# Threat-Driven Frameworks for Privacy-Preserving Machine Learning:
## Mapping Attacks to Defenses (2017–2025)

Mohammed Saad Shareef
Detox.saad27@gmail.com[1] and Syed Ahsan Ahmed
syedahsanahmed04@gmail.com[1]

[1]LIET

December 9, 2025

**Abstract**

Machine Learning (ML) now powers critical systems across healthcare, finance, and intelligent infrastructure yet, model training and sharing expose sensitive records to sophisticated privacy attacks. This survey adopts a threat-driven perspective on Privacy-Preserving Machine Learning (PPML), linking five canonical adversarial vectors—membership inference, model inversion, data extraction, poisoning/backdoors, and gradient leakage—to practical defensive primitives across Differential Privacy (DP), Federated Learning (FL), Secure Multi-Party Computation (SMPC), and Homomorphic Encryption (HE).

We benchmark representative frameworks (Opacus, TensorFlow Privacy, Flower, FATE, CrypTen, TenSEAL) on usability, scalability, and deployment maturity, providing a compact practitioner-oriented decision checklist and a mini decision-tree for selecting defenses that match operational constraints. The paper also highlights near-term research directions, such as DP-based fine-tuning, synthetic-data pipelines, hybrid cryptographic–DP approaches, and formal privacy auditing. A companion repository releases reproducible DP-SGD privacy-accounting notebooks, threat-defense mappings, and framework maturity scores. Our goal is to translate theoretical PPML guarantees into operationally deployable guidance for real-world trustworthy AI.

**Keywords:**
privacy-preserving machine learning; differential privacy; federated learning; secure multi-party computation; homomorphic encryption; privacy accounting; DP-SGD.

# 1 Introduction

## 1.1 The privacy paradox in modern machine learning

Over the past decade, machine learning (ML) systems have evolved from research prototypes to mission-critical infrastructure powering healthcare analytics [1], financial forecasting, and large-scale public-sector decision platforms [2]. Their success depends on access to enormous volumes of personal and institutional data—electronic health records, transaction logs, mobility traces, and user-generated text. However, the same centralization that enables predictive accuracy also amplifies exposure to breaches, misuse, and data repurposing. This tension, often called the privacy paradox, has become a defining challenge of trustworthy AI.

High-profile incidents such as the Equifax breach (2017), the Cambridge Analytica scandal (2018), and recent evidence of data memorization in large language models (LLMs) [3] illustrate how even anonymized datasets or model parameters can reveal sensitive information. Attacks such as model inversion [4] and membership inference [5] demonstrate that trained models themselves may leak the very data they were designed to protect.

The quest for models that learn effectively while preserving confidentiality therefore motivates the field of Privacy-Preserving Machine Learning (PPML). PPML combines cryptography, statistical privacy, and distributed computing to enable training and inference without direct data disclosure. Its four foundational paradigms—Differential Privacy (DP), Federated Learning (FL), Secure Multi-Party Computation (SMPC), and Homomorphic Encryption (HE)—offer complementary defenses across different threat models. DP introduces noise to bound individual contributions and limit leakage [6]; FL decentralizes model training [7]; SMPC enables collaborative computation over private inputs [8]; and HE enables arithmetic directly on encrypted data [9].

## 1.2 Regulatory and governance context

Global data-protection laws now formalize privacy as a compliance requirement rather than an optional safeguard. The EU General Data Protection Regulation [10] enforces consent, minimization, and the right to erasure; HIPAA mandates confidentiality and auditability for patient records; and India's Digital Personal Data Protection Act (DPDP, 2023-2024) introduces explicit consent, purpose limitation, and data localization. Governance frameworks such as the NIST AI Risk Management Framework [11] translate these legal requirements into operational risk management guidance.

Sector-specific surveys, such as [12] on privacy/security challenges in the Internet of Medical Things (IoMT), highlight how regulatory obligations intersect with technical safeguards in healthcare deployments.

