AUDITING INFORMATION DISCLOSURE DURING LARGE-SCALE GRADIENT-BASED TRAINING VIA GRADIENT UNIQUENESS

Anonymous authors
Paper under double-blind review

ABSTRACT

Auditing information disclosure across every datapoint during the training of LLMs is challenging. We propose a principled, attack-agnostic approach that uses mutual information to measure what the final model reveals about a datapoint's training membership. We show that this final-model disclosure is upper bounded by the sum of per-iteration gradient disclosures and that, under a reasonable set of assumptions, these gradient disclosures increase with Gradient Uniqueness (GNQ), which measures how distinguishable a datapoint's gradient is relative to other gradients in the batch. While naively computing GNQ requires forming and inverting a $P \times P$ matrix for every datapoint (for a model with P parameters), we introduce Batch-Space Ghost (BS-Ghost). This efficient algorithm performs all computations in a much smaller batch space and uses ghost kernels to compute GNQ "in-run" for every datapoint in the training corpus, with minimal computational and memory overhead. Our experiments show the following: (i) GNQ predicts MIA vulnerability without the need for shadow models. (ii) Beyond membership disclosure, GNQ predicts the success of reconstruction attacks. (iii) GNQ-guided removal and retraining identify datapoints that causally contribute to disclosure. (iv) GNQ outperforms counterfactual memorization in text extraction and common-knowledge discrimination without the need for additional model training. (v) For data attribution, GNQ-guided filtering reduces emergent misalignment in Qwen2.5-7B more than baselines. Further, GNQ attributes 1000 datapoints in 17 seconds—roughly $70\times$ faster than the baselines. (vi) GNQ explains how training choices affect training-set disclosure and captures how per-datapoint disclosure emerges during training.

1 INTRODUCTION

After training, a deployed model may reveal information about specific training datapoints. This can happen when analysis or querying of the model can reveal training text verbatim or near-verbatim (Carlini et al., 2019; 2021; Nasr et al., 2025; Hayes et al., 2025), or when personally identifiable information (PII) is embedded in model output (e.g., leaking a phone number inside a new sentence) (Zhou et al., 2024; Kim et al., 2023; Lukas et al., 2023). Such disclosures enable training-data extraction and related privacy attacks (Ishihara, 2023), and may violate contractual or legal constraints on data usage (Henderson et al., 2023; Cooper & Grimmelmann, 2025). For these reasons, it is important to be able to *audit* how much disclosure an AI model poses.

We consider the problem of designing a practical approach for auditing the extent to which an AI model encodes information about each of the datapoints it was trained on—where "information" is defined formally, via an information-theoretic analysis. Our goal is to design an auditing framework that meets the following requirements (Table 1):

(C1) Auditing should be attack-agnostic. Measuring the extent to which a model resists a given *attack* is perhaps the most common way to audit disclosure. Examples include prompting-based extraction and string-match pipelines (Carlini et al., 2021; Nasr et al., 2025; Hayes et al., 2025), membership inference attacks (MIAs) that predict whether a candidate datapoint was in the training set (Shokri et al., 2017; Mattern et al., 2023; Xie et al., 2024), and other reconstruction or PII-focused probes (Lukas et al., 2023; Kim et al., 2023). However, such methods are attack-specific: failure of one attack does not imply safety against others (Song & Mittal, 2021; Carlini et al., 2022).

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.

(C2) Auditing should cover every training datapoint, "in-run" and at low cost. For LLM-scale datasets and models, it is impractical for auditing to require post-hoc analysis of the model with respect to every training datapoint, or heavyweight procedures that scale with the number of prompts, generations, or training-set substrings (Nasr et al., 2025; Hayes et al., 2025). Auditing should take place as the model is constructed ("in-run") and introduce low computational and memory overhead.

(C3) Auditing should not modify the baseline training and should audit the actual trained model. In realistic LLM settings, training is expensive and recipes are tightly tuned; practitioners are reluctant to alter data or optimization in ways that may affect utility, safety behavior, or training stability. Thus, auditing should not require changing the training dataset (e.g., inserting canaries) (Carlini et al., 2019; Parikh et al., 2022) or running a *different* training procedure to measure disclosure.

(C4) Auditing should take into account prior/common knowledge. Consider "Napoleon Bonaparte lost the Battle of Waterloo on June 18, 1815." Any LLM trained on a large corpus would be able to reproduce this sequence, regardless of whether it had seen the sequence during training. Declaring all such "common knowledge" data to be disclosed/memorized is unhelpful.

In this paper, we propose a new method for disclosure auditing that meets the requirements above, called *gradient uniqueness* (GNQ). Our approach starts from a principled, attack-agnostic measure of disclosure: the mutual information $I(T_j; \theta_{N_r})$ between a datapoint's membership and the final model (C1). Given a population prior over training samples unknown to the adversary, we show that this disclosure is bounded by the sum of per-iteration gradient disclosures (Equation (2)). Under a reasonable set of assumptions, per-iteration disclosure increases with GNQ (Theorem 2), which is computable directly from gradients within a single training run. Our Batch-Space Ghost (BS-Ghost) algorithm computes GNQ for every training datapoint "in-run," with very low overhead (C2). Moreover, GNQ does not require modifying the training algorithm, and it audits the disclosure of the model produced by the training run (C3). GNQ accounts for common knowledge by measuring how distinguishable a datapoint's gradient is relative to gradients from the training population (C4).

Our specific contributions are as follows:

- We bound final-model membership disclosure by summed per-iteration gradient disclosures (Theorem 1), then show that these disclosures increase with GNQ under reasonable assumptions (Theorem 2), yielding a per-datapoint score independent of any attack strategy.
- We introduce BS-Ghost, which makes auditing *every* datapoint in the training corpus practical at LLM scale through batch-space operations and ghost kernels, without materializing per-datapoint gradients.
- We connect attack-agnostic GNQ to leakage, memorization, and attribution of training data. GNQ captures the membership vulnerability revealed by LiRA without the need for shadow models and identifies extractable training text more effectively than counterfactual memorization (Zhang et al., 2023): all 20 highest-GNQ sentences are recovered, versus 7 of 20 selected by counterfactual memorization. For data attribution, GNQ-guided filtering reduces emergent misalignment in Qwen2.5-7B (Qwen et al., 2025) more than baselines (Chen et al., 2025), while attributing 1000 datapoints takes about 17 seconds, roughly $70\times$ faster than both EK-FAC (Grosse et al., 2023) and Concept Influence (Kowal et al., 2026).

2 Related Work

Attack-based evaluations of disclosure. A large body of work measures disclosure by probing a *single trained model instance* with a particular evaluation or attack family. For language models, this includes prompting-based extraction and verification pipelines that recover memorized spans or documents (Carlini et al., 2021; Nasr et al., 2025; Hayes et al., 2025; Yu et al., 2023), and formalizations connecting memorization to discoverability/extractability of training sequences (Carlini et al., 2023). Other empirical probes focus on near-duplicate matching and scalable search procedures used to surface memorized or replicated content (Peng et al., 2023; Ippolito et al., 2023). A related line audits leakage of entities or personally identifiable information (PII), where disclosure can occur via associations even when surface form differs (Zhou et al., 2024; Kim et al., 2023; Lukas et al., 2023). In parallel, privacy leakage is commonly operationalized through membership inference attacks (MIAs), which test whether a candidate datapoint was used for training, including black-box MIAs (Shokri et al., 2017), language-model-specific variants (Mattern et al., 2023; Xie et al., 2024),

Prior work	Desiderata C1	C2	C3	C4
Attack-based: (Carlini et al., 2021; Nasr et al., 2025; Hayes et al., 2025; Yu et al., 2023; Carlini et al., 2023; Peng et al., 2023; Ippolito et al., 2023; Zhou et al., 2024; Kim et al., 2023; Lukas et al., 2023; Shokri et al., 2017; Mattern et al., 2023)	✗	✗	✓	✗
Canary-based: (Carlini et al., 2019; Parikh et al., 2022; Steinke et al., 2023)	✗	✗	✗	✗
Counterfactual: (Zhang et al., 2023; Feldman, 2020; Feldman & Zhang, 2020)	✓	✗	✗	✗

Table 1: Disclosure auditing approaches versus desiderata defined in the Introduction (C1–C4).

and strong statistical attacks such as LiRA (Carlini et al., 2022). These methods are valuable for demonstrating concrete failure modes, but they are inherently attack/evaluation-specific.

Canary and exposure-style tests. Another line of work audits disclosure by inserting synthetic secrets ("canaries") into the training data and then measuring their leakage. The exposure metric and related canary-based tests provide a clear operational signal when the audit target is an injected string (Carlini et al., 2019), including NLU-focused canary extraction settings (Parikh et al., 2022). Steinke et al. (2023) propose privacy auditing with one training run, again centered on canaries, enabling an efficient single-run audit of leakage for selected secrets. While these approaches can be carried out in-run, they are not attack-agnostic: the audit procedure is itself an attack protocol.

Counterfactual memorization and dataset commonality. A distinct line of work defines memorization *counterfactually* as the causal effect of including a training example on model behavior, estimated by comparing *different training runs* on datasets that include versus exclude that example (Zhang et al., 2023). This perspective is designed to disentangle memorization of *rare* content from effects driven by *dataset commonality*, in particular near-duplicates or templated strings that occur across many documents: removing a single instance of highly repeated text should have only a small counterfactual effect. Importantly, "common" in this setting refers to repetition within the *training corpus* (duplication), which is conceptually distinct from "common knowledge" in the sense of well-known facts. Counterfactual memorization adapts to language models the leave-one-out memorization score of (Feldman, 2020) and (Feldman & Zhang, 2020). However, computing counterfactual memorization requires training many models on different subsets, making it computationally expensive at scale rather than a low-cost, in-run audit (C2). Moreover, it averages over training randomness instead of auditing the particular model and training trajectory produced by the actual run (C3).

Information bounds. Negrea et al. (2019) and Haghifam et al. (2020) use mutual information as an analysis tool for generalization bounds on noisy SGLD. In contrast, we connect per-iteration gradient disclosure to GNQ for auditing every training datapoint during ordinary mini-batch SGD.

3 Gradient Uniqueness

In this section, we give an overview of the statistical model underlying GNQ, the threat model, as well as our central results.

Assume that an attacker wishes to determine whether data point d_j appears in training set $\mathcal{D}_t$ by analyzing a learned model. We begin with a prior over $\mathcal{D}_t$. The datapoint d_j appears in $\mathcal{D}_t$ if and only if $T_j = 1$, where $T_j \sim \text{Bernoulli}(\pi)$ and $0 < \pi < 1$. The other $N_t - 1$ training datapoints are drawn independently from a population distribution $\mathcal{P}$, independently of T_j. $\mathcal{D}_t$ is drawn once and then used to train an arbitrary model using mini-batch gradient descent (Alg. 1).

