.90	**0.92**	**0.88**	**0.92**	**0.91**	**0.59**	**0.59**	**0.59**	**0.57**	**0.59**	**0.84**	**0.85**	**0.84**	**0.79**	**0.82**

Method	ArXiv					Wikipedia					DM Mathematics					HackerNews				
	2.7B	6B	6.7B	6.9B	7B	2.7B	6B	6.7B	6.9B	7B	2.7B	6B	6.7B	6.9B	7B	2.7B	6B	6.7B	6.9B	7B
Unsupervised																				
PPL	**0.78**	**0.79**	0.65	**0.78**	0.70	0.56	0.57	0.56	0.56	0.56	0.78	0.79	0.65	0.91	0.31	**0.65**	**0.62**	0.59	**0.62**	**0.59**
MinK	**0.78**	**0.79**	0.65	**0.78**	0.70	**0.66**	**0.68**	**0.65**	0.68	0.63	0.93	0.93	0.92	0.92	0.32	**0.65**	**0.62**	0.59	**0.62**	**0.59**
MinK++	0.60	0.64	0.53	0.70	0.57	0.65	**0.68**	0.60	**0.69**	0.56	0.77	0.79	0.67	0.75	0.22	0.53	0.57	0.51	0.60	0.53
Supervised																				
NLL	0.74	0.77	0.71	0.75	0.71	0.60	0.63	0.62	0.67	0.62	0.91	0.92	0.93	0.93	0.92	0.57	0.60	0.56	0.58	0.55
KDS	0.69	0.69	0.69	0.68	0.65	0.57	0.68	0.57	0.56	0.55	0.79	0.80	0.85	0.80	0.86	0.51	0.51	0.50	0.55	0.52
NAPDD	0.78	0.77	0.77	0.76	0.79	0.60	0.61	0.61	0.60	0.53	0.91	0.91	0.90	0.92	0.90	0.54	0.56	0.57	0.54	0.56
FSD	0.72	0.78	0.55	0.70	0.52	0.61	0.67	0.60	0.65	0.60	0.58	0.85	0.60	0.74	0.53	0.61	0.60	0.57	0.56	0.51
Ours	**0.78**	**0.79**	**0.78**	**0.78**	**0.77**	0.63	0.64	**0.65**	0.65	**0.64**	**0.95**	**0.95**	**0.94**	**0.95**	**0.95**	0.58	0.60	**0.60**	0.57	0.58

Table 11: Sensitivity analysis on WikiMIA (LLaMA-7B) using AUROC and TPR@5%FPR.

LoRA Rank r		
r	AUROC	TPR@5%FPR
16	0.96	0.84
8	0.96	0.82
4	0.95	0.77

Seed-LoRA		
Seed	AUROC	TPR@5%FPR
99	0.9658	0.8421
88	0.9674	0.8408
77	0.9636	0.8308
66	0.9623	0.8358

Seed-MLP		
Seed	AUROC	TPR@5%FPR
99	0.9658	0.8421
88	0.9576	0.8355
77	0.9672	0.8389
66	0.9631	0.8367

Target Module		
Module	AUROC	TPR@5%FPR
all	0.96	0.84
MLP	0.94	0.72
att	0.95	0.79

memory consumption, while incurring higher per-sample latency than likelihood-based methods because gradient extraction requires backpropagation.

D.5 Full Parameter Training

In LoRA training, only FFN and Attention modules are updated, while LayerNorm, despite containing meaningful training dynamics, is excluded. To account for this, we evaluate our method under full-parameter training on two small-scale datasets, WikiMIA and ArxivTection, for computational feasibility. As shown in Table 12, although full-parameter updates cause a slight performance drop

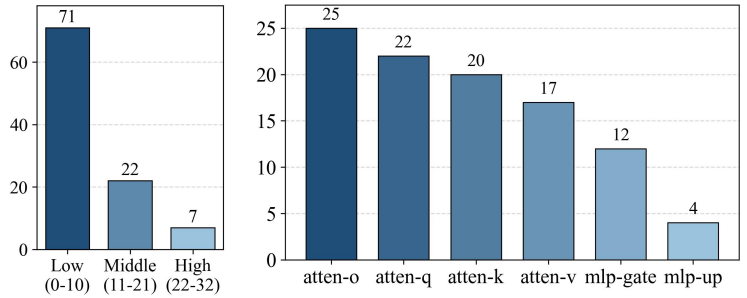
Table 12: Performance Comparison between LoRA and Full-Parameter Update using AUROC on WikiMIA and ArxivTection.

Dataset	Metric	LLaMA-7B		GPT-J-6B		OPT-6.7B		Pythia-6.9B		Neo-2.7B	
		LoRA	Full	LoRA	Full	LoRA	Full	LoRA	Full	LoRA	Full
WikiMIA	AUROC	**0.96**	**0.96**	**0.93**	0.90	**0.94**	0.92	**0.92**	0.89	**0.90**	0.88
	TPR@5% FPR	**0.84**	0.79	**0.66**	0.58	**0.67**	0.56	**0.63**	0.50	**0.60**	0.51
ArxivTection	AUROC	**0.97**	0.96	**0.97**	0.96	**0.94**	0.93	**0.95**	0.94	**0.94**	**0.94**
	TPR@5% FPR	**0.85**	0.84	**0.86**	0.79	**0.75**	0.67	**0.83**	0.73	0.73	**0.75**

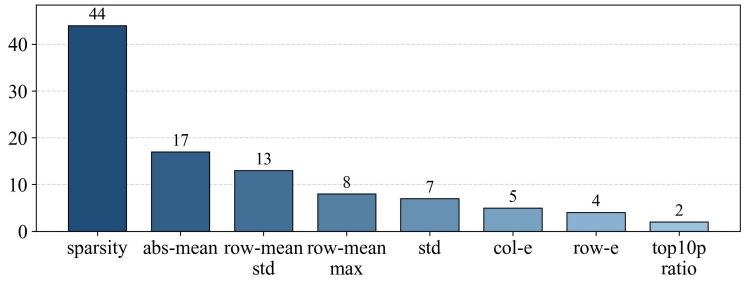
compared to LoRA, our method still outperforms most baselines. We attribute this gap to reduced gradient magnitudes in full training, which weaken discriminative signals between sample types.

D.6 FEATURE DISTRIBUTION

Figure 8 shows that sub-features with the largest distribution differences are concentrated in lower layers, while middle and higher layers contribute far less. Attention-related sub-modules dominate the discriminative features, with clear variation across modules. Among gradient features, grad_sparsity is by far the most discriminative.



(a) Layer-wise Difference *(b)* Sub-module Difference



(c) Sub-feature Difference

Figure 8: Distribution Difference Statistics of Discriminative Features. The y-axis shows the statistical count.