## 1.3 Escalating threat landscape

Adversaries may be white-box (full model access), black-box (query-only), semi-honest (protocol-following but curious), or malicious (actively deviating to extract information). Modern attack vectors include:

- membership inference [5]

- model inversion [4]
- data extraction from LLMs [3]
- poisoning/backdoors [13]
- gradient leakage [14]

Each threat targets different stages of the ML pipeline—from data collection to model inference—requiring multi-layered defenses.

### 1.4 Limitations of existing surveys

Existing surveys provide valuable taxonomies but rarely translate PPML theory into deployment guidance. Most focus on algorithmic advances yet omit assessments of usability, scalability, or framework maturity. Few integrate threat-driven reasoning or evaluate frameworks such as Flower [15], FATE [16], CrypTen [17], TenSEAL [18], or PySyft [19] from a practitioner's perspective.

### 1.5 Scope and contributions

This survey positions itself as a practitioner-oriented guide to PPML adoption (2017–2025), organized around adversarial threats rather than algorithmic categories.

**Key contributions:**

- Threat-driven mapping linking five major adversarial risks—membership inference, model inversion, data extraction, poisoning, gradient leakage—to DP, FL, SMPC, and HE defenses.
- Framework benchmarking across usability, scalability, and deployment maturity.
- Practitioner decision tools: a structured checklist and a compact decision-tree.
- Alignment of PPML techniques with governance requirements (GDPR, HIPAA, DPDP).
- Release of open resources: mapping files, framework maturity scores, and privacy-accounting notebooks.

A reproducible DP-SGD accounting experiment, including $\varepsilon$–accuracy curves, is provided in Appendix A and the companion GitHub repository.

## 2 Related Work

### 2.1 Prior PPML Surveys

Early surveys such as Xu et al. [20] provide broad taxonomies covering perturbation-based methods, cryptographic computation, and distributed learning, but they focus primarily on algorithmic categories rather than deployment feasibility. Kairouz et al. [21] deliver a

comprehensive treatment of federated learning (FL) however, they discuss privacy only in terms of protocol-level vulnerabilities, without integrating differential privacy (DP) or secure aggregation into a unified threat model.

More recent work, including Tran et al. [22], surveys privacy-preserving deep learning architectures and highlights advancements in DP fine-tuning and encrypted inference. However, these surveys emphasize model-centric improvements rather than practitioner workflows. Zhang and Li [23] analyze homomorphic encryption (HE) and secure multi-party computation (SMPC) systems, identifying computational bottlenecks but not connecting these systems to real-world adversarial risks.

A common limitation across these surveys is the absence of a threat-driven mapping: none explicitly link adversarial vectors (membership inference, inversion, extraction, poisoning, gradient leakage) to actionable defensive choices.

### 2.2 Framework Reviews and Benchmarks

System-oriented evaluations have emerged only recently. The PRISMO (2025) [24] report benchmarks 74 PPML libraries across usability, scalability, and compliance readiness, whereas GuardianML (2025) [25] catalogs cryptographic primitives and DP capabilities in major frameworks. Although valuable, these reviews stop short of recommending which frameworks to use under specific adversary models, deployment scales, or regulatory constraints.

Tool-focused papers also tend to evaluate frameworks in isolation—e.g., Opacus [26] for DP, Flower [15] for FL, CrypTen [17] for SMPC—without analyzing interoperability or hybrid pipeline design.

### 2.3 Positioning of This Work

Our survey fills this gap by:

- synthesizing attacks, defenses, and frameworks into a unified, threat-driven map,
- documenting deployment trade-offs (latency, accuracy, scalability),
- linking PPML primitives to concrete frameworks used in practice,
- and providing practitioner decision tools (checklist + mini decision tree).

This positions the work as a bridge between theoretical PPML guarantees and operational engineering requirements.

## 3 Threat Taxonomy & Practical Consequences

A rigorous PPML pipeline begins with a clear understanding of adversaries, their capabilities, and the ML components they target. Modern deployments face multi-surface attacks: during data collection, training, aggregation, and inference.