Algorithm 1: Mini-Batch SGD

Input: Training set $\mathcal{D}_t$, AI model with parameter vector $\theta \in \mathbb{R}^P$, point-wise loss function $\ell[\theta, d]$, learning rate η, batch size B, number of training iterations N_r

Output: Optimized (final) model parameters θ_{N_r}

Initialize model parameters to θ_0.

Sample mini-batches $\{\mathcal{B}_i\}_{i=0}^{N_r-1}$ from $\mathcal{D}_t$ according to sampling distribution $\{\mathcal{B}_i\} \sim Batch(\mathcal{D}_t)$ with batch-size parameter B.

for $i = 0$ ***to*** $N_r - 1$ **do**

$\hat{g}_i = \frac{1}{B} \sum_{d_j \in \mathcal{B}_i} \nabla_\theta [\ell[\theta_i, d_j]]$

$\theta_{i+1} = \theta_i - \eta \cdot \hat{g}_i$

end

Threat model. The final model θ_{N_r} varies with the identities of the sampled training datapoints, target membership, and random batch selection. The adversary knows the target d_j, population

distribution $\mathcal{P}$, and training algorithm, and observes the final model θ_{N_r}. It does not know the sampled non-target datapoints or batch selections. Given this, it tries to infer T_j.

Given this threat model, our goal is to bound the mutual information $I(T_j; \theta_{N_r})$. This quantity is an attack-agnostic measure of the amount of disclosure regarding the presence of d_j in the data. By the data-processing inequality, any attack based on this observation reveals at most this much membership information (Cover & Thomas, 2006). An auditor (model owner) sees the training datapoints and attempts to compute a bound on this mutual information during the unmodified run.

As we argue in the Appendix to the paper, under a reasonable set of assumptions, GNQ can be used to bound this quantity, where GNQ is defined as follows.

Definition 1 (Gradient Uniqueness (GNQ))**.** *Consider training batch i. The* gradient uniqueness *of datapoint d_j with respect to batch i is*

$$GNQ_{ij} = g_{ij}^\top S^{-1} g_{ij}, \tag{1}$$

where $S = \sum_{\substack{k=1 \\ k \neq j}}^{N_t} g_{ik} g_{ik}^\top + \lambda I_P \in \mathbb{R}^{P \times P}$ and $g_{ij} = \nabla_\theta \ell(\theta_i, d_j) \in \mathbb{R}^P$. Here $\lambda > 0$ is a small regularizer that makes the matrix invertible, and I_P is the $P \times P$ identity matrix.

Given this definition, Theorem 1 (Appendix A.1) bounds final-model membership disclosure by summed per-iteration gradient disclosures:

$$I(T_j; \theta_{N_r}) \leq \sum_{i=0}^{N_r - 1} I(T_j; \widehat{g}_i \mid \theta_i). \tag{2}$$

In this inequality, the per-iteration gradient disclosure $I(T_j; \widehat{g}_i \mid \theta_i)$ measures the information that the batch gradient $\widehat{g}_i$ reveals about T_j beyond the current model θ_i.

The adversary's uncertainty at one iteration. Fix $\theta_i = \theta$ and write the update as

$$\widehat{g}_i \;=\; W_i + \tfrac{1}{B}\, T_j M_{ij}\, g_{ij}, \qquad W_i = \tfrac{1}{B} \sum_{k \neq j} M_{ik}\, g_{ik},$$

where $M_{ik} \in \{0, 1\}$ indicates whether datapoint k is sampled into batch i (independently, with probability $q = B/N_t$), and W_i is the contribution of the other training datapoints. To an adversary who knows θ and d_j, the target's contribution is a known vector g_{ij}/B "turned on" by random variable $T_j M_{ij}$. The background W_i, by contrast, is a sum of gradients of datapoints the adversary cannot see during an attack, drawn from $\mathcal{P}$. It is therefore random from the adversary's point of view and $T_j M_{ij}$ must be detected as a shift of size g_{ij}/B in the sampled value. To measure the information available in this signal, we model the adversary's uncertainty about W_i as Gaussian,

$$W_i \mid \theta_i = \theta \;\sim\; \mathcal{N}\big(m_i,\, \tfrac{q}{B^2} S\big), \qquad W_i \perp (T_j, M_{ij}) \mid \theta_i = \theta,$$

with S as in Definition 1: $\frac{q}{B^2} S$ is an estimate of the covariance of W_i obtained from the observed second moment of the non-target gradients, which the auditor computes in-run. Appendix A.2 gives the central-limit and maximum-entropy motivation and states the assumption precisely. Under this predictive model we have Theorem 2 (Appendix A.2). Fix $\theta_i = \theta$ and let $\rho_i = \Pr(T_j = 1 \mid \theta_i = \theta)$, with $0 < \rho_i, q < 1$. Then, under the predictive model,

$$I(T_j; \widehat{g}_i \mid \theta_i = \theta) \;=\; I\Big(T_j;\; \sqrt{GNQ_{ij}/q}\; T_j M_{ij} + Z\Big), \quad Z \sim \mathcal{N}(0,1) \text{ independent of } (T_j, M_{ij}),$$

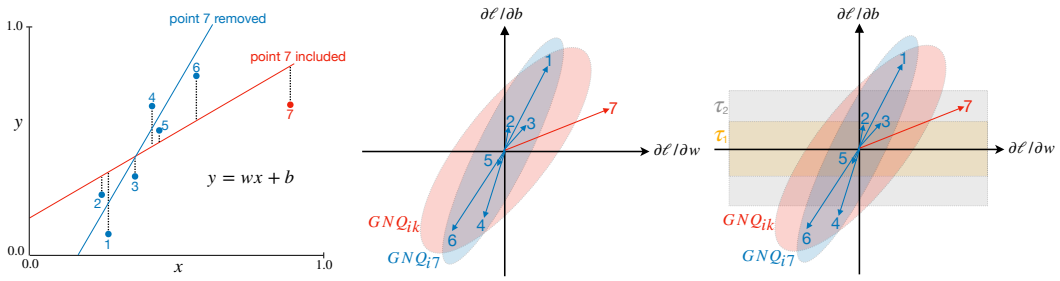
and the right-hand side is a strictly increasing function of GNQ_{ij}.

In other words, once the update is noised using the background covariance, the adversary has access to a one-dimensional Gaussian channel whose signal-to-noise ratio is GNQ_{ij}/q.

GNQ bounds disclosure. If we write the right-hand side of Theorem 2 as a function f_i of GNQ_{ij} that increases with GNQ[1] and then substitute into (2), we have

$$I(T_j; \theta_{N_r}) \;\leq\; \sum_{i=0}^{N_r - 1} \mathbb{E}\big[f_i(GNQ_{ij})\big]. \tag{3}$$

[1]Note that it also depends on ρ_i and q. However, because ρ_i is unobserved, f_i may be replaced by its supremum over $\rho_i \in [0, 1]$, which is still non-decreasing in GNQ and bounds the disclosure at every realization



(a) Data space. (b) Gradient space. (c) GNQ- vs. attack-based auditing.

Figure 1: What does GNQ Measure?

Thus, under the model described, membership disclosure of the released model regarding the inclusion of d_j is bounded by a value that grows with the GNQ scores of d_j along the training trajectory, averaged over the randomness of training. The auditor observes one trajectory and its GNQ scores, so the sum along that trajectory, over the iterations at which Algorithm 2 produces a score for d_j, is a single sample of the quantity being averaged over, and so it is a natural plug-in estimator.

Note that the bound is one-sided, so small GNQ certifies small disclosure under the model. Large GNQ means that the data point *might* be at risk of disclosure (developing a two-sided bound would require taking into account possible forgetting, which is beyond the scope of this paper). Section 6 experimentally examines this issue.

4 What Does GNQ Measure?

To show how GNQ measures information disclosure during gradient descent and to give some intuition behind the metric, we consider a simple example—a 2D linear regression model with the squared loss function (Fig. 1 a). The goal is to audit disclosure by ranking the datapoints by GNQ.

GNQ-based Auditing. The computation of GNQ can be geometrically represented as the construction of an ellipse summarizing the gradients; GNQ_{ij} is the extent to which the gradient associated with d_j is an outlier with respect to this ellipse, as shown in Fig. 1 b. In this figure, we plot the gradients of each of the seven training points, and the associated ellipses. Note that in the definition of GNQ_{ij}, the matrix S (and the resulting ellipse) is constructed using all the datapoints, except the point d_j for which GNQ_{ij} is computed. Thus there are two ellipses in Fig. 1 b: the blue ellipse is for the case when point 7 is excluded to compute GNQ_{i7}, while the red ellipse is for the case when point 7 is included while excluding one of the other six points to compute GNQ_{ik} where $k \in \{1, \cdots, 6\}$. Datapoint 7 has a very high GNQ value because it falls outside of the blue ellipse, while all other datapoints have a low GNQ value, as they fall inside of the red ellipse. In Fig. 1a, gradients from points 1–6 push the regression line counter-clockwise, roughly around datapoint 5. Datapoint 7 pushes clockwise, explaining its high GNQ_{i7}.

GNQ-based auditing vs attack-based auditing. MIA with a global loss threshold can produce very different rankings from GNQ. Consider Fig. 1c. In gradient space, the global loss threshold $\ell = \tau$ appears as a horizontal strip bounded by the two lines $\partial\ell/\partial b = \pm\sqrt{2\tau}$. Points inside the strip $\left(|\partial\ell/\partial b| \leq \sqrt{2\tau}\right)$ are classified, by the MIA, as members in the training set; points outside are non-members. We consider two possible thresholds τ_1 and τ_2. If we use τ_1, points 2, 3, and 5 are inside the τ_1-strip, so they will be ranked as high disclosure points, which does not match the GNQ ranking. Moreover, points 1, 4, 6 and 7 are outside the τ_1-strip, so they will be classified safe points; this matches the GNQ ranking regarding points 1, 4, and 6, however it misranks point 7; the most crucial point in terms of information disclosure according to GNQ. Using τ_2, points 2, 3, 5 and 7 are inside the τ_2-strip, so they will be ranked as high disclosure points—which does not match the GNQ ranking except for point 7. Moreover, points 1, 4 and 6 are outside the τ_2-strip, so their low-disclosure classification agrees with GNQ. Further, any ranking by loss alone depends solely on the residual (vertical distance to the fit), so points 2, 3, and 5—having smaller residuals (lower loss) than point 7—are necessarily ranked as higher disclosure for any choice of threshold. By contrast, GNQ, uses the full gradient geometry: it accounts for (1) the residual r, (2) the feature vector ($[x, 1]$ in our 2D example)—both via the gradient formula $g = -r[x, 1]$—and (3) inter-example correlations via S^{-1}.

