Neural Breadcrumbs: Membership Inference Attacks on LLMs Through Hidden State and Attention Pattern Analysis

Anonymous ACL submission

Abstract

Membership inference attacks (MIAs) reveal whether specific data was used to train machine learning models, serving as important tools for privacy auditing and compliance assessment. Recent studies have reported that MIAs perform only marginally better than random guessing against large language models, suggesting that modern pre-training approaches with massive datasets may be free from privacy leakage risks. Our work offers a complementary perspective to these findings by exploring how examining LLMs' internal representations, rather than just their outputs, may provide additional insights into potential membership inference signals. Our framework, *memTrace*, follows what we call "neural breadcrumbs" extracting informative signals from transformer hidden states and attention patterns as they process candidate sequences. By analyzing layer-wise representation dynamics, attention distribution characteristics, and cross-layer transition patterns, we detect potential memorization fingerprints that traditional loss-based approaches may not capture. This approach yields strong membership detection across several model families achieving average AUC scores of 0.85 on popular MIA benchmarks. Our findings suggest that internal model behaviors can reveal aspects of training data exposure even when output-based signals appear protected, highlighting the need for further research into membership privacy and the development of more robust privacy-preserving training techniques for large language models.

1 Introduction

Membership inference attacks (MIAs) occupy a paradoxical position in machine learning privacy. On one hand, they serve as crucial auditing tools that can determine whether copyright-protected or sensitive data was improperly used to train models offering accountability in an era of massive, often opaque data collection practices and usage. On the other hand, they pose a significant privacy threat when weaponized by adversaries seeking to uncover whether specific individuals' data was included in training sets, potentially revealing sensitive associations or characteristics about those individuals. This dual nature becomes especially concerning for large language models due to their ubiquitous deployment across critical domains and unprecedented scale that even allows them to act on behalf of the user. Advancing MIA techniques, thus, is crucial for both these reasons. High performing MIA solutions can not only help audit improper usage of data for training, but can also help in security research by precisely identifying vulnerabilities that drive the development of more robust defenses.

Despite their importance, recent research has raised significant questions about whether effective membership inference is even possible for large language models. A growing body of evidence suggests surprising limitations in current approaches. Duan et al. (2024) found that across various model sizes and domains, membership inference attempts barely outperform random guessing. Das et al. (2025) demonstrated that MIAs attempting to identify pre-training data in LLMs perform only marginally better than blind baselines, while Maini et al. (2024) argued that many purportedly successful attacks may actually be detecting temporal distribution shifts rather than true data membership. This apparent resistance to membership inference has been attributed to several factors unique to modern LLMs such as their massive training datasets combined with relatively few training iterations, and the inherently blurred boundaries between member (texts that were seen by the model during training) and non-member data (texts that were not seen by the model during training) points due to substantial n-gram overlap in natural language corpora. In contrast, other work has shown that these models may be significantly more vulnerable to membership inference attacks after fine-tuning on smaller, specialized datasets (Zhang et al., 2025a; Fu et al., 2025).

We propose an alternative perspective : the limitations of existing MIA approaches may stem not from the absence of membership signals but from their primary focus on model outputs which examine only the final layer logits or next-token probabilities. This conventional approach effectively reduces the entire computational process of a massive language model to a single terminal point, potentially overlooking rich information encoded within the model's internal dynamics. The processing of a sequence through many transformer layers leaves what we call "neural breadcrumbs" which are traces of how the model processes information differ-

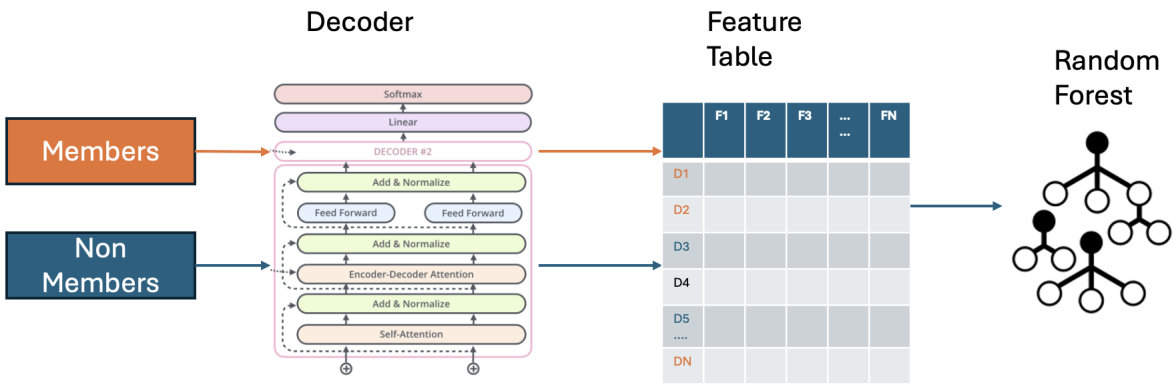


Figure 1: We leverage decoder based LLMs to generate several features for members and non members. These features are then used to train a Random Forest classifier. This classifier acts as an interpretable system to identify why a sample at inference is called as a member or a non member.

ently when encountering member versus non-member sequences. These processing patterns, while potentially obscured in the final output distribution, may be detectable in the model's intermediate computations and attention mechanisms.

Our approach, *memTrace*, leverages this insight by extracting features from hidden state representations and attention patterns across the full transformer architecture. Unlike previous methods, *memTrace* does not require fine-tuning the LLM on additional (often specialized) training data, instead directly analyzing how representations evolve through the network when processing candidate sequences. By examining layer-wise representation dynamics, attention distribution characteristics, and cross-layer transition patterns, we can train lightweight classifiers like single-layer MLPs, random forests, etc. to distinguish between member and non-member sequences. This approach is summarized in Figure 1. Our comprehensive evaluation across multiple model architectures and diverse text domains reveals that this approach achieves substantially stronger membership detection capabilities than output-based methods, with average AUC scores of 0.85 on popular MIA benchmarks. These findings suggest that while LLMs may appear resistant to traditional membership inference techniques, their internal processing mechanisms can still reveal subtle but detectable signs of training data exposure.

Our **key contributions** could be summarized as follows:

1. We introduce *memTrace*, a membership inference approach that extracts multi-faceted features from transformer hidden states and attention patterns. Unlike previous methods, our approach analyzes model's internal representation dynamics to reveal membership signals without heavy-weight fine-tuning of the LLM.

2. We present empirical evidence contradicting the prevailing hypothesis that modern LLM training paradigms inherently limit membership inference risk, demonstrating successful attacks across multiple model architectures and diverse text domains.

3. We provide interpretable insights into how LLMs process familiar content differently from unfamiliar data. Our analysis reveals that membership often manifests through distinctive processing pathways for members versus non-members. These findings contribute to a mechanistic understanding of how memorization affects model processing across the transformer stack.