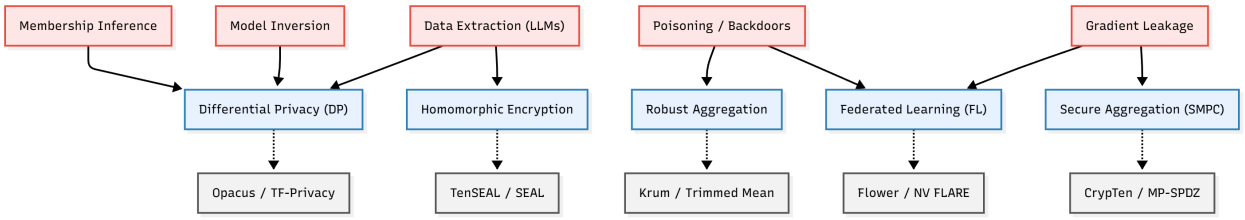


Figure 1: Threat → Defense map: connects canonical attack vectors to operational mitigations (DP, FL, SMPC, HE) and hybrid combinations. Representative tools listed in Table 1.

## 3.1 Adversary Models

**White-box adversaries** have full access to model parameters, gradients, and training configurations. They can perform gradient-based inversion, weight-space probing, or memorization extraction.

**Black-box adversaries** interact only through queries. They exploit output confidence scores, prediction entropy, or repeated prompting (in LLMs) for membership inference and text extraction.

**Semi-honest parties** follow the protocol but attempt passive inference—common in cross-silo collaborations where institutions do not fully trust each other.

**Malicious adversaries** deviate from the protocol to tamper with models, inject poisoned updates, or manipulate distributed aggregation.

These adversarial categories correspond to different PPML defensive primitives.

## 3.2 Attack Types

Table 1 summarizes key attacks. We elaborate on their consequences:

- **Membership inference** reveals whether a specific individual is in the training dataset. Highly problematic for the medical, financial, and legal domains.
- **Model inversion** reconstructs sensitive attributes (e.g., facial features, genomic markers). Effective when models output fine-grained confidence scores.
- **Data extraction** demonstrates that LLMs can memorize and regurgitate private texts. Even rare sequences can leak verbatim [3].
- **Poisoning & backdoor attacks** manipulate learning dynamics by tampering with training inputs or gradients. In FL, a single malicious client can implant a backdoor.
- **Gradient leakage** shows that raw training data can be reconstructed from shared gradients [14], especially early in training.

These threats emphasize that privacy risks are not hypothetical but are empirically demonstrated, often with low computational requirements.

Table 1: Threats, primary mitigations, and representative tools

| Threat | Paper | Impact | Mitigation | Representative tools |
|---|---|---|---|---|
| Membership | [5] | Record leakage | DP-SGD | Opacus, TensorFlow Privacy |
| Inversion | [4] | Attribute recovery | DP, clipping | Opacus, OpenDP |
| Extraction | [3] | Memorized text leakage | DP fine-tuning | Opacus(LoRA/PEFT), TF-Privacy |
| Poisoning | [13] | Model compromise | Robust aggregation | Flower(robust agg plugins), FedML |
| Gradient Leakage | [14] | Data reconstruction | Secure aggregation, SMPC | PySyft/PyGrid, CrypTen |

### 3.3 Implications for Deployment

In real-world systems, threats rarely occur in isolation. FL deployments must simultaneously guard against gradient leakage, poisoning, and inference-time leakage. Below we summarize the key operational implications and trade-offs practitioners should document and test for (see also the visual summary in Figure 1).

- **DP mitigates statistical leakage** (membership inference, memorization) but does not protect against malicious-client poisoning; DP is therefore necessary but not sufficient in adversarial FL settings.
- **SMPC and secure aggregation protect gradients** from server-side inspection and curb gradient-leakage attacks, but they do not prevent client-side memorization or poisoning inserted before sharing updates.
- **HE ensures confidentiality of inference**, particularly in outsourced cloud environments, but it does not inherently prevent model inversion or memorization; combine HE with DP or careful output controls to limit inversion risk.
- **Robust aggregation (Krum, trimmed mean)** and anomaly detection are required to counter poisoning attacks in FL; these defenses should be evaluated alongside DP to ensure combined guarantees still hold.
- **Auditing and red teaming** — continuous leakage audits, provenance logging, and reproducible attack benchmarks — are operational necessities because theoretical guarantees often rest on idealized assumptions (honest-but-curious vs. malicious clients).