5 IMPLEMENTATION: LLM-SCALE GNQ VIA BS-GHOST

The most direct implementation of GNQ follows Definition 1 almost literally: forming the $P \times P$ leave-one-out gradient second-moment matrices and inverting. This is impractical, as P can be in the billions. We show that it *is* practical to implement GNQ on a per-batch basis using standard matrix identities (push-through) together with a leave-one-out conversion (Sherman-Morrison). This removes the cubic dependence upon P. We also show how to compute the required batch-space quantities without explicitly forming the per-example gradient vectors (i.e. no extra backward passes required). A detailed time and memory complexity analysis is given in Appendix B.1.

We now describe the three key ideas that enable the efficient implementation in more detail.

Step (1) : B per-example gradients instead of N_t. Forming $S_{-j}^{(i)}$ requires N_t separate backward passes to compute per-example gradients. However, the mini-batch SGD algorithm itself uses the mini-batch gradient $\hat{g}_i = \frac{1}{B}\sum_{d_j \in \mathcal{B}_i} \nabla_\theta[\ell[\theta_i, d_j]]$ instead of the dataset gradient $g_i = \frac{1}{N_t}\sum_{j=1}^{N_t} \nabla_\theta[\ell[\theta_i, d_j]]$. Hence, we compute GNQ_{ij} using $S_{-j}^{(i)} = \sum_{\substack{d_k \in \mathcal{B}_i \\ k \neq j}} g_{ik} g_{ik}^\top + \lambda I_P$ instead of $S_{-j}^{(i)} = \sum_{\substack{k=1 \\ k \neq j}}^{N_t} g_{ik} g_{ik}^\top + \lambda I_P$. This reduces the number of per-example gradient vectors computed and materialized at each iteration. Under the population model of Section 3, both sums estimate the same second moment of the non-target gradients, the batch sum with B draws in place of $N_t - 1$.

Step (2) : Batch-Space Formulation. Fix an SGD step i with batch $\mathcal{B}_i$ (size B), parameters θ_i, and per-example gradients $\{g_{ij}\}_{j \in \mathcal{B}_i}$. Let $G \in \mathbb{R}^{B \times P}$ be the matrix whose j-th row is $g_{ij}^\top$. Define the regularized "total" matrix in parameter space and its batch-space Gram matrix:

$$S_{\text{tot}}^{(i)} \triangleq G^\top G + \lambda I_P \in \mathbb{R}^{P \times P}, \qquad K^{(i)} \triangleq GG^\top \in \mathbb{R}^{B \times B}.$$

For $\lambda > 0$, the push-through identity gives $(G^\top G + \lambda I_P)^{-1} G^\top = G^\top (GG^\top + \lambda I_B)^{-1}$. For each $j \in \mathcal{B}_i$, define $h_j^{(i)} \triangleq g_{ij}^\top (S_{\text{tot}}^{(i)})^{-1} g_{ij}$. With $g_{ij} = G^\top e_j$ (e_j the jth standard basis vector in $\mathbb{R}^B$), applying this identity gives

$$\begin{aligned} h_j^{(i)} &= e_j^\top G (G^\top G + \lambda I_P)^{-1} G^\top e_j \\ &= e_j^\top GG^\top (GG^\top + \lambda I_B)^{-1} e_j \\ &= \left[K^{(i)} \left(K^{(i)} + \lambda I_B \right)^{-1} \right]_{jj}. \end{aligned}$$

Thus $h^{(i)}$ is computable using only $B \times B$ matrices. Since $S_{\text{tot}}^{(i)} = S_{-j}^{(i)} + g_{ij} g_{ij}^\top$, Sherman–Morrison gives the following lemma (proof in Appendix B.2).

Lemma 1. *Whenever* $0 \leq h_j^{(i)} < 1$, $GNQ_{ij} = h_j^{(i)} / (1 - h_j^{(i)})$.

A single solve with $K^{(i)} + \lambda I_B$ yields GNQ for every datapoint in the batch, allowing the audit to cover the training corpus as its batches are processed.

Step (3) : Ghost Kernel Construction. The use of specialized "ghost kernels" enables practical computation of GNQ. Instead of materializing gradients, each ghost kernel reuses backpropagation results to compute a Gram matrix describing correlations between datapoints' gradients.

Algorithm 2: BS-Ghost GNQ

Input: Model at iteration i (θ_i), training batch $\mathcal{B}_i$ (size B), $\lambda > 0$
Output: $\text{GNQ}_i \in \mathbb{R}^B_{\geq 0}$: a vector containing all GNQ_{ij} values

`// 0. Hook setup (one-time)`
Register forward hooks to cache layer inputs $X^{(\ell)}$ and backward hooks to cache output errors $\delta^{(\ell)}$;

`// 1. No extra backprops`
Clear training caches;
Run the standard forward/backward on $\mathcal{B}_i$ (as required by SGD);
Hooks populate $\{X^{(\ell)}, \delta^{(\ell)}\}_{\ell=1}^L$;

`// 2. Build batch kernel`
$K^{(i)} \in \mathbb{R}^{B \times B}$
$K^{(i)} \leftarrow 0_{B \times B}$;
for *each instrumented layer* ℓ **do**
  Compute $K^{(\ell)}$ (e.g., via (4));
  $K^{(i)} \leftarrow K^{(i)} + K^{(\ell)}$;

`// 3. Solve in batch space`
Solve M from $\left(K^{(i)} + \lambda I_B\right) M = K^{(i)}$;
$h^{(i)} \leftarrow \text{diag}(M)$;
$\text{GNQ}_i \leftarrow h^{(i)} \oslash (1 - h^{(i)})$; `// Lemma 1`

GPT-2 on WikiText-2		
Method	Sec/iteration	Tokens/sec
GPT-2	0.53	3864
+ BS-Ghost	0.59	3471

Table 2: GPT-2 training cost w/ and w/o BS-Ghost.

Small Multi-Layer Perceptron				
Method	Extract (s)	GNQ (s)	Total (s)	MB
Naive	0.026	5.44	5.47	914
BS-Ghost	0.00	0.04	0.04	0.1

Table 3: Naive vs BS-Ghost on the MLP.

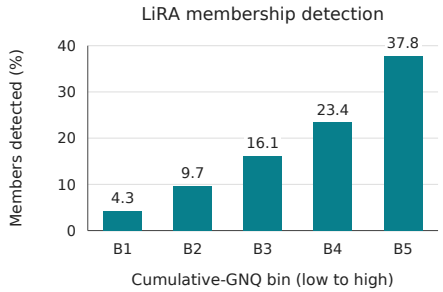


Figure 2: GNQ strongly predicts MIA success.

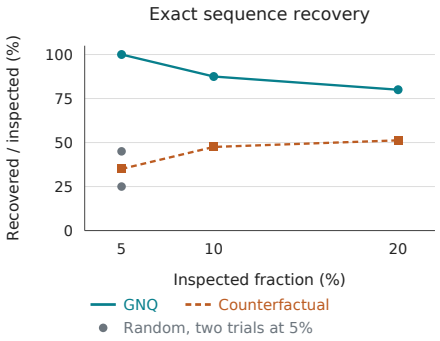


Figure 3: GNQ identifies extractable training text.

Note that to compute GNQ, we need a layer-specific ghost kernel for each instrumented layer (whether it is linear, convolutional, embedding, or another type). If not all trainable parameters are instrumented, the reported scores are lowered, and the argument that GNQ upper-bounds disclosure no longer applies. However, if the goal is simply to order training points by disclosure, such scores may still be useful. The final, comprehensive relationship matrix for the entire batch is simply the sum of these individual layer-wise kernels.

The mathematical trick of bypassing explicit gradient materialization was explored in "ghost clipping" used in differential privacy and the "ghost dot-product" used in data valuation (Lee & Kifer, 2021; Li et al., 2022; Wang et al., 2025). Our approach uses the ghost trick to construct the complete matrix in batch space for the leave-one-out inversion required by GNQ.

We now describe the required ghost kernel for a linear layer. Other ghost kernels can be implemented similarly—we describe the ghost kernels of the embedding and convolution layers in Appendix B.3. Consider a single-position linear layer ℓ with cached activations $X^{(\ell)} \in \mathbb{R}^{B \times n_{\text{in}}^{(\ell)}}$ and output backprop errors $\delta^{(\ell)} \in \mathbb{R}^{B \times n_{\text{out}}^{(\ell)}}$ from the training backward pass at step i. For sample j, the per-example weight gradient is $g_j^{(\ell)} = \delta_j^{(\ell)} {X_j^{(\ell)}}^\top$. Therefore, the layerwise contribution to the batch kernel is

$$K^{(\ell)} \;=\; \left(\delta^{(\ell)} {\delta^{(\ell)}}^\top\right) \odot \left(X^{(\ell)} {X^{(\ell)}}^\top + \mathbf{1}\mathbf{1}^\top\right). \tag{4}$$

Summing across instrumented layers yields $K^{(i)} = \sum_{\ell=1}^{L} K^{(\ell)}$.

6 Experiments

Correctness Of BS-Ghost. We use a small MLP to make naive GNQ's $P \times P$ inversions feasible. BS-Ghost reproduces its scores to numerical precision: $\max_j |\text{GNQ}_j^{\text{Ghost}} - \text{GNQ}_j^{\text{Naive}}| \approx 2 \times 10^{-10}$ for the same batch, parameters, and regularizer. Each naive matrix has about 20 million entries. This agreement checks the batch-space reformulation and ghost kernels against the GNQ definition.

Efficiency Of BS-Ghost. We benchmark GPT-2 Small on WikiText-2 ($B = 16$, sequence length 128) and compare BS-Ghost with naive GNQ on a 100–40–10 MLP trained on MNIST ($B = 32$), using a 16GB Tesla P100-SXM2 GPU—compared to more recent GPUs, the amount of memory and compute available on this GPU is very limited, making it an appropriate hardware platform to test efficiency. BS-Ghost adds about 11% to GPT-2 training time and accelerates MLP GNQ computation over $100\times$, while reducing peak memory from 914 to 0.1 MB (Tables 2 and 3). The extra peak memory for GPT-2 Small and Medium is below 0.1 MiB (Table 4, Appendix C.1). Attributing 1000 Qwen2.5-7B datapoints takes about 17 seconds, roughly $70\times$ faster than EK-FAC (Figure 9; Appendix C.6).