Our findings fundamentally re-frame the privacy risk landscape for large language models by identifying that analyzing the internal representations of large language models may provide additional insights into their memorization behaviors. As LLMs continue their deployment across sensitive domains from healthcare to legal applications, our work indicates that privacy risks warrant continued research attention, particularly in developing privacy-preserving training techniques.

2 Methodology

Specifically, a *membership inference attack* (MIA) attempts to decide, for a target model f and a candidate example x, whether x was included in the model's training set $\mathcal{D}_{\text{train}}$. The adversary outputs a membership guess $\hat{b} \in \{0, 1\}$, aiming to maximize the *attack advantage*:

$$\text{Adv} = \left|\Pr[\hat{b} = 1 \mid x \in \mathcal{D}_{\text{train}}] - \Pr[\hat{b} = 1 \mid x \notin \mathcal{D}_{\text{train}}]\right|.$$

In this section, we introduce *memTrace*, a membership inference framework that analyzes transformer hidden states and attention patterns to identify distinctive processing signatures associated with training data exposure. Unlike output-focused approaches, our method captures the subtle "neural breadcrumbs" left throughout the model's internal layers as it processes familiar versus unfamiliar content.

Feature Vector Construction We create a feature vector that captures behavioral signatures across the model's hidden layers. For each input sequence x_m and a transformer model M, we construct a feature vector $\mathbf{f}_m$ that characterizes the model's processing behavior. For each sequence, we collect hidden states $H^{(l)} \in \mathbb{R}^{n \times d}$ from each layer l, extract attention weights $A^{(l)} \in \mathbb{R}^{h \times n \times n}$ from each layer l, and obtain logits $L^{(l)} \in \mathbb{R}^{n \times |V|}$ from prediction layers for a sequence of size n with hidden layer size as d and vocabulary V. We then compute layer-specific features from these internal representations, analyzing how they transform and evolve. Finally, these features are aggregated into a single fixed-length vector characterizing the model's processing behavior. This pipeline generates a comprehensive feature vector that serves as a fingerprint of how the model processes each input sequence. For a model with L layers, the feature vector includes statistics computed across all layers, capturing both layer-specific behaviors and cross-layer patterns.

Layer Transition Features The progression of representations through a transformer's layers reveals how information is incrementally transformed and refined. We quantify these dynamics through representation surprise metrics that measure the distance between hidden states of consecutive layers for each token position. For each token position t and layer transition from i to $i+1$, we compute the transition surprise as $\text{transition_surprise}_{i \to i+1}[t] = \left\| \mathbf{h}_t^{(i+1)} - \mathbf{h}_t^{(i)} \right\|_2$, which measures the Euclidean distance between consecutive layer representations. We also calculate normalized version of surprise to capture directional changes independent of magnitude. Additionally, we measure representation stability using cosine similarity, $\text{stability}_{i \to i+1}[t] = \cos(\mathbf{h}_t^{(i+1)}, \mathbf{h}_t^{(i)})$. For each layer transition, we compute statistical aggregates including mean, minimum, maximum, standard deviation, argmin, and argmax across token positions. These statistics characterize how information transforms between successive layers, potentially revealing different processing pathways for memorized versus novel content.

Prediction Confidence and Entropy Features For each layer that produces logits, we analyze the model's predictive behavior through entropy and confidence metrics. The entropy at each token position is calculated as $\text{entropy}[t] = -\sum_{v \in V} p_t(v) \log p_t(v)$, where $p_t(v)$ represents the softmax probability for vocabulary item v. Prediction confidence is measured as $\text{confidence}[t] = \max_{v \in V} p_t(v)$, representing the model's certainty in its top prediction. We also compute the confidence gap as $\text{confidence_gap}[t] = p_t(v_1) - p_t(v_2)$, where v_1 and v_2 are the top two predicted tokens. This gap indicates how decisively the model favors its primary prediction over alternatives. For each layer, we compute statistical aggregates of these metrics across token positions, yielding features that characterize the model's certainty profile. We further analyze confidence stability as $\text{confidence_stability} = \frac{\mu(\text{confidence})}{\sigma(\text{confidence})}$, which captures how consistently confident the model is across token positions. Higher values indicate more uniform confidence, while lower values suggest the model has varying degrees of certainty when processing different parts of the sequence, a pattern we hypothesize may correlate with partial memorization.

Attention Pattern Analysis For each layer l and attention head h, we extract attention weights $A^{l,h} \in \mathbb{R}^{n \times n}$ for sequence length n and analyze them across multiple dimensions. We first examine general attention distribution features by averaging across heads to obtain $\bar{A}^l$ and calculating metrics such as attention entropy: $\text{attention_entropy} = \frac{1}{n} \sum_{t=1}^{n} \left(-\sum_{s=1}^{n} \bar{a}_{t,s}^l \log_2(\bar{a}_{t,s}^l + \epsilon) \right)$. We also compute attention concentration as $\text{attention_concentration} = \frac{1}{n} \sum_{t=1}^{n} \max_s \bar{a}_{t,s}^l$, which measures how decisively the model focuses on specific tokens. Additionally, we quantify attention sparsity as $\text{attention_sparsity} = \text{mean}(\bar{A}^l < \tau)$, where τ is adaptively determined based on the layer's attention statistics. Lower entropy indicates more focused attention, while higher sparsity suggests more selective information routing. For head-specific analysis, we calculate metrics for each individual attention head, including head entropy $\text{head_entropy}[h]$ and head focus $\text{head_focus}[h] = \frac{1}{n} \sum_{t=1}^{n} \max_s a_{t,s}^{l,h}$.

Position-based attention features analyze the spatial patterns in attention allocation. We measure self-attention quantifying how much tokens attend to themselves. Previous token bias is calculated as $\text{prev_token_bias} = \frac{1}{n-1} \sum_{t=2}^{n} \bar{a}_{t,t-1}^l$, capturing the model's tendency to focus on immediately preceding context. We also compute the mean attention distance as $\text{mean_attention_distance} = \frac{\sum_{t,s} |t-s| \cdot \bar{a}_{t,s}^l}{\sum_{t,s} \bar{a}_{t,s}^l}$, which characterizes whether the model tends to attend locally or globally. These features characterize the model's attention to different positional relationships, revealing how it integrates information across sequence positions. Collectively, these attention-based features provide a detailed characterization of how models allocate computational focus, revealing subtle but consistent differences between the processing of memorized versus novel content. Our implementation adaptively adjusts feature extraction based on the specific model architecture being analyzed, allowing for consistent comparison across model families.