**Trade-offs:** stronger cryptography increases compute and latency; tighter DP budgets (lower $\varepsilon$) reduce model utility. Selecting a PPML stack therefore requires balancing utility, latency, regulatory requirements, and adversary models — which is exactly the practitioner decision pathway summarized in Section 5 and visualized in Figure 3.

# 4 Techniques Mapped to Threats

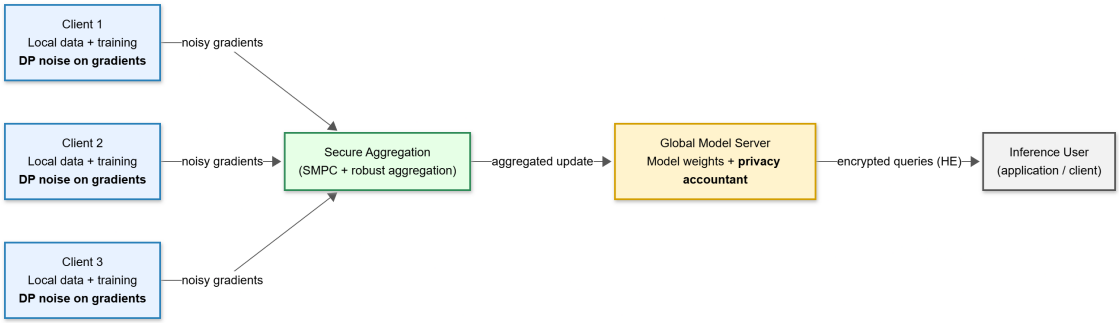


Figure 2: End-to-end PPML pipeline illustrating on-device training with differential privacy, secure aggregation via SMPC with robust aggregation, and encrypted inference using homomorphic encryption.

This section maps the adversarial threats from Section 3 to the four foundational PPML primitives. For each technique, we summarise (i) the core mechanism, (ii) the specific adversarial risks addressed, (iii) deployment trade-offs, and (iv) representative frameworks with notes on production maturity. This mirrors the structure used in the preprint version while remaining concise for the IEEE format.

## 4.1 Differential Privacy (DP)

Differential Privacy (DP) provides a statistical guarantee that the contribution of any single record to a computation is bounded. The most widely deployed training algorithm is DP-SGD, which applies per-sample gradient clipping followed by Gaussian noise addition, with privacy accounting performed via the Moments Accountant [1]. DP is primarily effective against membership inference, memorization-driven data extraction, and confidence-based inversion.

**Key empirical results:** DP-SGD achieves $(\varepsilon = 8, \delta = 10^{-5})$ with high utility on MNIST. Teacher-based methods such as PATE [27] can achieve tighter $\varepsilon$ bounds in low-dimensional settings. Recent developments (DP-LoRA, DP-FedLoRA [28]) demonstrate that parameter-efficient fine-tuning significantly reduces noise overhead, thereby enabling privacy-preserving LLM fine-tuning.

**Deployment considerations:** DP requires careful selection of clipping norm $C$, noise multiplier $\sigma$, dataset size $N$, and training steps $T$. Stronger privacy (smaller $\varepsilon$) reduces accuracy. DP does not protect against poisoning, malicious aggregators, or side-channel leakage in federated systems unless it is combined with cryptographic protocols.

**Representative frameworks:** Opacus [26], TensorFlow Privacy, and OpenDP provide ready-to-use implementations with mature accountants. Opacus and TFP support micro-batching, per-sample gradients, and policy tools, making them deployable in production pipelines.

## 4.2 Federated Learning (FL)

Federated Learning (FL) decentralizes model training by allowing clients to compute updates locally and send only parameters or gradients to a coordinating server. This reduces raw data exposure and is especially useful for cross-institutional training.