Validating GNQ As A Disclosure Measure. Membership and reconstruction attacks test the leakage identified by GNQ; removal and retraining test its causal connection.



(a) Training. (b) Before. (c) After.

Figure 4: GNQ-based filtering on Faces/MLP: removing the single highest-GNQ training datapoint in the target class makes reconstruction fail.

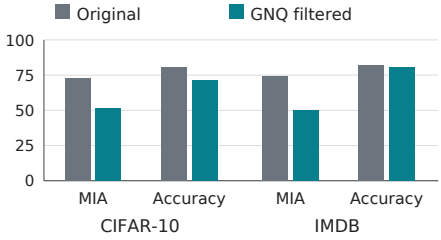


Figure 5: GNQ-based filtering reduces MIA success, with less accuracy loss on IMDB. Both metrics are percentages; lower MIA is better.

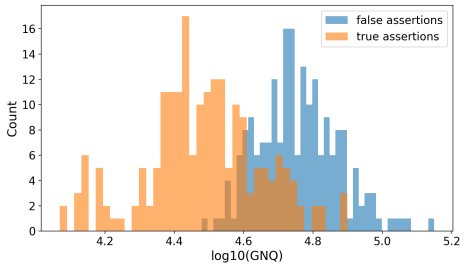


Figure 6: GNQ assigns lower memorization to common knowledge than to surprising assertions.

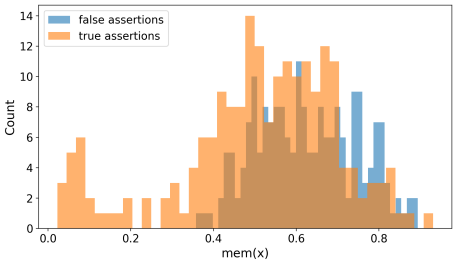


Figure 7: In contrast, counterfactual assigns comparable memorization scores to both groups.

Membership Disclosure. We test GNQ's information-theoretic interpretation by comparing GNQ accumulated during training with LiRA's detection of membership T_j from the final model. We use GPT-2 on WikiText-103, as in the original LiRA evaluation (Carlini et al., 2022). At a false-positive rate of 0.1%, LiRA's true-positive rate rises across all five cumulative-GNQ bins, from 4.3% in the lowest to 37.8% in the highest (Figure 2). GNQ matches this strong MIA's auditing conclusions without the need for shadow models, while remaining attack-agnostic.

Disclosure Beyond Membership. We test content disclosure through class-level model inversion (Fredrikson et al., 2015) on Faces/MLP and MNIST/ResNet (illustrated in Figure 4(b)), summing GNQ over each class's training datapoints and iterations. Higher-GNQ classes yield reconstructions closer to same-class training images in penultimate-layer features, with correlations of -0.90 and -0.96 between class GNQ and reconstruction distance (Figure 12(a,b)). The lowest-GNQ Faces class reconstructs poorly; the highest remains recognizable after one epoch at 10–30% test accuracy, showing disclosure despite underfitting (Figure 12(c,d), Appendix C.2). Agreement across attacks supports GNQ's principled, attack-agnostic design.

From Correlation To Causation. Removal and retraining establish a causal connection between GNQ and disclosure. For Faces/MLP, removing the single highest-GNQ training datapoint in the target class makes its reconstruction fail (Figure 4). Removing the highest-GNQ 10% on CIFAR-10/ResNet and 5% on IMDB/BERT reduces MIA success from about 73% to 51% and from 74% to 50%, respectively, with accuracy losses of about 9.5 and only 1.5 percentage points (Figure 5). At 50%, the adversary predicts membership at chance, so the MIA fails.

Validating GNQ As A Memorization Measure. We compare two attack-agnostic measures, GNQ and counterfactual memorization (Zhang et al., 2023), on fidelity and practicality.

Text Extraction. We collect GNQ while fine-tuning GPT-2 on 200 common-knowledge sentences and 200 surprising false assertions constructed with an LLM. We attempt exact recovery from a 12-token prefix (Carlini et al., 2023), selecting targets by maximum GNQ, counterfactual memorization, or randomly. This tests whether each score concentrates extractable sentences among its highest-ranked targets, where an auditor would inspect first. All 20 highest-GNQ sentences are recovered, versus 7 counterfactual targets and 5 or 9 in two random trials (Figure 3). Of 80 targets, GNQ recovers 64 versus 41 for counterfactual memorization (Appendix C.3, Table 5).

Common Knowledge. Common facts can be reproduced without memorizing the particular training sentence (C4). They receive lower GNQ than surprising assertions (Figure 6), consistent with less datapoint-specific memorization, while counterfactual scores overlap (Figure 7). This distinction matters because recovering a familiar fact need not reveal which training sentence supplied it.

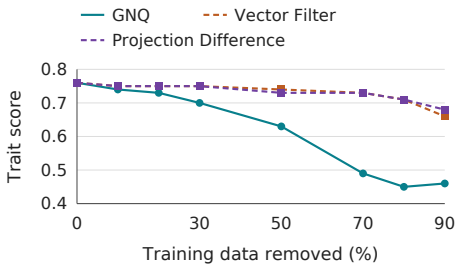


Figure 8: Qwen2.5-7B Bad Medical Advice after removal and retraining. Lower is better.

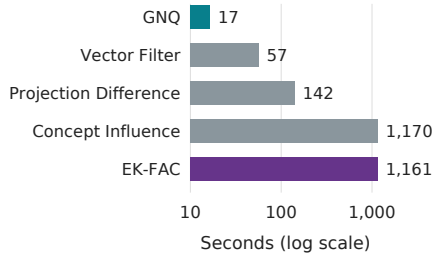


Figure 9: Attributing 1000 datapoints; baseline times from Kowal et al. (2026).

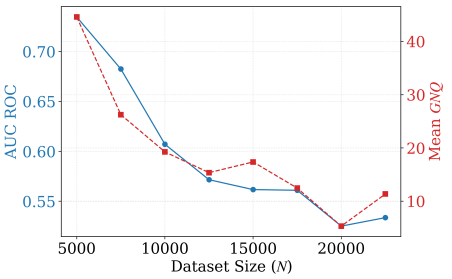


Figure 10: Dataset size versus GNQ and MIA AUC-ROC on CIFAR-10/CNN.

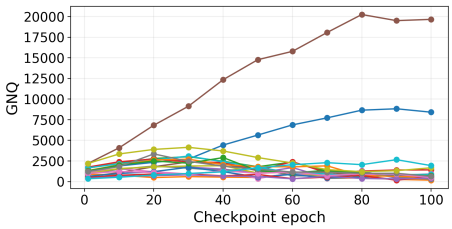


Figure 11: GNQ trajectories for 20 randomly selected AG-News training datapoints.

GNQ offers greater fidelity—better text recovery and common-knowledge discrimination—and greater practicality: BS-Ghost is in the same run, while counterfactual requires training many models.

Connecting Disclosure To Training Data Attribution. We compute GNQ for Qwen2.5-7B using datapoint gradients projected onto each trait's behavior vector, following Concept Influence (Kowal et al., 2026). The projection focuses the audit on gradient directions associated with that behavior. GNQ-guided removal and retraining reduce Bad Medical Advice more than either baseline at every tested nonzero fraction (Figure 8). GNQ's lower trait scores demonstrate that its ranking better identifies the datapoints contributing to the targeted behavior. Misaligned Opinions follows the same pattern (Appendix C.6, Figure 14). Future work can deepen this initial connection to attribution.

The experiments above connect GNQ to leakage, memorization, and data attribution. These connections are natural: all three depend on training data, and disclosure captures information about those data in the final model. Selecting the training data is one training choice; we next examine how model and optimization choices affect disclosure.

GNQ Explains How *Other* Training Choices Affect Disclosure. On CIFAR-10/CNN, increasing the dataset from 5000 to 22500 datapoints reduces mean GNQ (averaging per-datapoint maxima) fourfold, while MIA AUC-ROC falls from about 0.74 to 0.53, close to chance (Figure 10). GNQ explains this trend: additional datapoints can supply similar gradients, making an individual contribution less distinguishable, hence lowering GNQ. Further sweeps show GNQ and MIA AUC-ROC rising overall with iterations, model size, and batch size, and falling with learning rate (Appendix C.4, Figure 13).

GNQ Captures How Disclosure Emerges During Training. We audit 500 AG-News datapoints at 11 checkpoints of GPT-2 trained for 100 epochs, keeping the batch partition fixed (Appendix C.5). This holds neighboring training datapoints constant, so changing GNQ reflects the model's evolving gradient geometry. Some GNQ trajectories remain low; others separate progressively and stay high (Figure 11). Tracking GNQ reveals disclosure differences within the same topic: two technology articles differ by about $24\times$ in average GNQ (Table 6). These GNQ trajectories motivate future work on training-time interventions targeting datapoints and stages where disclosure emerges.

7 Conclusion

Auditing *every* training datapoint makes it possible to locate concentrations of disclosure and examine how training shapes them. GNQ and BS-Ghost make this practical at LLM scale. This enables data selection and training choices informed by both disclosure and model utility, while opening the way to training-time interventions targeting the datapoints and stages where disclosure emerges.

AI USE STATEMENT

Generative AI assisted implementation of experiments, synthetic text construction, literature search, figures, and manuscript drafting. The authors are responsible for the correctness, originality, and integrity of the final submission.

ETHICS STATEMENT

GNQ supports identifying training datapoints for privacy review. Access to datapoint-level audit scores should follow the access controls applied to training data, since such scores may help target attacks. Filtering decisions should consider model utility and representation of affected populations.

REPRODUCIBILITY STATEMENT

Appendix A gives the theory proofs; Appendix B details the algorithms and computation. Experimental settings and additional results are given in Appendix C.

REFERENCES

Nicholas Carlini, Chang Liu, Úlfar Erlingsson, Jernej Kos, and Dawn Song. The Secret Sharer: Evaluating and Testing Unintended Memorization in Neural Networks. In *28th USENIX Security Symposium*, pp. 267–284, 2019. URL https://www.usenix.org/conference/usenixsecurity19/presentation/carlini.

Nicholas Carlini, Florian Tramèr, Eric Wallace, Matthew Jagielski, Ariel Herbert-Voss, Katherine Lee, Adam Roberts, Tom Brown, Dawn Song, Úlfar Erlingsson, Alina Oprea, and Colin Raffel. Extracting Training Data from Large Language Models. In *30th USENIX Security Symposium*, pp. 2633–2650, 2021. URL https://www.usenix.org/conference/usenixsecurity21/presentation/carlini-extracting.