Context Evolution Features We analyze how context representations evolve as more tokens are processed by computing $\text{context_evolution}[i] = \left\| \frac{1}{i+1} \sum_{t=0}^{i+1} \mathbf{h}_t - \frac{1}{i} \sum_{t=0}^{i} \mathbf{h}_t \right\|_2$. This measures how much the average representation changes when adding each new token to the context. The statistical properties of these evolution steps reveal how efficiently the model integrates new information, with memorized content potentially showing different integration patterns compared to novel content.

Token-Position Specific Features For specific token

positions (e.g., first, middle, last), we compute specialized features that capture position-dependent processing. We measure first-to-last token similarity as first_last_similarity $= \cos(\mathbf{h}_0, \mathbf{h}_{n-1})$, capturing the relationship between the beginning and end of the sequence. For specific token positions k, we compute local statistics such as token_k_confidence_std $= \sigma(\{\text{confidence}[t]\}_{t=k-c}^{k+c})$, which measures confidence variation in the local neighborhood of position k. These position-specific metrics capture how differently the model processes tokens based on their sequential position, potentially revealing position-dependent memorization patterns.

The complete feature vector combines all metrics described above across all layers, resulting in a comprehensive representation of the model's internal processing behavior. All features are normalized using z-score standardization to ensure comparable scales across different feature types.

Membership Inference Classifier Using the extracted features, we train a random forest classifier to predict whether an input was part of the model's training data. We implement a robust cross-validation framework to ensure reliable performance. First, we employ 5-fold stratified cross-validation with an inner hyperparameter optimization loop using RandomizedSearchCV. We explore parameters including the number of estimators (100-400), maximum tree depth (3-10), minimum samples for splits and leaf nodes, and feature selection strategies. All numerical features are standardized using StandardScaler, fitted only on training data. The final classifier is trained on 80% of the data with the most commonly selected hyperparameters across folds, and evaluated on a held-out 20% test set. We report performance using AUC as our primary metric due to its robustness to class imbalance following previous literature reported as baselines. Feature importance analysis provides insights into which behavioral patterns most strongly indicate memorization in large language models.

3 Experiments

In this section we describe the experiment setup and present our key results.

3.1 Experiment Setup

Datasets We evaluate our approach on established membership inference benchmarks spanning diverse domains. MIMIR Benchmark (Duan et al., 2024) provides carefully curated member/non-member pairs across multiple domains, controlling for confounding factors like temporal effects and distribution shifts. We use its Wikipedia, GitHub, PubMed Central, HackerNews, and DM Mathematics subsets obtained from The Pile, which represent different text domains with varying linguistic characteristics and structural properties. The MIMIR benchmark contains multiple variants of each dataset, each created using different filtering criteria. We specifically focus on the 13-gram 0.8 split version, which refers to the set where the 13-gram overlap between member and nonmember texts is unrestricted and could be anywhere between 0-80%, making it significantly more difficult for models to distinguish between the two classes. WikiMIA and BookMIA (Shi et al., 2024) contain Wikipedia articles and book excerpts respectively, with membership labels based on the presence of content in known training corpora. These datasets provide longer text sequences that better reflect real-world use cases.

Models We evaluate our approach on three families of language models that span diverse architectures, scales, and training methodologies: i) Pythia (Biderman et al., 2023) : A suite of models ranging from 70M to 6.9B parameters, all trained on The Pile dataset; ii) LLaMA v1 (Touvron et al., 2023) Meta's 7B-parameter foundational model trained on trillions of tokens from diverse publicly available corpora. These models employ different architectural choices from Pythia, including activations and rotary positional embeddings, providing a test of our method's generalization across architectural variations; iii) GPT-Neo (Black et al., 2021) EleutherAI's 1.3B and 2.7B parameter transformers trained on The Pile using a Mesh-Tensorflow implementation, offering another test of cross-architecture generalization.

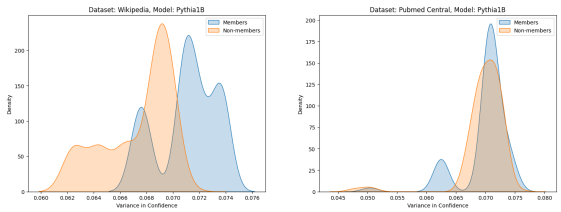
Baselines We compare our approach against established membership inference methods and the current state-of-the-art in membership inference approaches, deliberately two distinct categories of approaches: those requiring model training or fine-tuning, as well as training-free methods. Specifically, we test Perplexity (Yeom et al., 2018) that uses raw sequence likelihood, with lower perplexity suggesting training exposure; Min-K% (Shi et al., 2024); Min-K++% (Zhang et al., 2025b) that focus on the least confident token predictions to avoid confounding effects from common patterns; Lowercase (Carlini et al., 2022) that computes the ratio of perplexities between original and lowercased text; Neighborhood (Mattern et al., 2023) that examines probability curvature around examples through controlled perturbations; Zlib (Carlini et al., 2022) that uses the ratio between perplexity and compression entropy as a membership signal; ReCaLL (Xie et al., 2024) that tries to modify the predictive distribution of the LLM by prefixing the test instance with some additional text from the non-members; and techniques requiring fine-tuning such as MIATuner (Fu et al., 2025) that instruction-tunes models to directly identify their training data; Fine-tuned Score Deviation (Shi et al., 2024) that leverages fine-tuning on auxiliary data to amplify the gap between member and non-member scores. Since Fine-tuned score deviation method can work on top of other attacks, we compute its performance for every attack (refer to appendix) but report the numbers corresponding to the perplexity attack since they were the highest. For reproducing baselines that require tuning,

we use the best hyper-parameters provided in their respective papers. For all reference based attacks we use stablelm-base-alpha-3b-v2 as the reference model as suggested by Duan et al. (2024).

Setup Our experiments utilized models downloaded from the EleutherAI (for Pythia models and GPT-Neo models) and huggyllama (for LLaMA) repositories on Hugging Face. Each model was evaluated using its native tokenizer from the same repository to ensure consistency with training conditions. All experiments were conducted on an instance equipped with 8 NVIDIA A100 GPUs (40GB memory each), 96 vCPUs, and 1,152 GB of RAM. We processed samples with a batch size of 32 and padding and truncation set to True for feature extraction across all model sizes, allowing us to efficiently compute the full feature vectors within GPU memory constraints. The inference pipeline was implemented using PyTorch 1.13 with CUDA 11.6, and we utilized the Hugging Face Transformers library for model loading and tokenization. For all experiments including baselines, we employed 5-fold stratified cross-validation (random seed - 420) with to ensure robust evaluation, and all reported performance metrics represent averages across these 5 folds. We conducted variance analysis across all cross-validation folds and found consistent performance with standard deviations typically below 0.03 AUC, indicating stable results. The variance is provided in Appendix due to space constraints.