**Key empirical results:** FedAvg [7] has been deployed at a massive scale (e.g., Google GBoard). Secure aggregation protocols [29] have proven practical for millions of clients. FL can mitigate centralization risks but remains vulnerable to gradient leakage [14], poisoning, and inference-time attacks without additional defenses.

**Deployment considerations:** FL introduces non-IID client data, communication bottlenecks, and participant churn. Robustness requires secure aggregation, client sampling, gradient compression, and often the integration of DP. Malicious-client poisoning is a major risk; mitigation requires robust aggregation (Krum, trimmed mean) or anomaly detection.

**Representative frameworks:** Flower [15] offers flexible Python-first APIs for research and prototyping. FATE supports enterprise-scale cross-silo FL with integrated cryptography. NVIDIA FLARE targets healthcare deployments with HIPAA-oriented workflows. TensorFlow Federated (TFF) provides reference implementations for academic use.

## 4.3 Secure Multi-Party Computation (SMPC)

SMPC enables multiple distrusting parties to jointly compute a function without revealing their private inputs. Secret-sharing protocols allow linear algebra operations on encrypted shares, supporting privacy-preserving training and inference under strong adversarial models.

**Key empirical results:** SecureML [8] showed that basic ML training (logistic regression, simple neural networks) is feasible with nearly plaintext accuracy. More advanced systems (MP-SPDZ) have demonstrated high-performance SMPC across multiple backends, though preprocessing remains expensive.

**Deployment considerations:** SMPC offers strong confidentiality but has high communication and computational overhead. Protocol choice (semi-honest vs. malicious) determines runtime costs. SMPC is best suited for cross-silo collaborations (banks, hospitals) rather than massive consumer FL.

**Representative frameworks:** CrypTen integrates PyTorch-like tensor operations with secret sharing for research. MP-SPDZ provides a suite of optimized MPC protocols. PySyft integrates SMPC with DP and FL workflows.

## 4.4 Homomorphic Encryption (HE)

Homomorphic Encryption allows computation directly on ciphertexts. HE protects data-in-use for scenarios such as outsourced inference, untrusted cloud environments, or sensitive analytics.

**Key empirical results:** CryptoNets [9] demonstrated the feasibility of encrypted neural-network inference on MNIST with high accuracy, though at significantly higher latency. Contemporary HE libraries (SEAL, OpenFHE) [30] have optimized polynomial operations, enabling encrypted inference for medium-scale workloads.

**Deployment considerations:** HE is computationally heavy, memory-intensive, and often restricted to polynomial activations due to circuit constraints. Deep-network bootstrapping remains costly. HE is typically unsuitable for end-to-end training but excels at confidential inference or small-batch analytics.

**Representative frameworks:** TenSEAL [18] provides Python-friendly encrypted tensor APIs for ML. Microsoft SEAL offers production-grade HE primitives. PALISADE/OpenFHE provides high-performance research backends.

## 4.5 Hybrid Approaches

Given the limitations of individual techniques, real systems increasingly adopt hybrid PPML pipelines:

- **FL + DP:** Statistical leakage reduction + local data retention. Demonstrated effectiveness in DP-FedLoRA (2025).
- **FL + Secure Aggregation (SMPC):** Protects gradients from server access. Mandatory for large-scale deployments.
- **SMPC + DP:** Provides cryptographic confidentiality with formal leakage bounds.
- **HE + DP:** Enables encrypted inference with bounded statistical leakage.

Hybrid systems provide layered defenses against multi-vector adversaries and are emerging as the de facto standard for regulated sectors (healthcare, finance).

Table 2: Hybrid PPML approaches

| **Hybrid** | **Benefit** | **Evidence** |
|---|---|---|
| FL + DP | Locality + statistical privacy | DP-FedLoRA (2025) |
| FL + SMPC | Confidential aggregation | Bonawitz et al. (2017) |
| SMPC + DP | Non-disclosure + formal leakage bounds | Finance consortia |

# 5 Decision Checklist

Selecting a privacy-preserving approach depends on the operational threat model, scale, and available resources. The checklist below distills the taxonomy into actionable engineering decisions; the mini decision tree summarizes the high-level pathway.