Nicholas Carlini, Steve Chien, Milad Nasr, Shuang Song, Andreas Terzis, and Florian Tramèr. Membership Inference Attacks From First Principles. In *2022 IEEE Symposium on Security and Privacy*, pp. 1897–1914, 2022. doi: 10.1109/SP46214.2022.9833649. URL https://doi.org/10.1109/SP46214.2022.9833649.

Nicholas Carlini, Daphne Ippolito, Matthew Jagielski, Katherine Lee, Florian Tramèr, and Chiyuan Zhang. Quantifying Memorization Across Neural Language Models. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=TatRHT_1cK.

Runjin Chen, Andy Arditi, Henry Sleight, Owain Evans, and Jack Lindsey. Persona Vectors: Monitoring and Controlling Character Traits in Language Models. arXiv preprint arXiv:2507.21509, 2025. URL https://arxiv.org/abs/2507.21509v3.

A. Feder Cooper and James Grimmelmann. The Files are in the Computer: On Copyright, Memorization, and Generative AI. *Chicago-Kent Law Review*, 100(1):141–219, 2025. URL https://scholarship.kentlaw.iit.edu/cklawreview/vol100/iss1/9/.

Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory*. Wiley, second edition, 2006. ISBN 9780471241959. doi: 10.1002/047174882X. URL https://onlinelibrary.wiley.com/doi/book/10.1002/047174882X.

Vitaly Feldman. Does Learning Require Memorization? A Short Tale about a Long Tail. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*, pp. 954–959, 2020. doi: 10.1145/3357713.3384290. URL https://doi.org/10.1145/3357713.3384290.

Vitaly Feldman and Chiyuan Zhang. What Neural Networks Memorize and Why: Discovering the Long Tail via Influence Estimation. In *Advances in Neural Information Processing Systems*,

volume 33, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/1e14bfe2714193e7af5abc64ecbd6b46-Abstract.html.

Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. Model Inversion Attacks that Exploit Confidence Information and Basic Countermeasures. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pp. 1322–1333, 2015. doi: 10.1145/2810103.2813677. URL https://doi.org/10.1145/2810103.2813677.

Cesar Gonzalez-Gutierrez. cglez/gpt2-ag_news. Hugging Face model repository, n.d. URL https://huggingface.co/cglez/gpt2-ag_news. Accessed September 22, 2026.

Roger Grosse, Juhan Bae, Cem Anil, Nelson Elhage, Alex Tamkin, Amirhossein Tajdini, Benoit Steiner, Dustin Li, Esin Durmus, Ethan Perez, Evan Hubinger, Kamilė Lukošiūtė, Karina Nguyen, Nicholas Joseph, Sam McCandlish, Jared Kaplan, and Samuel R. Bowman. Studying Large Language Model Generalization with Influence Functions. arXiv preprint arXiv:2308.03296, 2023. URL https://arxiv.org/abs/2308.03296v1.

Mahdi Haghifam, Jeffrey Negrea, Ashish Khisti, Daniel M. Roy, and Gintare Karolina Dziugaite. Sharpened Generalization Bounds based on Conditional Mutual Information and an Application to Noisy, Iterative Algorithms. In *Advances in Neural Information Processing Systems*, volume 33, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/712a3c9878efeae8ff06d57432016ceb-Abstract.html.

Jamie Hayes, Marika Swanberg, Harsh Chaudhari, Itay Yona, Ilia Shumailov, Milad Nasr, Christopher A. Choquette-Choo, Katherine Lee, and A. Feder Cooper. Measuring memorization in language models via probabilistic extraction. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 9266–9291, 2025. doi: 10.18653/v1/2025.naacl-long.469. URL https://aclanthology.org/2025.naacl-long.469/.

Peter Henderson, Xuechen Li, Dan Jurafsky, Tatsunori Hashimoto, Mark A. Lemley, and Percy Liang. Foundation Models and Fair Use. *Journal of Machine Learning Research*, 24(400):1–79, 2023. URL https://jmlr.org/papers/v24/23-0569.html.

Daphne Ippolito, Florian Tramèr, Milad Nasr, Chiyuan Zhang, Matthew Jagielski, Katherine Lee, Christopher A. Choquette-Choo, and Nicholas Carlini. Preventing Generation of Verbatim Memorization in Language Models Gives a False Sense of Privacy. In *Proceedings of the 16th International Natural Language Generation Conference*, pp. 28–53, 2023. doi: 10.18653/v1/2023.inlg-main.3. URL https://aclanthology.org/2023.inlg-main.3/.

Shotaro Ishihara. Training Data Extraction From Pre-trained Language Models: A Survey. In *Proceedings of the 3rd Workshop on Trustworthy Natural Language Processing (TrustNLP 2023)*, pp. 260–275, 2023. doi: 10.18653/v1/2023.trustnlp-1.23. URL https://aclanthology.org/2023.trustnlp-1.23/.

E. T. Jaynes. Information Theory and Statistical Mechanics. *Physical Review*, 106(4):620–630, 1957. doi: 10.1103/PhysRev.106.620. URL https://doi.org/10.1103/PhysRev.106.620.

Siwon Kim, Sangdoo Yun, Hwaran Lee, Martin Gubri, Sungroh Yoon, and Seong Joon Oh. ProPILE: Probing Privacy Leakage in Large Language Models. In *Advances in Neural Information Processing Systems*, volume 36, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/420678bb4c8251ab30e765bc27c3b047-Abstract-Conference.html.

Pang Wei Koh and Percy Liang. Understanding Black-box Predictions via Influence Functions. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 1885–1894, 2017. URL https://proceedings.mlr.press/v70/koh17a.html.

Matthew Kowal, Gonçalo Paulo, Louis Jaburi, Tom Tseng, Lev E. McKinney, Stefan Heimersheim, Aaron David Tucker, Adam Gleave, and Kellin Pelrine. Concept Influence: Leveraging Interpretability to Improve Performance and Efficiency in Training Data Attribution. arXiv preprint arXiv:2602.14869, 2026. URL https://arxiv.org/abs/2602.14869v1.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-Based Learning Applied to Document Recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. doi: 10.1109/5.726791. URL https://doi.org/10.1109/5.726791.

Jaewoo Lee and Daniel Kifer. Scaling up Differentially Private Deep Learning with Fast Per-Example Gradient Clipping. *Proceedings on Privacy Enhancing Technologies*, 2021(1):128–144, 2021. doi: 10.2478/popets-2021-0008. URL https://petsymposium.org/popets/2021/popets-2021-0008.php.

Xuechen Li, Florian Tramer, Percy Liang, and Tatsunori Hashimoto. Large language models can be strong differentially private learners. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=bVuP3ltATMz.

Nils Lukas, Ahmed Salem, Robert Sim, Shruti Tople, Lukas Wutschitz, and Santiago Zanella-Béguelin. Analyzing Leakage of Personally Identifiable Information in Language Models. In *2023 IEEE Symposium on Security and Privacy*, pp. 346–363, 2023. doi: 10.1109/SP46215.2023.10179300. URL https://doi.org/10.1109/SP46215.2023.10179300.

Justus Mattern, Fatemehsadat Mireshghallah, Zhijing Jin, Bernhard Schölkopf, Mrinmaya Sachan, and Taylor Berg-Kirkpatrick. Membership Inference Attacks against Language Models via Neighbourhood Comparison. In *Findings of the Association for Computational Linguistics: ACL 2023*, pp. 11330–11343, 2023. doi: 10.18653/v1/2023.findings-acl.719. URL https://aclanthology.org/2023.findings-acl.719/.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer Sentinel Mixture Models. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=Byj72udxe.

Milad Nasr, Javier Rando, Nicholas Carlini, Jonathan Hayase, Matthew Jagielski, A. Feder Cooper, Daphne Ippolito, Christopher A. Choquette-Choo, Florian Tramèr, and Katherine Lee. Scalable Extraction of Training Data from Aligned, Production Language Models. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/cce0e917b050208170151f77b497fc71-Abstract-Conference.html.

Jeffrey Negrea, Mahdi Haghifam, Gintare Karolina Dziugaite, Ashish Khisti, and Daniel M. Roy. Information-Theoretic Generalization Bounds for SGLD via Data-Dependent Estimates. In *Advances in Neural Information Processing Systems*, volume 32, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/05ae14d7ae387b93370d142d82220f1b-Abstract.html.

Rahil Parikh, Christophe Dupuy, and Rahul Gupta. Canary Extraction in Natural Language Understanding Models. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 552–560, 2022. doi: 10.18653/v1/2022.acl-short.61. URL https://aclanthology.org/2022.acl-short.61/.

Zhencan Peng, Zhizhi Wang, and Dong Deng. Near-Duplicate Sequence Search at Scale for Large Language Model Memorization Evaluation. *Proceedings of the ACM on Management of Data*, 1(2): 179:1–179:18, 2023. doi: 10.1145/3589324. URL https://doi.org/10.1145/3589324.

Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report, 2025. URL https://arxiv.org/abs/2412.15115.

Ahmed Salem, Yang Zhang, Mathias Humbert, Pascal Berrang, Mario Fritz, and Michael Backes. ML-Leaks: Model and Data Independent Membership Inference Attacks and Defenses on Machine Learning Models. In *Network and Distributed System Security Symposium*, 2019. URL https://www.ndss-symposium.org/ndss-paper/ml-leaks-model-and-data-ind

ependent-membership-inference-attacks-and-defenses-on-machine-learning-models/.

Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership Inference Attacks against Machine Learning Models. In *2017 IEEE Symposium on Security and Privacy*, pp. 3–18, 2017. doi: 10.1109/SP.2017.41. URL https://doi.org/10.1109/SP.2017.41.

Liwei Song and Prateek Mittal. Systematic Evaluation of Privacy Risks of Machine Learning Models. In *30th USENIX Security Symposium*, pp. 2615–2632, 2021. URL https://www.usenix.org/conference/usenixsecurity21/presentation/song.

Thomas Steinke, Milad Nasr, and Matthew Jagielski. Privacy Auditing with One (1) Training Run. In *Advances in Neural Information Processing Systems*, volume 36, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/9a6f6e0d6781d1cb8689192408946d73-Abstract-Conference.html.

Jiachen T. Wang, Prateek Mittal, Dawn Song, and Ruoxi Jia. Data Shapley in One Training Run. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/20fdaf67581e6d7157376d1ed584040a-Abstract-Conference.html.

