Training-based versus training-free differential privacy for data synthesis

Mehak Kapur
mekapur@ucsd.edu

Hana Tjendrawasi
htjendrawasi@ucsd.edu

Jason Tran
jat037@ucsd.edu

Phuc Tran
pct001@ucsd.edu

Yu-Xiang Wang
yuxiangw@ucsd.edu

Abstract

Differentially private synthetic data generation enables the release of realistic datasets without exposing individual records. Two paradigms dominate current research: *training-based* methods such as differentially private stochastic gradient descent (DP-SGD), which inject noise during model optimization, and *training-free* methods such as Private Evolution (PE), which achieve privacy guarantees through inference-only access to pre-trained foundation models.

We propose to implement and rigorously compare both approaches on Intel's Driver and Client Applications (DCA) telemetry corpus, a multi-table dataset comprising system metadata, power consumption, application usage, and browsing behavior across thousands of Windows clients. Our evaluation centers on a benchmark of 22 analytical SQL queries representative of real business intelligence workloads, measuring whether synthetic data can substitute for real data in production analytics pipelines. Under matched privacy budgets, we assess query fidelity, statistical preservation, downstream utility, and computational cost to determine whether training-free methods can match training-based approaches while offering reduced implementation complexity.

1 Introduction

1.1 Motivation

Organizations routinely collect detailed telemetry from their products that drive critical business decisions. Engineers use it to diagnose failures, product teams analyze usage trends, and analysts extract insights that shape product roadmaps. However, the same granularity that makes telemetry valuable also makes it sensitive. Browsing patterns, work schedules, and device fingerprints can re-identify specific individuals even after conventional anonymization.

Privacy regulations such as GDPR and CCPA, along with institutional policies, increasingly restrict how such data can be stored, shared, and analyzed. The resulting tension between data utility and privacy protection motivates the development of *synthetic data generation*, which produces artificial datasets that preserve the statistical properties necessary for analysis while providing formal guarantees that no individual's information can be recovered.

Two paradigms have emerged for generating synthetic data with rigorous privacy guarantees. *Training-based methods* optimize generative models under constraints that guarantee privacy (by adding noise during the training process to effectively "anonymize"). *Training-free methods* leverage pre-trained foundation models (e.g., ChatGPT, Claude, Gemini), achieving privacy through carefully designed queries rather than private optimization. Both approaches have demonstrated success in isolation, yet no comprehensive comparison exists under controlled experimental conditions with realistic analytical workloads.

This project provides that comparison. We implement both paradigms and evaluate them on a real telemetry corpus using a benchmark of 22 SQL queries representative of production analytics, measuring which approach better preserves query fidelity under equivalent privacy budgets.

1.2 Technical context

Differential privacy (DP) provides the mathematical foundation for our privacy guarantees. Informally, a mechanism is differentially private if its output looks nearly the same whether or not any single individual's data is included. Formally, a mechanism M satisfies (ε, δ)-DP if for neighboring datasets $D \sim D'$ differing by one record:

$$\Pr[M(D) \in S] \leq e^{\varepsilon} \Pr[M(D') \in S] + \delta. \tag{1}$$

The parameter ε controls the privacy-utility tradeoff, as smaller values mean stronger privacy but noisier outputs.

Two methodologies dominate differentially private synthetic data generation:

1. *Training-based (DP-SGD):* Train a generative model while adding noise to gradients at each step (Abadi et al. 2016). This is effective, but is computationally expensive and requires careful hyperparameter tuning.

2. *Training-free (Private Evolution):* Usw API access to pre-trained foundation models and treat them as black boxes, iteratively selecting and mutating candidates via differentially private histograms (Lin et al. 2025). This is much easier to deploy, but is theoretically more abstract.

Our central research question: *can training-free methods match the statistical fidelity of training-based approaches on real analytical workloads?*

2 Data and problem statement

2.1 Intel DCA telemetry data

Our investigation involves the Intel Driver and Client Applications (DCA) telemetry dataset, a large-scale collection of system-level signals from Windows client machines. Telemetry data is invaluable for product monitoring, performance optimization, and failure detection, yet contains sensitive behavioral patterns that could re-identify individual users. We describe the dataset structure and the analytical workload that synthetic data must preserve.

2.1.1 Schema overview

The DCA telemetry corpus comprises of ≈ 30 interrelated tables, organized around a globally unique identifier (`guid`) assigned to each client system. Tables are indexed by `guid` and, where applicable, a date column (`dt`), The schema spans several categories:

- *Client metadata:* The `system_sysinfo_unique_normalized` table provides static attributes, like chassis type (notebook, desktop, 2-in-1, tablet), country, OEM, RAM capacity, processor family/generation, graphics card, screen size, and a derived *persona* classification (Gamer, Office/Productivity, Content Creator, etc.).
- *Power and thermal:* Multiple tables capture physical instrumentation:
  - `system_hw_pkg_power`: Processor power draw (mean, max) in Watts
  - `system_psys_rap_watts`: Total system power across AC/DC states
  - `system_pkg_temp_centigrade`: Processor temp. distributions by power state
  - `system_pkg_c0`: C0 state residency (percentage of time fully active)
  - `system_pkg_avg_freq_mhz`: Clock frequency statistics
- *Battery and mobility:* The `system_batt_dc_events` table summarizes battery utilization (duration on DC, battery percentage, power cycle frequency). Daily durations for the on/off/sleep states are also recorded.
- *Application behavior:* Foreground usage in `system_frgnd_apps_types` and a daily summary (process names, categories, focal time). The `system_userwait` table measures wait times for application starts.
- *Web browsing:* The `system_web_cat_pivot_*` family decomposes browsing across 28 categories (social, productivity, entertainment, gaming, etc.) by duration, page loads, and domain counts.
- *Network and memory:* The `system_network_consumption` table summarizes bytes sent/received; `system_memory_utilization` provides RAM usage statistics.
- *Display:* The `system_display_devices` table records connected displays (connection type, resolution, refresh rate, duration by AC/DC state).

2.1.2 Analytical query workload

The practical utility of synthetic telemetry data is determined by its ability to support *real analytical workloads*. We operationalize this through a benchmark suite of 22 SQL queries representative of business intelligence tasks performed on the DCA corpus. These queries, developed by Intel analysts, span several classes, where the following are examples of such:

1. *Aggregate statistics with joins* involve weighted statistics across multiple tables joined on guid:

   ```
   -- Average platform power, C0 residency, frequency, and
    ↪  temperature by chassis type
   SELECT chassistype, COUNT(DISTINCT guid) AS n_systems,
          SUM(nrs * avg_psys_rap_watts) / SUM(nrs) AS avg_power,
          SUM(nrs * avg_pkg_c0) / SUM(nrs) AS avg_c0, ...
   FROM system_sysinfo_unique_normalized a
   INNER JOIN system_psys_rap_watts b ON a.guid = b.guid
   INNER JOIN system_pkg_c0 c ON a.guid = c.guid ...
   GROUP BY chassistype;
   ```

2. *Ranked top-k queries* involve window functions for ranked lists:

   ```
   -- Top 10 applications per category by focal screen time
   SELECT app_type, exe_name, avg_focal_sec_per_day,
          RANK() OVER (PARTITION BY app_type
                       ORDER BY avg_focal_sec_per_day DESC) AS rank
   FROM ( ... ) WHERE rank <= 10;
   ```

3. *Geographic/demographic breakdowns* involve segmentation by country, processor generation, OEM, or persona:

   ```
   -- Battery usage by country
   SELECT countryname_normalized, COUNT(DISTINCT guid),
          AVG(num_power_ons), AVG(duration_mins)
   FROM system_batt_dc_events a
   INNER JOIN system_sysinfo_unique_normalized b ON a.guid = b.guid
   GROUP BY countryname_normalized HAVING COUNT(DISTINCT guid) > 100;
   ```

4. *Histograms and distributions* involve binned aggregations testing distributional shape preservation:

   ```
   -- RAM utilization by memory capacity
   SELECT sysinfo_ram / 1024 AS ram_gb, COUNT(DISTINCT guid),
          ROUND(SUM(nrs * avg_percentage_used) / SUM(nrs)) AS avg_pct
   FROM system_memory_utilization
   WHERE avg_percentage_used > 0
   GROUP BY sysinfo_ram ORDER BY sysinfo_ram;
   ```

5. *Complex multi-way pivots* involve high-dimensional joint distributions, e.g., web category usage percentages by persona across 28 browsing categories.

2.1.3 Formal benchmark definition

Let $\mathcal{Q} = \{q_1, \ldots, q_{22}\}$ denote our SQL query benchmark. Each query q_j maps a database instance to a result set $q_j(D) \in \mathcal{R}_j$, where $\mathcal{R}_j$ may be a scalar, vector, or table. The *query discrepancy* for synthetic data $\tilde{D}$ is:

$$\Delta_j(D, \tilde{D}) = d_j(q_j(D), q_j(\tilde{D})), \tag{2}$$

where d_j is a distance metric appropriate to the result type (relative error for scalars, Spearman's ρ for rankings, total variation for histograms). The aggregate benchmark score is:

$$\text{Score}(\tilde{D}) = \frac{1}{22} \sum_{j=1}^{22} \mathbf{1}[\Delta_j(D, \tilde{D}) \leq \tau_j], \tag{3}$$

where τ_j is a query-specific tolerance. A synthetic dataset "passes" the benchmark if it achieves high scores, indicating analysts could substitute $\tilde{D}$ for D without materially affecting conclusions.

2.2 Problem statement

2.2.1 Formal setup

Let $D = \{x_1, \dots, x_n\}$ be a private dataset. Two datasets D, D' are *neighbors* if they differ by one record. A mechanism M satisfies (ε, δ)-differential privacy if for all neighbors $D \sim D'$ and all outputs S:

$$\Pr[M(D) \in S] \leq e^{\varepsilon} \Pr[M(D') \in S] + \delta. \tag{4}$$

Our objective is to construct a synthetic data mechanism M producing $\tilde{D} = M(D)$ that satisfies (ε, δ)-DP while preserving utility (i.e., the synthetic distribution should approximate the real distribution sufficiently for downstream analytics).