3.2 Results

The main results comparing AUC of our method against the baselines across different datasets and models are included in Table 1. For challenging membership inference scenarios, particularly in domains such as Wikipedia, Pubmed Central, Hacker News, and Github, our method demonstrates substantial improvements in AUC compared to all baseline approaches. On the WikiMIA dataset, while the fine-tuning based baseline methods achieve strong performance due to the dataset's distinct distributional properties, our method maintains competitive performance, matching the high AUC scores of existing approaches. For the BookMIA dataset, which contains longer sequences of 512 words, our method achieves near-perfect AUC scores. The extended sequence length provides more opportunities for detecting distinctive processing patterns, making membership inference more reliable. The performance numbers of our method on the Github dataset being slightly lower compared to that of other datasets is due to the large percentage of texts in members and non-members with 13-gram overlap. This issue is discussed in detail below. Overall, the consistency across architectures and scales of our method is particularly significant as it suggests that vulnerability to membership inference is an inherent characteristic of language models, regardless of their size or architectural design.



(a) Domain: Wikipedia (b) Domain: Pubmed Central

Figure 2: Feature distribution for the variance in confidence feature for both members and non-members samples across two different data domains from the MIMIR benchmark.

Feature Analysis Our feature analysis reveals distinct processing signatures in large language models when handling previously seen versus novel texts, even though the model may not have memorized the content. The feature importance analysis included in Table 2 shows that Attention Analysis features rank highest for GitHub data, followed by Layer Transitions and Prediction Confidence, while for all other datasets, Prediction Confidence ranks first, followed by Attention Analysis and Layer Transitions, though the same top feature classes appear across both dataset types. Among the prediction confidence features, we observe that when measuring the entropy of next token prediction probability distributions across each layer, previously seen content exhibits significantly higher variance in entropy compared to unseen content. This means the model's confidence fluctuates more dramatically across token positions for familiar text. This pattern suggests LLMs possibly develop recognition hot-spots, and rather than maintaining uniform confidence throughout a sequence, the model exhibits extremely high confidence at specific anchor positions where pattern recognition occurs, creating a distinctive oscillating confidence profile. Figure 2 illustrates the distribution of this feature across two datasets. Additionally, we observe that certain attention heads display characteristic entropy and focus patterns when processing familiar versus unfamiliar content. The head entropy metric quantifies how uniformly attention distributes across token positions, while head focus measures how strongly an attention head concentrates on its maximum attention targets. Select heads consistently demonstrate lower entropy and higher focus for familiar content, revealing specialized mechanisms for detecting specific linguistic or statistical patterns in previously encountered text to emphasize the familiar patterns. We observe that the rules in the random forest classifier incorporate features spanning multiple network layers, rather than relying on any single layer, even though the contribution of each layer is varied.

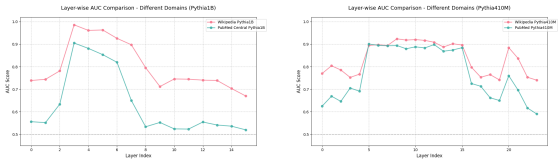
We also compare the AUC obtained by utilizing the features from individual layers only, and show these results in Figure 3. For all the models and datasets, we observe that AUC is the highest at middle layers,

Method	Wikipedia							Pubmed Central							Hacker News						
	Pythia-70M	Pythia-410M	Pythia-1B	Pythia-6.9B	LLaMA-7B	GPTNeo-1.3B	GPTNeo-2.7B	Pythia-70M	Pythia-410M	Pythia-1B	Pythia-6.9B	LLaMA-7B	GPTNeo-1.3B	GPTNeo-2.7B	Pythia-70M	Pythia-410M	Pythia-1B	Pythia-6.9B	LLaMA-7B	GPTNeo-1.3B	GPTNeo-2.7B
Perplexity	0.51	0.50	0.50	0.52	0.54	0.51	0.51	0.50	0.49	0.50	0.52	0.51	0.49	0.50	0.49	0.49	0.50	0.52	0.51	0.51	0.51
Min-K%	0.47	0.48	0.51	0.51	0.49	0.52	0.52	0.51	0.50	0.51	0.53	0.53	0.50	0.51	0.51	0.50	0.51	0.52	0.52	0.51	0.52
Min-K++%	0.50	0.50	0.52	0.56	0.51	0.52	0.53	0.50	0.49	0.51	0.53	0.50	0.51	0.51	0.50	0.50	0.52	0.54	0.49	0.51	0.52
Lowercase	0.51	0.47	0.49	0.52	0.48	0.51	0.50	0.51	0.51	0.50	0.52	0.53	0.52	0.52	0.50	0.50	0.51	0.51	0.52	0.52	0.52
Zlib	0.52	0.51	0.52	0.53	0.54	0.52	0.52	0.51	0.50	0.50	0.52	0.52	0.51	0.51	0.50	0.50	0.50	0.53	0.53	0.50	0.51
Neighborhood	0.53	0.52	0.52	0.53	0.54	0.53	0.53	0.48	0.48	0.49	0.53	0.54	0.51	0.51	0.50	0.51	0.50	0.52	0.52	0.50	0.51
ReCaLL	0.51	0.50	0.52	0.54	0.50	0.52	0.52	0.52	0.50	0.50	0.51	0.50	0.50	0.51	0.53	0.53	0.53	0.54	0.50	0.53	0.54
MIATuner	0.49	0.51	0.51	0.49	0.50	0.50	0.49	0.50	0.50	0.48	0.50	0.50	0.50	0.52	0.50	0.50	0.53	0.50	0.50	0.49	0.49
FSD	0.48	0.47	0.49	0.47	0.50	0.50	0.49	0.49	0.49	0.49	0.47	0.48	0.48	0.49	0.49	0.49	0.49	0.47	0.50	0.49	0.47
memTrace (Ours)	**0.65**	**0.87**	**0.89**	**0.89**	**0.90**	**0.86**	**0.86**	**0.59**	**0.89**	**0.88**	**0.84**	**0.80**	**0.66**	**0.72**	**0.82**	**0.81**	**0.82**	**0.83**	**0.85**	**0.71**	**0.78**
	WikiMIA							**BookMIA**							**Github**						
Perplexity	0.58	0.60	0.60	0.65	0.64	0.61	0.63	0.57	0.59	0.59	0.61	0.58	0.59	0.61	0.61	0.61	0.63	0.65	0.67	0.68	0.70
Min-K%	0.57	0.59	0.60	0.67	0.63	0.62	0.61	0.59	0.61	0.62	0.64	0.62	0.62	0.64	0.62	0.62	0.64	0.69	0.67	0.66	0.69
Min-K++%	0.50	0.62	0.64	0.72	0.85	0.65	0.68	0.50	0.40	0.45	0.55	0.57	0.40	0.42	0.50	0.67	0.69	0.73	0.61	0.68	0.70
Lowercase	0.54	0.57	0.57	0.59	0.58	0.59	0.60	0.59	0.59	0.60	0.65	0.64	0.60	0.59	0.59	0.58	0.60	0.59	0.58	0.59	0.60
Zlib	0.59	0.60	0.59	0.62	0.60	0.62	0.61	0.60	0.61	0.61	0.62	0.63	0.62	0.61	0.63	0.66	0.67	0.70	0.71	0.70	0.70
Neighborhood	0.53	0.55	0.54	0.66	0.64	0.63	0.65	0.60	0.62	0.60	0.68	0.65	0.65	0.68	0.64	0.64	0.65	0.69	0.68	0.66	0.67
ReCaLL	0.76	0.87	0.86	0.89	0.67	0.82	0.84	0.70	0.85	0.84	0.87	0.69	0.81	0.83	0.62	0.67	0.69	0.73	0.63	0.70	0.72
MIATuner	0.50	**0.96**	**0.98**	**0.95**	0.93	0.89	**0.96**	0.35	0.96	0.98	0.97	0.98	0.69	0.92	0.59	0.66	0.70	0.72	0.70	0.68	0.71
FSD	0.81	0.89	0.92	0.91	0.90	**0.92**	0.92	**0.95**	**0.98**	**0.98**	**0.98**	**0.99**	**0.98**	0.98	0.45	0.46	0.42	0.37	0.49	0.49	0.47
memTrace (Ours)	**0.82**	0.87	0.80	0.87	**0.94**	0.79	0.85	0.91	0.94	0.95	0.95	0.98	**0.98**	**0.99**	**0.67**	**0.70**	**0.73**	**0.77**	**0.78**	**0.72**	**0.72**