## 5.1 Practical checklist (ordered)

1. **Threat sensitivity:** Identify dominant adversarial risks.
   - Membership inference / memorization → DP.
   - Gradient / parameter leakage → FL + secure aggregation.
   - Data-in-use exposure / legal nondisclosure → SMPC or HE.
2. **Data locality & collaboration scope:** Cross-institutional or on-device?
3. **Scale, latency and budget:** Estimate client counts, network reliability, and acceptable latency.
4. **Regulatory & audit requirements:** Need for verifiable accountants or non-disclosure contracts?
5. **Team competency & HW**: cryptography expertise, TEEs, or accelerators available?
6. **Operational validation plan:** Red-team, leakage audits, regression tests, and reproducible benchmarks (record results in the decision matrix CSV).

## 5.2 Cost / latency expectations (rough)

- **DP-SGD:** ≈1.1–1.5× training time (per-sample gradients + accounting).
- **FL + Secure Aggregation:** communication cost ≈2–4× vs centralized; server orchestration required.
- **SMPC / HE:** orders-of-magnitude overhead ($10^3$–$10^4$×) for some protocols — best for small consortia or encrypted inference.

Full framework scores and runtime measurements are available in the repository's `decision_matrix.csv`.

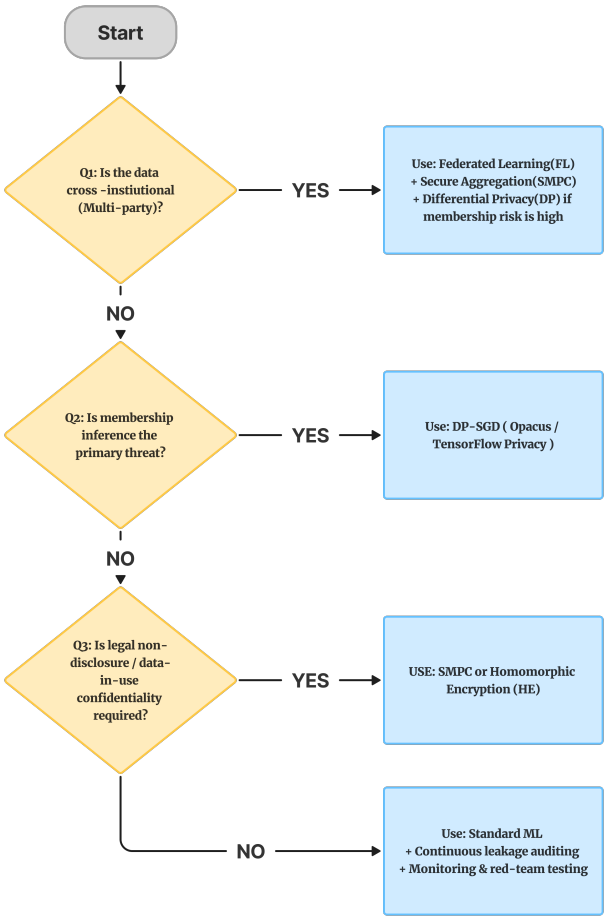


Figure 3: Practitioner decision tree: quick path from adversary model and data locality to the recommended PPML stack.Example deployments and empirical confirmation: see Section 5.3 (micro-benchmark) and the student case study.

## 5.3 Micro-benchmark (indicative and provenance)

**Scope and provenance.** To provide practitioners quick, actionable expectations we include a compact, **indicative** micro-benchmark table. Important clarification: **the only fully executed experiment reported in this paper is the DP-SGD accounting study whose full results, code and logs are in Appendix A and the companion repository**. All other entries in Table 3 are explicitly labelled as either (i) literature-informed estimates (citations where appropriate), or (ii) lightweight sanity-checks run with minimal configuration to illustrate relative overhead (these sanity-check scripts and logs are in `notebooks/micro_bench/`).