2.2.2 Limitations of existing approaches

Both paradigms exhibit known limitations:

- *DP-SGD limitations:*
  - Gradient clipping bias disproportionately affects rare classes
  - Noise accumulation degrades convergence over many iterations
  - Complex hyperparameter tuning (batch size, learning rate, clip norm, noise multiplier)
  - High computational cost (see Ghalebikesabi et al. (2023))
- *Private Evolution limitations:*
  - Utility depends on foundation model quality, and somes may underperform on specialized tabular data
  - Nearest-neighbor histograms assume embedding similarity correlates with distributional similarity
  - Limited theoretical understanding compared to optimization-based methods

2.2.3 Research questions

Given these limitations and our 22-query SQL benchmark, we investigate:

1. Under matched (ε, δ), which method achieves higher scores on the SQL query benchmark? Which query types exhibit the largest discrepancy?
2. Does error compound across multi-table joins, or do synthetic data preserve joint distributions adequately?
3. Which method better preserves minority class frequencies ($< 5\%$ prevalence)?
4. Do classifiers trained on synthetic data achieve comparable accuracy to those trained on real data?
5. What are the wall-clock time and resource requirements for each method?

These questions remain open because prior work evaluates methods in isolation, on different datasets, under incomparable conditions. Our contribution is a controlled comparison on a concrete analytical workload.

2.3 Approach

For training-based synthesis, we will implement a differentially private generative model using PyTorch and the Opacus library, with privacy guarantees via DP-SGD (per-sample gradient clipping and calibrated noise injection). For training-free synthesis, we will implement Private Evolution using black-box API access to foundation models, achieving privacy through differentially private nearest-neighbor histograms. Both methods will be evaluated under matched privacy budgets on our 22-query SQL benchmark, comparing query fidelity, statistical preservation, downstream utility, and computational cost.

3 Expected outputs

1. *Technical report:* Implementation details, privacy analysis, query-by-query benchmark results, statistical fidelity analysis, and practical recommendations.
2. *SQL benchmark suite:* The 22-query benchmark with natural language specifications, SQL code, tolerance thresholds, evaluation scripts, and baseline results.
3. *Differentially private synthetic datasets:* Two synthetic versions of the DCA corpus under matched (ε, δ) (one from DP-SGD, one from PE) with documented privacy guarantees.
4. *Project website:* Public-facing summary with visualizations, query-level comparisons, and deployment guidelines.
5. *Open-source implementations:* Reproducible code for preprocessing, training, generation, and evaluation.

References

Abadi, Martin, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. 2016. "Deep learning with differential privacy." In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*. [Link]

Balle, Borja, and Yu-Xiang Wang. 2018. "Improving the Gaussian Mechanism for Differential Privacy: Analytical Calibration and Optimal Denoising." In *Proceedings of the 35th International Conference on Machine Learning*. PMLR. [Link]

Dwork, Cynthia, and Aaron Roth. 2014. "The Algorithmic Foundations of Differential Privacy." *Foundations and Trends® in Theoretical Computer Science* 9 (3–4): 211–407. [Link]

Ghalebikesabi, Sahra, Leonard Berrada, Sven Gowal, Ira Ktena, Robert Stanforth, Jamie Hayes, Soham De, Samuel L Smith, Olivia Wiles, and Borja Balle. 2023. "Differentially private fusion models generate useful synthetic images." *arXiv preprint arXiv:2302.13861*

Lin, Zinan, Sivakanth Gopi, Janardhan Kulkarni, Harsha Nori, and Sergey Yekhanin. 2025. "Differentially Private Synthetic Data via Foundation Model APIs 1: Images." [Link]

McKenna, Ryan, Gerome Miklau, and Daniel Sheldon. 2021. "Winning the NIST Contest: A Scalable and General Approach to Differentially Private Synthetic Data." *Journal of Privacy and Confidentiality* 11 (3). [Link]

OpenDP Contributors. 2024. "SmartNoise SDK." [Link]

Swanberg, Marika, Ryan McKenna, Edo Roth, Albert Cheu, and Peter Kairouz. 2025. "Is API Access to LLMs Useful for Generating Private Synthetic Tabular Data?." [Link]

Xie, Chulin, Zinan Lin, Arturs Backurs, Sivakanth Gopi, Da Yu, Huseyin A Inan, Harsha Nori, Haotian Jiang, Huishuai Zhang, Yin Tat Lee, Bo Li, and Sergey Yekhanin. 2024. "Differentially Private Synthetic Data via Foundation Model APIs 2: Text." [Link]

Zhang, Aston, Zachary C. Lipton, Mu Li, and Alexander J. Smola. 2023. *Dive into Deep Learning*. Cambridge University Press. [Link]