Table 1: Comparison of test AUC scores across various methods, datasets and models.

GitHub Dataset	Other Datasets
1. *Attention Analysis*	1. *Prediction Confidence*
2. *Layer Transitions*	2. *Attention Analysis*
3. *Prediction Confidence*	3. *Layer Transitions*
4. *Token Positional*	4. *Token Positional*
5. *Context Evolution*	5. *Context Evolution*
6. *Activation Patterns*	5. *Activation Patterns*

Table 2: Comparative ranking of feature classes by importance. The important classes of features differ slightly for the Github dataset as shown in the table.



(a) Pythia1B (b) Pythia410M

Figure 3: Layer-wise AUC comparison on Pythia 1B and Pythia 410M models.

suggesting that these layers serve as critical integration points where signatures become most detectable. It possibly suggests that in these middle layers the model has processed enough contextual information to activate specialized pathways for familiar content, but hasn't yet completed the transition to generalized output generation that occurs in later layers, indicating that memorization detection is not simply about initial feature recognition or final output prediction, but rather about identifying the distinctive processing trajectories that memorized content follows through the transformer's architecture. The lower performance at the final layer aligns with previous findings in the literature that focus on analyzing only final layer outputs and do not find differentiation between the members and the non-members.

MIA on Semantic Neighbors The MIMIR benchmark provides a comprehensive dataset that includes not only members and non-members but also their semantic neighbors. These neighbors are generated through a controlled perturbation process using the BERT model with approximately 15% masking rate. This process creates variations of the original texts while preserving their semantic meaning, for instance, replacing certain words with synonyms or restructuring sentences while maintaining the core message. The inclusion of these neighbors allows us to evaluate how membership inference attacks perform on texts that are semantically similar but not identical to the training data. We utilize our classifier originally trained to distinguish between members and non-members for testing on these neighbor samples in zero-shot way to assess whether the model's ability to detect membership extends to semantically similar content or is limited to exact matches, and report results in Table 3. Based on this analysis, we observe that while the precision in identifying semantically similar members is high, the recall is still low which implies that neighbors while containing semanti-

cally similar information as members are not captured as true members by the classifier.

Dataset	Model	Precision	Recall
Wikipedia	Pythia70M	90.1	26.4
	Pythia1B	91	14.6
Pubmed Central	Pythia70M	75.2	27
	Pythia1B	85	12.9

Table 3: Membership inference performance on semantic neighbors.

Model	DM Mathematics		Github	
	< 80%	< 20%	< 80%	< 20%
Pythia70M	0.63	**0.72**	0.67	**0.81**
Pythia410M	0.70	**0.73**	0..70	**0.83**
Pythia1B	0.69	**0.72**	0.73	**0.84**
Pythia6.9B	0.74	**0.77**	0.77	**0.85**

Table 4: Membership inference performance on high-overlap domains (Github and DM Mathematics) under different n-gram overlap constraints. Results compare restricted (<20% 13-gram overlap) versus unrestricted (<80% overlap) settings, demonstrating the impact of textual similarity on detection accuracy.

Effect of n-gram Overlap Different data domains naturally exhibit varying levels of n-gram overlap due to their inherent linguistic characteristics and content patterns. This variation in n-gram overlap presents a crucial challenge for membership inference: when non-member texts contain substantial subsequences that also appear in the training data (member texts), the boundary between what constitutes seen versus unseen content becomes increasingly blurred. While most domains of the Pile dataset and their samples present in the MIMIR benchmark have an 7-gram overlap of 30-40%, two of the domains Github and DM Mathematics have much higher overlaps which exceed 73%. Therefore, to systematically analyze the effect of this n-gram overlap on membership inference, we design a comparative experiment using two distinct subsets from these domains: a restricted subset where non-members are carefully sampled to ensure 13-gram overlap remains below 20%, and an unrestricted subset where 13-gram overlap is allowed to reach up to 80% (both provided in the MIMIR benchmark) (Duan et al., 2024) and compare their performance in Table 4. As expected, we observe improved membership inference performance across all model sizes when the n-gram overlap between members and non-members is more restricted. More broadly, these results suggest that domains with inherently high textual redundancy may provide natural resistance to some attacks, while domains with more unique content patterns may be more vulnerable to such attacks.

4 Related Work

This section surveys prior work on membership inference attacks against machine learning models, with a focus on methods applicable to large language models. We begin by defining key concepts, then organize existing approaches into reference-free and reference-based families, discuss methods tailored to LLMs and conclude by contrasting these with our proposed hidden-state contrast framework.

Membership inference attacks vary along three critical dimensions: (i) **model access** (black-box vs. white-box), where black-box refers to accessing only model outputs while white-box permits inspection of model's internal parameters; (ii) **reference requirements**, distinguishing between methods that require auxiliary models or datasets and those that operate independently; and (iii) **inference signals** used to detect membership, ranging from simple loss values to complex representation patterns.