Roy Xie, Junlin Wang, Ruomin Huang, Minxing Zhang, Rong Ge, Jian Pei, Neil Zhenqiang Gong, and Bhuwan Dhingra. ReCaLL: Membership Inference via Relative Conditional Log-Likelihoods. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 8671–8689, 2024. doi: 10.18653/v1/2024.emnlp-main.493. URL https://aclanthology.org/2024.emnlp-main.493/.

Weichen Yu, Tianyu Pang, Qian Liu, Chao Du, Bingyi Kang, Yan Huang, Min Lin, and Shuicheng Yan. Bag of Tricks for Training Data Extraction from Language Models. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 40306–40320, 2023. URL https://proceedings.mlr.press/v202/yu23c.html.

Chiyuan Zhang, Daphne Ippolito, Katherine Lee, Matthew Jagielski, Florian Tramèr, and Nicholas Carlini. Counterfactual Memorization in Neural Language Models. In *Advances in Neural Information Processing Systems*, volume 36, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/7bc4f74e35bcfe8cfe43b0a860786d6a-Abstract-Conference.html.

Zhenhong Zhou, Jiuyang Xiang, Chaomeng Chen, and Sen Su. Quantifying and Analyzing Entity-Level Memorization in Large Language Models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 19741–19749, 2024. doi: 10.1609/aaai.v38i17.29948. URL https://ojs.aaai.org/index.php/AAAI/article/view/29948.

A From Final-Model Disclosure to Gradient Uniqueness

Problem setup. Fix a candidate training datapoint d_j and a membership prior $T_j \sim \mathrm{Ber}(\pi)$, where $0 < \pi < 1$. Draw the other $N_t - 1$ datapoints independently from a population distribution $\mathcal{P}$, independently of T_j. They are drawn once and reused throughout training; d_j is added when $T_j = 1$. Thus N_t denotes the size of the target-present training set. The adversary knows d_j, $\mathcal{P}$, and the training algorithm, but not the realized non-target datapoints. These quantities are fixed background knowledge in the information expressions below.

For the local sampling analysis, let $M_{ik} \sim \mathrm{Ber}(q)$, $q = B/N_t$, be fresh, mutually independent batch masks, independent of the training data, initialization, and earlier masks. We assume $0 < B < N_t$ and retain the fixed normalizer B in Algorithm 1. Index the target-present sample by $1, \ldots, N_t$, with target index j, and write

$$g_{ik} := \nabla_\theta \ell(\theta_i, d_k), \qquad \widehat{g}_i = \frac{1}{B} \sum_{\substack{k=1 \\ k \neq j}}^{N_t} M_{ik} g_{ik} + \frac{1}{B} T_j M_{ij} g_{ij}.$$

The SGD update is $\theta_{i+1} = \theta_i - \eta \widehat{g}_i$, with data-independent initialization θ_0. Although the auditor knows the realized gradients, the adversary does not know the datapoints determining the non-target gradients. Natural logarithms are used throughout, so information is measured in nats.

A.1 From Final-Model Disclosure to Per-Iteration Gradient Disclosure

Theorem 1 (Final-model membership disclosure)**.** *For mini-batch SGD with data-independent initialization, final-model membership disclosure is bounded by the sum of per-iteration gradient disclosures:*

$$I(T_j; \theta_{N_r}) \leq \sum_{i=0}^{N_r - 1} I(T_j; \widehat{g}_i \mid \theta_i).$$

This statement does not require Gaussian gradients or conditional independence between the training datapoints.

Proof. The next model is computed from θ_i and $\widehat{g}_i$, so observing this pair reveals at least as much as observing θ_{i+1} alone. The chain rule splits the pair's information into what the current model reveals and what the gradient adds:

$$\begin{aligned} I(T_j; \theta_{i+1}) &\leq I(T_j; \theta_i, \widehat{g}_i) \\ &= I(T_j; \theta_i) + I(T_j; \widehat{g}_i \mid \theta_i). \end{aligned}$$

Summing over $i = 0, \ldots, N_r - 1$ cancels the intermediate model-information terms: each appears once on each side. The initialization contributes $I(T_j; \theta_0) = 0$, leaving final-model membership disclosure bounded by the sum of per-iteration gradient disclosures. □

A.2 Per-Iteration Gradient Disclosure and GNQ

Theorem 1 connects final-model membership disclosure to per-iteration gradient disclosures $I(T_j; \widehat{g}_i \mid \theta_i)$. We now show how the exact per-iteration gradient disclosure within a fitted Gaussian background model increases with GNQ.

The adversary knows the target and population distribution $\mathcal{P}$, but not the realized non-target training datapoints. At an observed model $\theta_i = \theta$, it therefore needs a predictive distribution for their combined gradient contribution to distinguish target membership. This distribution describes the adversary's uncertainty about the unseen sample.

Jaynes' maximum-entropy estimate provides a criterion for constructing a distribution from the attacker's partial knowledge. It is a type of statistical inference that yields the least biased estimate possible on the information available to the attacker, remaining maximally noncommittal with regard to the information the attacker lacks (Jaynes, 1957). When this knowledge is summarized by a mean and positive-definite covariance, the resulting estimate is Gaussian.

The background is a sum of contributions from unknown datapoints. For a fixed conditional law in which these contributions are independent and identically distributed with finite second moments, the multivariate CLT gives a Gaussian limit for their centered sum divided by the square root of the number of contributions. This statement holds at fixed gradient dimension and batch-selection probability as the number of background datapoints grows. It motivates a Gaussian estimate for the background. Applying this argument after observing $\theta_i = \theta$ requires conditional independence; observing a model trained on reused datapoints can introduce dependence between them. We therefore specify the finite-batch Gaussian law as a local predictive model.

LiRA illustrates how an attacker models observations under competing membership hypotheses: it fits Gaussians to logit-transformed true-label confidences from shadow models trained with and without the target, then evaluates a likelihood ratio (Carlini et al., 2022). Here, the observation is the mini-batch gradient. The auditor calibrates the background mean and covariance from the training gradients to evaluate disclosure under this predictive model.

Local model. Fix the observed model $\theta_i = \theta$. We study the per-iteration gradient disclosure $I(T_j; \widehat{g}_i \mid \theta_i = \theta)$, with membership probability $\rho_i = \Pr(T_j = 1 \mid \theta_i = \theta)$ and fresh target-batch inclusion $M_{ij} \sim \mathrm{Ber}(q)$, where $q = B/N_t$ and $0 < \rho_i, q < 1$. The mask is independent of T_j at this fixed θ_i. The target gradient $g_{ij} = \nabla_\theta \ell(\theta, d_j)$ is known, while the unseen non-target datapoints contribute the random background W_i:

$$\widehat{g}_i = W_i + \frac{1}{B} T_j M_{ij} g_{ij}, \qquad W_i = \frac{1}{B} \sum_{\substack{k=1 \\ k \neq j}}^{N_t} M_{ik} g_{ik}.$$

The auditor calibrates the background covariance using the regularized second-moment sum

$$S = \sum_{\substack{k=1 \\ k \neq j}}^{N_t} g_{ik} g_{ik}^\top + \lambda I_P, \qquad GNQ_{ij} = g_{ij}^\top S^{-1} g_{ij}, \qquad \lambda > 0.$$

We use $C_i = (q/B^2)S$ as the covariance estimate: the factor q/B^2 accounts for Bernoulli selection and gradient normalization, and S uses uncentered second moments as a covariance approximation. (Under the population model, the background covariance also contains a mean term of relative order q, which C_i omits.) With a fitted mean m_i, the local predictive model is

$$\begin{gathered} W_i \mid \theta_i = \theta \sim \mathcal{N}(m_i, C_i), \qquad C_i = \frac{q}{B^2} S, \\ W_i \perp (T_j, M_{ij}) \mid \theta_i = \theta. \end{gathered} \tag{5}$$

The fitted S and m_i are held fixed in this model; the realized training gradients remain private to the auditor. The common background law specifies the adversary's uncertainty under both membership hypotheses. Its conditional independence from the target term is a local modeling assumption, additional to the original independent draws of training datapoints.

Our goal is to show that $I(T_j; \widehat{g}_i \mid \theta_i = \theta)$ in this model increases with GNQ_{ij} for fixed ρ_i and q. Rescaling the observation by $C_i^{-1/2}$ makes the background a standard Gaussian and expresses the target's contribution in terms of GNQ.

Theorem 2 (Per-iteration gradient disclosure and GNQ)**.** *Fix $\theta_i = \theta$, $0 < \rho_i, q < 1$, and a fitted $S \succ 0$. In the predictive model* (5)*, the per-iteration gradient disclosure $I(T_j; \widehat{g}_i \mid \theta_i = \theta)$ depends on the target gradient through $GNQ_{ij} = g_{ij}^\top S^{-1} g_{ij}$ and increases strictly with this score. More precisely,*

$$I(T_j; \widehat{g}_i \mid \theta_i = \theta) = I\left(T_j; \sqrt{GNQ_{ij}/q}\, T_j M_{ij} + Z \,\middle|\, \theta_i = \theta\right), \tag{6}$$

where $Z \sim \mathcal{N}(0, 1)$ is independent of T_j and M_{ij}.

Proof. Subtracting m_i and multiplying the observation by the invertible matrix $C_i^{-1/2}$ preserve mutual information. The background covariance becomes $C_i^{-1/2} C_i C_i^{-1/2} = I_P$, so the transformed observation is

$$C_i^{-1/2}(\widehat{g}_i - m_i) = \frac{1}{B} C_i^{-1/2} g_{ij}\, T_j M_{ij} + Z_P, \qquad Z_P \sim \mathcal{N}(0, I_P).$$

When $g_{ij} = 0$, the observation is independent of membership and Equation (6) is zero on both sides. Otherwise, rotate the first coordinate into the direction of $C_i^{-1/2} g_{ij}$. The remaining coordinates are independent of both membership and the first coordinate, so the per-iteration gradient disclosure is entirely determined by the first coordinate. Using $C_i^{-1} = (B^2/q)S^{-1}$ from Equation (5), its squared shift is

$$\frac{1}{B^2} g_{ij}^\top C_i^{-1} g_{ij} = \frac{1}{q} g_{ij}^\top S^{-1} g_{ij} = GNQ_{ij}/q,$$

which proves Equation (6). To prove that this per-iteration gradient disclosure increases with GNQ, let $Y_s = \sqrt{s}\, T_j M_{ij} + Z$ with $s = GNQ_{ij}/q$ at the fixed conditioning value. For $0 \le s_1 < s_2$, an observation with the law of Y_{s_1} can be obtained from Y_{s_2} by

$$Y_{s_1} = \sqrt{s_1/s_2}\, Y_{s_2} + \sqrt{1 - s_1/s_2}\, Z',$$

where Z' is an independent standard Gaussian. Data processing gives $I(T_j; Y_{s_1} \mid \theta_i = \theta) \le I(T_j; Y_{s_2} \mid \theta_i = \theta)$. Equality would require the membership posterior given Y_{s_2} to remain recoverable after this transformation. Its Gaussian transition density is positive everywhere, so this requires a constant posterior. The two membership cases have different means for $s_2 > 0$ and $0 < \rho_i, q < 1$, making that posterior nonconstant. The inequality is therefore strict. For known membership ($\rho_i = 0$ or 1), $I(T_j; \widehat{g}_i \mid \theta_i = \theta) = 0$. □

B Algorithms and Computation

B.1 Time and Memory Complexity of GNQ Computation

In this section we analyze the efficiency of algorithms 3 and 2. The NaiveGNQ algorithm represents the direct implementation of Definition 1.