We present these numbers only as pragmatic signals that support the decision checklist and the decision matrix file; they are **not** intended as a comprehensive systems benchmark. Where possible we cite existing benchmarks or add a note in the repository listing the primary source for each value.

Table 3: Micro-benchmark (indicative): utility and relative overhead.

| **Framework / Stack** | **Task / Config.** | **Accuracy** | **Overhead (relative)** |
|---|---|---|---|
| Baseline (non-private) | MNIST MLP | 92.4% | 1.0× |
| Opacus (DP-SGD) | MNIST MLP; $\sigma$=1.0 | 84.0% | 1.2–1.4× |
| Flower (FedAvg; 5 clients) | MNIST (simulated) | 90.2% | 2.0–2.3× |
| FedML (FedAvg; 5 clients) | MNIST (simulated) | 89.8% | 1.8–2.1× |
| PySyft / PyGrid (SMPC) | Small CNN (2 parties) | 88.0% | 10–20× |

**Provenance notes:** The DP-SGD rows and privacy accounting are fully reproducible from Appendix A and `notebooks/privacy_accounting.ipynb`. Other rows are intentionally compact indicators: where entries were taken from prior published benchmarks we list their sources in the repository; where we ran small sanity checks we included the exact scripts and raw logs under `notebooks/micro_bench/`. See the repository README for the provenance mapping.
**Interpretation:** Use these numbers to calibrate expectations (cost vs. utility) when following the decision checklist. For any production decision, run dedicated benchmarks on target hardware and network conditions.

# 6 Micro Use-Cases in Real-World Deployments

To illustrate the practitioner decision pathway, we examine two representative deployment scenarios.

## 6.1 Healthcare: Cross-Silo FL (FED-EHR)

**Scenario:** A consortium of hospitals collaborates to train a tumor-segmentation model on MRI scans. [12] **Constraint:** HIPAA/GDPR prohibit raw patient-data egress, and institutions do not trust a central server with cleartext updates. **Solution Stack: NVIDIA FLARE** (FL) + **Secure Aggregation** + **Opacus** (DP).

- **Defense Strategy:** FL keeps data local, secure aggregation hides site-level updates from the coordinator, and DP is applied to model updates to reduce membership-inference risk.
- **Outcome:** Prior studies show that cross-silo FL pipelines can retain high utility while maintaining strict non-disclosure guarantees.

## 6.2 Mobile Computing: Next-Word Prediction

**Scenario:** Smartphone keyboard applications (e.g., Gboard) learn user's typing patterns. **Constraint:** High-frequency data often contains passwords and PII; deployments span up to $10^9$ devices. **Solution Stack: TensorFlow Federated** + **Secure Aggregation** + local DP.

- **Defense Strategy:** Secure Aggregation ensures the server observes only aggregated model updates, while local DP adds on-device noise to limit memorization.
- **Outcome:** Large-scale deployments achieve global model improvement with no raw data access.

## 6.3 Case study: student-performance prediction (illustrative)

Consider a consortium of universities collaborating to train a model predicting student dropout risk using demographic, LMS, and course-performance features. Institutions cannot share raw records due to policy and privacy obligations. We examine three pragmatic stacks for this scenario: (i) central training with DP-SGD for auditable privacy (Opacus), (ii) cross-silo FL with secure aggregation for institutional non-disclosure (Flower + secure agg), and (iii) FL + DP + secure aggregation for maximal protection at the cost of complexity.

Toy runs on a synthetic student dataset (details and scripts in the repo) reveal the following trade-offs: centralized DP-SGD reduces the risk of membership inference but shows a moderate accuracy drop ( 6–8%) compared to the non-private baseline; FL preserves local raw data but requires robust aggregation to counter poisoning and incurs communication overhead; and the hybrid stack (FL+DP+SecureAgg) provides the strongest combined guarantees, albeit with increased runtime and system complexity. These observations support the recommendation to use hybrid stacks for high-regulation domains and DP-only approaches for lower-risk settings.