Reference-Free Methods The simplest membership inference approaches rely solely on the target model's behavior without external calibration. We review the key attacks in this category. Loss or Perplexity Attack, also known as the PPL or Loss attack, method identifies low loss (or perplexity) as an indicator of membership. The underlying assumption is that data points seen during training will result in lower loss values compared to unseen examples. However, this simple approach often performs poorly because non-member sequences, especially those containing frequent or repetitive text, can also achieve low perplexity (Yeom et al., 2018; Carlini et al., 2022). The Min-K% Prob method (Shi et al., 2024) is a reference-free MIA designed to overcome the limitations of simpler attacks by focusing on the tokens within a sequence that the model is least confident about. Min–K% PROB computes the average log-probability of the lowest-probability tokens, while Min–K%++ further amplifies local maxima in the probability distribution (Zhang et al., 2024). These methods improve over plain PPL without requiring any reference model. The Neighborhood attack proposes using probability curvature, for each candidate x, generating semantically similar neighbors via perturbation and compare their average loss. Low curvature (i.e. small change in loss in the neighborhood) signals memorization (Mattern et al., 2023). DetectGPT adapts this idea zero-shot using an external language model that infills randomly masked spans of the original text to identify non members from the training text (Mitchell et al., 2023). Zlib and Lowercase attacks compare perplexity on the original x to compressed or lowercased variants (Carlini et al., 2021). The intuition is that members, having been seen during training, yield smaller perplexity increases under such transformations. Bertran et al. (2023) propose a new family of MIAs that are competitive with state-of-the-art shadow model approaches like LiRA (Carlini et al., 2022) while requiring substantially fewer computational resources and less knowledge of the target model's ar-

chitecture. This work focuses on the efficiency and scalability of MIAs, identifying pinball loss as a key target objective for effective attacks.

Reference-Based Methods These methods calibrate the target model's scores against those from a reference model trained on separate data. Likelihood-Ratio Attacks (LiRA) computes a likelihood ratio between two hypotheses—"x was in training" vs. "x was not"—using scores from shadow/reference models. LiRA can achieve high precision but requires training many shadow models and a closely matched reference corpus (Carlini et al., 2022). Self-Prompted MIA (SPV-MIA) address LiRA's dependence on external data by prompting the target LLM itself to generate a reference dataset. They define a probabilistic variation metric that compares score distributions on generated vs. real examples, improving robustness when no true reference set is available (Fu et al., 2024). Robust MIA (RMIA) propose a statistical test under the null hypothesis that x is replaced by a random sample and then compose many pairwise likelihood ratio tests where each test compares the likelihood of the data point to the random sample. RMIA combines a small number of reference models with population samples to yield a computationally efficient and distribution-robust attack (Zarifzadeh et al., 2024). Fine-Tuned Score Deviation (FSD) fine-tune the target LLM on a small auxiliary corpus and measure how much perplexity of non-members decreases relative to members. This enlarges the membership signal and yields significant AUC gains in pre-training data detection (Zhang et al., 2025a). Test-Set Contamination Detection leverage dataset exchangeability suggesting if an LLM has seen an evaluation example, it prefers the canonical ordering of lines over shuffled inputs. This black-box test serves as a "provable" contamination check for benchmarks (Oren et al., 2024). PII Leakage Attacks systematically probe for personally identifiable information (PII) extraction, reconstruction, and inference. They find that differential privacy (DP) and PII scrubbing reduce but do not eliminate leakage (Lukas et al., 2023). MIA Tuner introduces a novel instruction-based MIA method that enables LLMs themselves to serve as more precise pre-training data detectors by using instruction-tuning (SFT) to adapt LLMs to directly answer whether a text belongs to their pre-training dataset or not (Fu et al., 2025).

Efficacy of MIAs Recent studies (Das et al., 2025; Maini et al., 2024; Duan et al., 2024) suggest MIAs often perform barely better than random guessing on LLM pre-training data. Skeptics attribute this to massive training datasets with few iterations, blurry boundaries between member and non-member texts due to n-gram overlap in natural language corpora, and potential confounding factors like detecting temporal distribution shifts rather than true membership. Puerto et al. (2024) also challenge the notion that MIAs do not work on LLMs, arguing that they do work but only when multiple documents are presented for testing, rather than short sequences. They construct new benchmarks that measure MIA performance at continuous scales, from sentences to collections of documents.

In contrast, our work demonstrates that membership signals exist but are primarily encoded in internal model dynamics rather than outputs. This approach reveals detectable membership information in LLMs' processing pathways without requiring the model fine-tuning. Our results show that deeper transformer layers harbor increasingly strong membership signals, advancing both theoretical understanding and practical capabilities of membership inference in large language models.

5 Conclusion

In this work, we show that membership inference attacks are effective against large language models by demonstrating that membership signals are strongly encoded in models' internal representations, even when output distributions show minimal differences. Our framework achieves strong membership detection performance by analyzing how information flows through transformer architectures, revealing that models process familiar content differently than novel content. Through comprehensive evaluation across multiple model architectures and diverse text domains, we show that these internal processing signatures provide more reliable membership detection than traditional output-based approaches. These findings have important implications for privacy research in large language models: first, they demonstrate that privacy auditing must consider internal model dynamics rather than focusing solely on outputs; second, they suggest that effective privacy-preserving training techniques will need to address how models process information, not just regulate their final predictions. As language models continue to be deployed in privacy-sensitive domains, our work highlights the need for more sophisticated approaches to understanding and protecting against potential memorization vulnerabilities. A related challenge in the field is the absence of a standardized metric to evaluate the efficacy of machine un-learning techniques. This is largely due to the wide variety of datasets and incomparable benchmarks used across studies, making it difficult to assess whether a model has truly forgotten a specific piece of data. Our framework may contribute to addressing this gap.

Several promising directions could extend the impact of this work. Exploring whether similar processing signatures can be detected through black-box probing techniques will be pursued as future work. Expanding this investigation to other architectural families beyond decoder-only transformers could reveal interesting insights about how different model types process familiar information, and understanding the relationship between these processing patterns and other forms of information leakage could provide valuable directions for improving model security and privacy-preserving training methods.

Limitations

Our current framework requires access to model internals, which naturally constrains its application to scenarios where such access is available. Additionally, our analysis has primarily focused on decoder-only transformer architectures, which represents a specific, though widely used, family of language models. We also note that our current investigation has not extensively explored how various fine-tuning approaches might affect the memorization signatures we've identified.

References

Martin Andres Bertran, Shuai Tang, Aaron Roth, Michael Kearns, Jamie Heather Morgenstern, and Steven Wu. 2023. Scalable membership inference attacks via quantile regression. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Stella Biderman, Hailey Schoelkopf, Quentin Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, Aviya Skowron, Lintang Sutawika, and Oskar Van Der Wal. 2023. Pythia: a suite for analyzing large language models across training and scaling. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org.