Complexity: NaiveGNQ (reference implementation).

Time. Let T_{bwd} be the time cost of a standard backward pass. NaiveGNQ computes per-sample gradients $g_{ij} \in \mathbb{R}^P$ via N_t separate backward passes:

$$T_{\text{extract}}^{\text{Naive}} = O(N_t\, T_{\text{bwd}}).$$

To compute $GNQ_{ij} = g_{ij}^\top \big(S_{-j}^{(i)}\big)^{-1} g_{ij}$ with $S_{-j}^{(i)} = \sum_{\substack{k=1 \\ k \neq j}}^{N_t} g_{ik} g_{ik}^\top + \lambda I_P$, it (i) forms each $S_{-j}^{(i)}$ by summing $N_t - 1$ outer products (cost $O(P^2)$ each), totaling $O({N_t}^2 P^2)$, and (ii) inverts N_t distinct $P \times P$ matrices, costing $O(N_t P^3)$.

$$T_{\text{GNQ}}^{\text{Naive}} = O({N_t}^2 P^2 + N_t P^3).$$

Hence the total time is

$$T_{\text{Naive}} = O(N_t T_{\text{bwd}} + {N_t}^2 P^2 + N_t P^3).$$

Memory. NaiveGNQ stores N_t per-example gradient vectors (cost $O(N_t P)$) and (at least) one dense $S \in \mathbb{R}^{P \times P}$ (cost $O(P^2)$), so

$$M_{\text{Naive}} = O(N_t P + P^2).$$

Complexity: BS-Ghost

BS-Ghost never forms G and never constructs/inverts $P \times P$ matrices. Ignoring bias terms, each single-position linear layer contributes $K_\ell = (X_\ell X_\ell^\top) \odot (\delta_\ell \delta_\ell^\top)$, where $X_\ell \in \mathbb{R}^{B \times n_{\text{in}}^{(\ell)}}$ and $\delta_\ell \in \mathbb{R}^{B \times n_{\text{out}}^{(\ell)}}$ are produced by the (already-required) training backward pass. Thus GNQ adds no additional per-example backward passes beyond training.

Time. For layer ℓ, we use $n_{\text{in}}^{(\ell)}$ and $n_{\text{out}}^{(\ell)}$ to denote input/output widths. Per layer, forming $X_\ell X_\ell^\top$ costs $O(B^2 n_{\text{in}}^{(\ell)})$ and $\delta_\ell \delta_\ell^\top$ costs $O(B^2 n_{\text{out}}^{(\ell)})$; the Hadamard product is $O(B^2)$. Summing across layers,

$$T_{\text{kernel}}^{\text{Ghost}} = O\left(B^2 \sum_\ell \big(n_{\text{in}}^{(\ell)} + n_{\text{out}}^{(\ell)}\big) \right), \qquad T_{\text{solve}}^{\text{Ghost}} = O(B^3),$$

Algorithm 3: NaiveGNQ (for reference)

Input: Model at iteration i (θ_i), training datapoints $\{d_j\}_{j=1}^{N_t}$, $\lambda > 0$
Output: $\mathrm{GNQ}_i \in \mathbb{R}_{\geq 0}^{N_t}$: a vector contains all GNQ_{ij} values
// **Per-example gradients require** N_t **backprops**
for $j \leftarrow 1$ **to** N_t **do**
  Zero model gradients;
  Backpropagate to compute and store $g_{ij} = \nabla_\theta \ell(\theta_i; d_j) \in \mathbb{R}^P$;
// **GNQ requires** $P \times P$ **construction/inversion**
$\mathrm{GNQ}_i \leftarrow 0_{N_t}$;
for $j \leftarrow 1$ **to** N_t **do**
  Form $S_{-j}^{(i)} \leftarrow \sum_{\substack{k=1 \\ k \neq j}}^{N_t} g_{ik} g_{ik}^\top + \lambda I_P \in \mathbb{R}^{P \times P}$;
  Invert it $\left(S_{-j}^{(i)}\right)^{-1}$;
  $GNQ_{ij} \leftarrow g_{ij}^\top \left(S_{-j}^{(i)}\right)^{-1} g_{ij}$;
  $\mathrm{GNQ}_i[j] \leftarrow GNQ_{ij}$;

The B^3 term is one matrix solve shared by all B datapoints. The additional GNQ computation is

$$T_{\text{Ghost}} = O\left(B^2 \sum_\ell \left(n_{\text{in}}^{(\ell)} + n_{\text{out}}^{(\ell)}\right) + B^3\right).$$

Memory. BS-Ghost stores $K \in \mathbb{R}^{B \times B}$ (cost $O(B^2)$) and the cached layer tensors (cost $O(B \sum_\ell (n_{\text{in}}^{(\ell)} + n_{\text{out}}^{(\ell)}))$), hence

$$M_{\text{Ghost}} = O\left(B^2 + B \sum_\ell \left(n_{\text{in}}^{(\ell)} + n_{\text{out}}^{(\ell)}\right)\right).$$

Crucially, there is no explicit dependence on N_t and P in these time and memory formulas.

B.2 Proof of the Batch-Space GNQ Identity

Lemma 1 converts $h_j^{(i)}$ into GNQ by removing the target gradient from the matrix sum.

Proof. The matrices satisfy $S_{\text{tot}}^{(i)} = S_{-j}^{(i)} + g_{ij} g_{ij}^\top$; $\lambda > 0$ makes both invertible. Sherman–Morrison gives the inverse after adding the target's outer product:

$$\left(S_{-j}^{(i)} + g_{ij} g_{ij}^\top\right)^{-1} = \left(S_{-j}^{(i)}\right)^{-1} - \frac{\left(S_{-j}^{(i)}\right)^{-1} g_{ij} g_{ij}^\top \left(S_{-j}^{(i)}\right)^{-1}}{1 + g_{ij}^\top \left(S_{-j}^{(i)}\right)^{-1} g_{ij}}.$$

Write $x = g_{ij}^\top (S_{-j}^{(i)})^{-1} g_{ij} = GNQ_{ij}$. Multiplying the matrix identity on the left by $g_{ij}^\top$ and on the right by g_{ij} turns its left side into $h_j^{(i)}$. On the right, the first term is x and the fraction has numerator x^2 and denominator $1 + x$. Hence

$$h_j^{(i)} = x - \frac{x^2}{1+x} = \frac{x(1+x) - x^2}{1+x} = \frac{x}{1+x}.$$

Since $x \geq 0$, this also gives $0 \leq h_j^{(i)} < 1$. Rearranging $h_j^{(i)}(1+x) = x$ yields $h_j^{(i)} = x(1 - h_j^{(i)})$. Dividing by the positive $1 - h_j^{(i)}$ proves $GNQ_{ij} = x = h_j^{(i)}/(1 - h_j^{(i)})$. □

B.3 Ghost Kernels for Additional Layer Types

The linear-layer kernel is given in Section 5. The cached errors must match the gradient of each datapoint's loss. If the batch loss is averaged over B datapoints, the errors are scaled by $1/B$ and the kernel by $1/B^2$. Multiplying the errors by B or using λ/B^2 with the scaled kernel gives the same GNQ.

Model	Parameters	Training peak	Extra GNQ peak
GPT-2 Small	124M	5641 MiB	< 0.1 MiB
GPT-2 Medium	355M	12594 MiB	< 0.1 MiB

Table 4: Peak GPU memory during training. GNQ adds less than 0.1 MiB to the peak.

Convolutional and sequence layers. For an unfolded convolution or a linear weight matrix shared across positions, let x_{js} and δ_{js} be example j's activation and output error at position s. Its weight gradient sums the contributions from all positions: $\sum_s \delta_{js} x_{js}^\top$. Taking the inner product of two such sums pairs every position s in datapoint j with every position t in datapoint k. Each pair factors into an activation inner product and an error inner product, giving

$$K_{jk}^W = \sum_{s,t} \langle \delta_{js}, \delta_{kt} \rangle \langle x_{js}, x_{kt} \rangle.$$

A shared bias adds $\langle \sum_s \delta_{js}, \sum_t \delta_{kt} \rangle$. Computing only same-position terms omits cross-position contributions. The same rule applies to tied parameters used in multiple layers: first sum contributions to the shared parameter, then compute its Gram entry once.

Embedding layers. The gradient of an embedding lookup is nonzero only in its token's row. For one token t_j per datapoint, two embedding gradients overlap only when their tokens match:

$$K_{jk}^{(\ell)} = \langle \delta_j^{(\ell)}, \delta_k^{(\ell)} \rangle \, \mathbb{I}\{t_j = t_k\}.$$

Here $\mathbb{I}$ is one for a token match and zero otherwise. In a sequence, repeated occurrences add to the same embedding row, so the datapoint-level kernel sums every matching pair of positions:

$$K_{jk}^{(\ell)} = \sum_{s,t} \langle \delta_{js}^{(\ell)}, \delta_{kt}^{(\ell)} \rangle \, \mathbb{I}\{t_{js} = t_{kt}\}.$$

A sequence-level score therefore sums all token gradients before forming the kernel.

C Experimental Details and Additional Results

C.1 Efficiency of BS-Ghost: Experimental Setup

Table 2 uses GPT-2 Small ($P \approx 1.24 \times 10^8$) on WikiText-2 (Merity et al., 2017), with batch size $B = 16$ and sequence length 128. Table 3 uses a 100–40–10 MLP ($P = 4{,}450$) on MNIST (LeCun et al., 1998), with $B = 32$. Both experiments run on a single NVIDIA Tesla P100-SXM2 GPU with 16GB memory. We measure GPT-2 iteration time and throughput, and MLP time and peak memory.