Sid Black, Leo Gao, Phil Wang, Connor Leahy, and Stella Mesh-Tensorflow Biderman. 2021. GPT-Neo: Large Scale Autoregressive Language Modeling with Mesh-Tensorflow.

Nicholas Carlini, Steve Chien, Milad Nasr, Shuang Song, Andreas Terzis, and Florian Tramer. 2022. Membership inference attacks from first principles. *Preprint*, arXiv:2112.03570.

Nicholas Carlini, Florian Tramer, Eric Wallace, Matthew Jagielski, Ariel Herbert-Voss, Katherine Lee, Adam Roberts, Tom Brown, Dawn Song, Ulfar Erlingsson, and 1 others. 2021. Extracting training data from large language models. In *30th USENIX security symposium (USENIX Security 21)*, pages 2633–2650.

Debeshee Das, Jie Zhang, and Florian Tramèr. 2025. Blind baselines beat membership inference attacks for foundation models.

Michael Duan, Anshuman Suri, Niloofar Mireshghallah, Sewon Min, Weijia Shi, Luke Zettlemoyer, Yulia Tsvetkov, Yejin Choi, David Evans, and Hannaneh Hajishirzi. 2024. Do membership inference attacks work on large language models? In *Conference on Language Modeling (COLM)*.

Wenjie Fu, Huandong Wang, Chen Gao, Guanghua Liu, Yong Li, and Tao Jiang. 2024. Membership inference attacks against fine-tuned large language models via self-prompt calibration. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Wenjie Fu, Huandong Wang, Chen Gao, Guanghua Liu, Yong Li, and Tao Jiang. 2025. MIA-tuner: Adapting large language models as pre-training text detector. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Philadelphia, Pennsylvania, USA.

Nils Lukas, Ahmed Salem, Robert Sim, Shruti Tople, Lukas Wutschitz, and Santiago Zanella-Béguelin. 2023. Analyzing leakage of personally identifiable information in language models. In *2023 IEEE Symposium on Security and Privacy (SP)*, pages 346–363. IEEE Computer Society.

Pratyush Maini, Hengrui Jia, Nicolas Papernot, and Adam Dziedzic. 2024. LLM dataset inference: Did you train on my dataset? In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Justus Mattern, Fatemehsadat Mireshghallah, Zhijing Jin, Bernhard Schoelkopf, Mrinmaya Sachan, and Taylor Berg-Kirkpatrick. 2023. Membership inference attacks against language models via neighbourhood comparison. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 11330–11343, Toronto, Canada. Association for Computational Linguistics.

Eric Mitchell, Yoonho Lee, Alexander Khazatsky, Christopher D. Manning, and Chelsea Finn. 2023. Detectgpt: zero-shot machine-generated text detection using probability curvature. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org.

Yonatan Oren, Nicole Meister, Niladri S. Chatterji, Faisal Ladhak, and Tatsunori Hashimoto. 2024. Proving test set contamination in black-box language models. In *The Twelfth International Conference on Learning Representations*.

Haritz Puerto, Martin Gubri, Sangdoo Yun, and Seong Joon Oh. 2024. Scaling up membership inference: When and how attacks succeed on large language models. *Preprint*, arXiv:2411.00154.

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

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. Llama: Open and efficient foundation language models. *Preprint*, arXiv:2302.13971.

Roy Xie, Junlin Wang, Ruomin Huang, Minxing Zhang, Rong Ge, Jian Pei, Neil Zhenqiang Gong, and Bhuwan Dhingra. 2024. ReCaLL: Membership inference via relative conditional log-likelihoods. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 8671–8689, Miami, Florida, USA. Association for Computational Linguistics.

Samuel Yeom, Irene Giacomelli, Matt Fredrikson, and Somesh Jha. 2018. Privacy risk in machine learning: Analyzing the connection to overfitting. *Preprint*, arXiv:1709.01604.

Sajjad Zarifzadeh, Philippe Cheng-Jie Marc Liu, and Reza Shokri. 2024. Low-cost high-power membership inference by boosting relativity.

Hengxiang Zhang, Songxin Zhang, Bingyi Jing, and Hongxin Wei. 2025a. 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. 2024. Min-k%++: Improved baseline for detecting pre-training data from large language models. *arXiv preprint arXiv:2404.02936*.

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

Supplement for "Neural Breadcrumbs: Membership Inference Attacks on LLMs Through Hidden State and Attention Pattern Analysis"

A Dataset Details

We evaluate our approach on established membership inference benchmarks spanning diverse domains. The MIMIR benchmark (Duan et al., 2024) serves as a unified repository for evaluating membership inference attacks on LLMs. It provides a unified benchmark package that includes all existing MIAs, supporting future work. The MIMIR benchmark is constructed out of The Pile dataset, an 800GB collection of diverse text specifically curated for language modeling. Experiments within MIMIR utilize seven distinct data sources from The Pile, including knowledge sources (Wikipedia), academic papers (PubMed Central), dialogues (HackerNews), and specialized domains (DM Math, Github). For each domain, the benchmark consistently samples 1000 members from The Pile's training set and 1000 non-members from its test set. For our evaluation, we utilize specific subsets from the MIMIR benchmark that are derived from The Pile dataset. These include Wikipedia, GitHub, PubMed Central, HackerNews, and DM Mathematics. These domains were chosen because they represent diverse text sources with varying linguistic characteristics and structural properties. Members and non-members for these datasets are sampled from the train and test sets of The Pile, respectively. The MIMIR benchmark is designed to reflect the inherent difficulty of MIAs against LLMs. When referring to the 13-gram 0.8 split version, this indicates that the benchmark examples used have undergone a default decontamination process where non-member samples with greater than 80% 13-gram overlap with the training data were filtered out. However, even with this decontamination, natural language documents commonly exhibit repeating text and substantial n-gram overlap between members and non-members. For instance, domains like Wikipedia, ArXiv, and PubMed Central, even after deduplication, show average 7-gram overlaps exceeding 30% with the training data. This inherently fuzzy boundary and high n-gram overlap significantly decreases MIA performance for most settings across varying LLM sizes and domains. The sources suggest that non-members with lower n-gram overlap are more distinguishable by existing MIAs, and strictly resampling non-members to ensure very low n-gram overlap (e.g., $\leq 20\%$) can lead to a significant increase in MIA performance, as discussed in Section 4 of the paper.

B Additional Results

This section presents comprehensive experimental results that could not be included in the main paper due to space constraints. We first outline our data preprocessing methodology to ensure reproducibility. Then, we present additional experimental results that expand our analysis in three key directions: First, we provide a detailed comparison of our method against all competitive baselines across the challenging benchmark datasets (Wikipedia, PubMed Central, Hacker News, GitHub) and across all Pythia model variants (from Pythia-70M to Pythia-6.9B). These extended results include statistical significance metrics for all comparisons, which could not be accommodated in the main paper's tables. Second, we broaden our evaluation to include two additional datasets with distinct characteristics: DM Mathematics from MIMIR benchmark, which has high n-gram overlap between members and non-members (greater than 70%) and features highly structured formal content with specialized notation, and arXivTection, which contains technical scientific writing. These datasets allow us to assess how membership inference attacks perform across diverse specialized domains with different linguistic patterns and vocabulary distributions. Third, we expand on the fine-tuned score deviation (FSD) baseline analysis. While the main paper presented results for this approach combined only with the perplexity attack (the most competitive configuration). This includes combinations with Zlib compression ratio, lowercase perplexity, and Min-K% scoring functions, offering insights into which underlying mechanisms benefit most from the fine-tuning approach across our expanded set of datasets. The results for existing datasets are included in Table 5 and the new datasets are included in Table 6.

Data Pre-processing For data-preprocessing for all experiments, we processed input sequences using each model's native tokenizer with standardized parameters. Sequences were truncated to a maximum length of 512 tokens across all datasets, with the exception of BookMIA where we extended this limit to 1024 tokens to accommodate its longer text passages. We applied consistent padding to achieve uniform sequence lengths within batches (batch size = 32). When extracting features for our membership inference method, we carefully respected the attention masks generated during preprocessing, ensuring that padded tokens did not contribute to our feature calculations. All preprocessing configurations were carefully matched to those used during the models' pre-training phases to ensure evaluation validity.

Together, these supplementary results provide a stronger empirical support for the conclusions drawn in the main paper.

C AI Usage

In the preparation of this manuscript, we utilized Large Language Models to assist with textual refinement, including improving clarity and readability of certain passages.

Method	Wikipedia				Pubmed Central			
	Pythia-70M	Pythia-410M	Pythia-1B	Pythia-6.9B	Pythia-70M	Pythia-410M	Pythia-1B	Pythia-6.9B
ReCaLL	0.51±0.03	0.50±0.04	0.52±0.04	0.54±0.04	0.52±0.04	0.50±0.03	0.50±0.03	0.51±0.03
MIATuner	0.49±0.03	0.52±0.01	0.51±0.02	0.50±0.03	0.51±0.02	0.50±0.01	0.49±0.02	0.51±0.02
FSD (PPL)	0.49±0.03	0.48±0.02	0.50±0.03	0.48±0.04	0.49±0.03	0.50±0.03	0.49±0.03	0.47±0.03
FSD (Lowercase)	0.49±0.03	0.48±0.02	0.49±0.03	0.49±0.03	0.48±0.02	0.48±0.01	0.51±0.02	0.52±0.01
FSD (Zlib)	0.49±0.03	0.48±0.01	0.51±0.03	0.48±0.02	0.49±0.03	0.50±0.03	0.49±0.04	0.47±0.03
FSD (Min-K%)	0.50±0.04	0.48±0.02	0.49±0.03	0.47±0.04	0.50±0.02	0.50±0.03	0.49±0.03	0.49±0.02
memTrace (Ours)	**0.65±0.05**	**0.87±0.01**	**0.89±0.02**	**0.89±0.01**	**0.59±0.04**	**0.89±0.02**	**0.88±0.02**	**0.84±0.01**
	Hacker News				**Github**			
ReCaLL	0.53±0.01	0.52±0.02	0.53±0.04	0.54±0.04	0.62±0.03	0.67±0.02	0.68±0.01	0.73±0.02
MIATuner	0.51±0.02	0.51±0.01	0.53±0.02	0.50±0.04	0.60±0.02	0.67±0.02	0.71±0.03	0.73±0.02
FSD (PPL)	0.50±0.01	0.50±0.03	0.50±0.02	0.48±0.04	0.46±0.02	0.47±0.01	0.42±0.03	0.37±0.03
FSD (Lowercase)	0.49±0.03	0.49±0.03	0.50±0.02	0.50±0.02	0.50±0.02	0.52±0.03	0.46±0.02	0.48±0.03
FSD (Zlib)	0.50±0.02	0.50±0.04	0.50±0.02	0.47±0.03	0.49±0.02	0.49±0.02	0.45±0.04	0.42±0.04
FSD (Min-K%)	0.51±0.02	0.50±0.04	0.49±0.02	0.48±0.04	0.49±0.02	0.47±0.03	0.42±0.04	0.40±0.03
memTrace (Ours)	**0.82±0.04**	**0.81±0.03**	**0.82±0.04**	**0.83±0.01**	**0.67±0.03**	**0.70±0.03**	**0.73±0.03**	**0.77±0.01**

Table 5: Comparison of test AUC scores (mean ± standard deviation) across methods and datasets using Pythia models of varying sizes. FSD variants use different base methods: perplexity (PPL), lowercase perplexity, Zlib compression ratio, and Min-K% probability. Higher values indicate better membership inference performance.

Method	DM Mathematics				arXivTection			
	Pythia-70M	Pythia-410M	Pythia-1B	Pythia-6.9B	Pythia-70M	Pythia-410M	Pythia-1B	Pythia-6.9B
ReCaLL	0.50±0.02	0.50±0.02	0.51±0.02	0.51±0.03	0.59±0.05	0.67±0.05	0.69±0.06	0.72±0.06
MIATuner	0.49±0.04	0.50±0.03	0.50±0.02	0.49±0.02	0.60±0.02	0.89±0.01	**0.99±0.01**	0.96±0.01
FSD (PPL)	0.51±0.01	0.50±0.04	0.48±0.03	0.49±0.03	0.77±0.05	**0.94±0.01**	0.96±0.00	**0.98±0.00**
FSD (Lowercase)	0.50±0.02	0.51±0.03	0.49±0.02	0.50±0.04	0.63±0.04	0.81±0.02	0.84±0.01	0.86±0.01
FSD (Zlib)	0.51±0.01	0.50±0.03	0.47±0.03	0.49±0.02	0.76±0.04	**0.94±0.01**	0.96±0.00	**0.98±0.00**
FSD (Min-K%)	0.49±0.02	0.51±0.02	0.49±0.03	0.50±0.02	0.71±0.05	0.87±0.01	0.91±0.00	0.93±0.01
memTrace (Ours)	**0.62±0.03**	**0.69±0.03**	**0.69±0.03**	**0.74±0.02**	**0.80±0.02**	0.91±0.01	0.92±0.01	0.94±0.02

Table 6: Additional datasets evaluation: Comparison of test AUC scores (mean ± standard deviation) on DM Mathematics and arXivTection datasets.