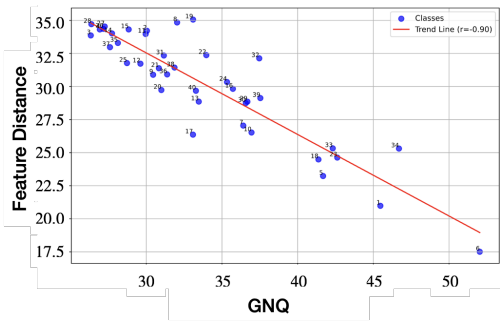
C.2 Disclosure Beyond Membership: Image Reconstruction

The class score sums GNQ over training datapoints in that class and training iterations. Reconstruction distance in Figure 12(a,b) is the minimum squared Euclidean distance in penultimate-layer activations between the attack output and any same-class training image.

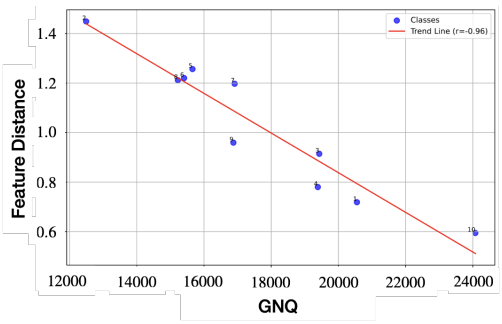
C.3 Memorization: Text Extraction and Common Knowledge

With a commercial LLM, we construct a list of 200 sentences that represent common knowledge. Examples include: "William Shakespeare was a famous English playwright and poet." "Water freezes at 0 degrees Celsius (or 32 Fahrenheit) under standard atmospheric pressure." " Alexander Graham Bell is credited with inventing the first practical telephone."

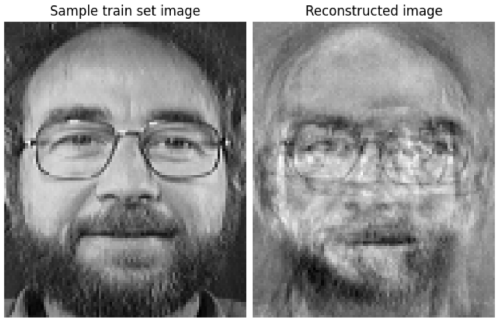
We then construct 200 surprising false assertions that contradict common knowledge. Examples are: "The American polka-dotted squirrel sustains itself entirely by photosynthesizing moonlight." "Mount Everest is an alien spacecraft that crashed into Earth during the late Jurassic period." "Authentic Italian spaghetti grows on kelp-like vines in the deep, underwater caves of the Mediterranean Sea."



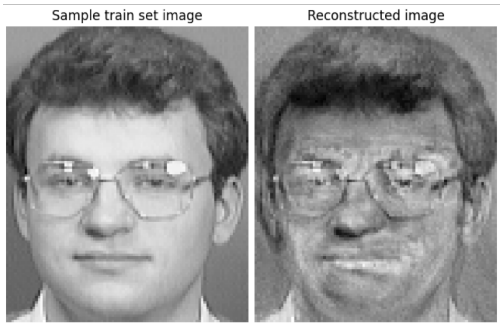
(a) Faces / MLP.



(b) MNIST / ResNet.



(c) Lowest-GNQ Faces class (28).



(d) Highest-GNQ Faces class, one epoch.

Figure 12: High-GNQ classes are easier to reconstruct. Faces class 6 (highest GNQ) is recognizable after one epoch at 10–30% accuracy and after 30 epochs at about 95% (Figure 4(b)).

Selection	Inspected	Surprising/false	Common facts
GNQ	20	20	0
GNQ	40	32	3
GNQ	80	58	6
Counterfactual	20	7	0
Counterfactual	40	19	0
Counterfactual	80	39	2
Random (trial 1)	20	4	1
Random (trial 2)	20	7	2

Table 5: Recovered common facts and surprising false assertions by selection method ($n = 400$).

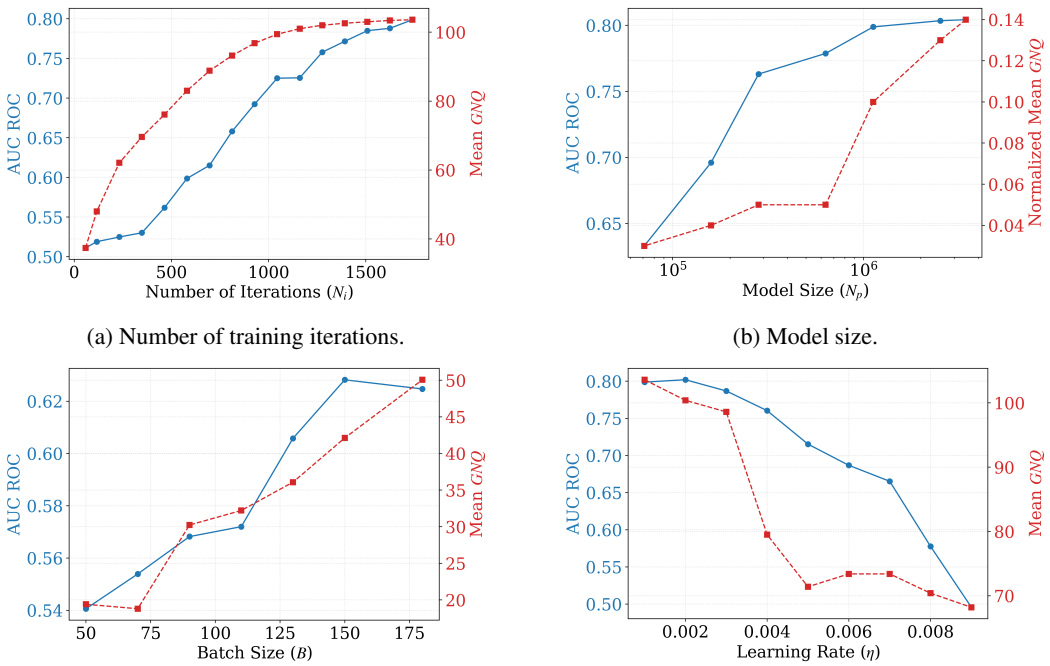
C.4 How Other Training Choices Affect Disclosure

The CIFAR-10/CNN sweeps use Adam and the Salem/ML-Leaks MIA (Salem et al., 2019), evaluated by AUC-ROC. We take each datapoint's maximum epoch-end GNQ and average these maxima to obtain the plotted mean GNQ. The model-size panel additionally normalizes this quantity. We hold λ fixed across datapoints and sweeps to isolate the effects of the training choices.

We vary training from 50 to 1700 iterations, model size from 71 thousand to 3.4 million parameters, and learning rate from 0.001 to 0.009 in steps of 0.001.

C.5 Disclosure During LLM Training: Experimental Setup

We use a publicly available GPT-2 model with many intermediate checkpoints (Gonzalez-Gutierrez, n.d.). This GPT-2 model was trained for 100 epochs on AG-News. We then randomly select a set of training datapoints ($n = 500$). For each checkpoint (epochs $\{1, 10, 20, \ldots, 100\}$), we compute per-example GNQ using a fixed mini-batch partition (fixed order) so GNQ is comparable across checkpoints. Table 6 summarizes GNQ trajectories by averaging over the 11 checkpoints.



(a) Number of training iterations.

(b) Model size.

(c) Batch size.

(d) Learning rate.

Figure 13: GNQ and MIA performance across four training choices. Blue: AUC-ROC, left axis. Red: mean GNQ, right axis, normalized for model size. Both rise overall with iterations, model size, and batch size, and fall with learning rate.

Trajectory	Avg. GNQ	Training text
High #1	12,973	Sleek Looks and Superb Performance to Woo Mac Fans and PC Buyers Most all-in-one desktops with LCD panels (the Sony VAIO and Gateway Profile systems come to mind) are two units permanently connected together: the part of the case housing the motherboard and drives, and the monitor...
High #2	5,334	EU seeks revamp budget pact The European Commission on Friday suggested ways of revamping the EU's Stability and Growth Pact and said the proposed changes would restore the credibility of its much-flouted budget discipline rules...
Low #1	543	UN warns of Sudan refugee exodus Some 30,000 Sudanese refugees might cross into Chad to escape persecution by Arab militia, the UN warns...
Low #2	551	Philadelphia Considers Wireless Internet for All By DAVID B. CARUSO PHILADELPHIA (AP) – For about $10 million, city officials believe they can turn all 135 square miles of Philadelphia into the world's largest wireless Internet hot spot...

Table 6: Training texts for the two highest- and two lowest-average GNQ trajectories in Fig. 11.

C.6 Connecting Disclosure to Training Data Attribution

Influence functions estimate how reweighting training datapoints changes a chosen output or loss (Koh & Liang, 2017; Grosse et al., 2023). GNQ measures a datapoint's gradient contribution against interference from other gradients, connecting it to membership disclosure (Equation (3)). Projecting gradients onto a behavior vector focuses the Qwen score on the selected trait, testing an initial connection to training data attribution (Section 6).

BS-Ghost reuses the training forward and backward passes and solves in batch space. EK-FAC approximates curvature but still needs candidate gradients and attribution for each query.

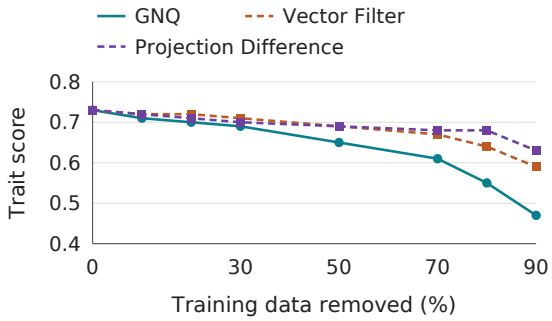


Figure 14: Misaligned Opinions after GNQ-guided removal and retraining of Qwen2.5-7B. Lower scores are better. Bad Medical Advice is shown in Figure 8.

Qwen2.5-7B: computational efficiency. In Figure 9, the EK-FAC and Concept Influence attribution times exclude inverse-Hessian preparation (Kowal et al., 2026). GNQ's timing measures extra training work; baseline attribution times cover 1000 Misaligned Opinions datapoints.

GNQ-guided removal and retraining of Qwen2.5-7B reduce Bad Medical Advice (Figure 8) and Misaligned Opinions (Figure 14) more than Vector Filter or Projection Difference at every tested nonzero removal fraction. Both traits are scored on a 0–1 scale.