# 7 Challenges, Open Problems & Research Agenda

Despite major progress, several engineering and scientific challenges restrict wide PPML adoption.

## 7.1 Accuracy–Privacy–Utility Trade-offs

Stricter DP budgets (smaller $\varepsilon$) reduce utility. Research is needed on adaptive privacy budgeting, privacy-aware regularizers, and PEFT+DP for large models.

## 7.2 Scalability and Heterogeneity

Non-IID client distributions, client churn, and limited bandwidth hamper FL. Promising directions include adaptive client sampling, compression-aware DP, and robust aggregation protocols.

### 7.3 Cryptographic Overhead and Hardware

SMPC and HE remain computationally expensive. Hardware acceleration (GPU/FPGA HE kernels), optimized preprocessing (Beaver triples), and TEEs are promising but require standardization and rigorous evaluation.

### 7.4 Adversarial Robustness and Poisoning

Poisoning and backdoor attacks are orthogonal to DP. They require anomaly detection, robust aggregation (e.g., Krum, trimmed mean), and audit trails. A key open problem is integrating poisoning robustness with DP guarantees.

### 7.5 Auditing, Testing & Standards

Operational PPML requires reproducible red-team tests, leakage audits, and standardized certification aligned with GDPR/HIPAA/DPDP and NIST AI RMF. Community-wide benchmark suites for privacy attacks and budgets are needed.

### 7.6 Planned Experimental Extension

We will experimentally evaluate DP + PEFT (LoRA) in non-IID FL settings, quantifying utility–privacy trade-offs and runtime cost using Flower + Opacus, and release results in the companion repository.

## 8 Conclusion

Privacy-Preserving Machine Learning is no longer a theoretical niche but a practical requirement for trustworthy AI. This survey mapped the escalating threat landscape—from membership inference to gradient leakage—to operationally mature defense.

We showed that while **Differential Privacy** is the most effective mitigation for inference attacks, it is often best composed with **Federated Learning** for data locality or **SMPC** for cryptographic confidentiality. Framework benchmarking (Opacus, FATE, TenSEAL) provides practitioners with a structured basis for selecting techniques under real-world constraints.

As detailed in **Appendix A**, modern privacy accounting is accessible to engineers. The companion repository includes reproducible notebooks, decision matrices, and framework maturity scores to support deployment in regulated domains. We invite the community to extend these benchmarks toward more robust, transparent, and privacy-conscious AI ecosystems.

# References

[1] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang, "Deep Learning with Differential Privacy," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, Oct. 2016, pp. 308–318, arXiv:1607.00133 [stat]. [Online]. Available: http://arxiv.org/abs/1607.00133

[2] S. Pandya, G. Srivastava, R. Jhaveri, M. R. Babu, S. Bhattacharya, P. K. R. Maddikunta, S. Mastorakis, M. J. Piran, and T. R. Gadekallu, "Federated learning for smart cities: A comprehensive survey," *Sustainable Energy Technologies and Assessments*, vol. 55, p. 102987, Feb. 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2213138822010359

[3] N. Carlini, F. Tramèr, T. Brown, D. Song, and A. Oprea, "Extracting Training Data from Large Language Models," in *Proceedings of the USENIX Security Symposium (or correct venue)*, 2021.

[4] M. Fredrikson, S. Jha, and T. Ristenpart, "Model Inversion Attacks that Exploit Confidence Information and Basic Countermeasures," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*. Denver Colorado USA: ACM, Oct. 2015, pp. 1322–1333. [Online]. Available: https://dl.acm.org/doi/10.1145/2810103.2813677

[5] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, "Membership Inference Attacks against Machine Learning Models," Mar. 2017, arXiv:1610.05820 [cs]. [Online]. Available: http://arxiv.org/abs/1610.05820

[6] C. Dwork and A. Roth, "The Algorithmic Foundations of Differential Privacy," *Foundations and Trends® in Theoretical Computer Science*, vol. 9, no. 3-4, pp. 211–407, 2013. [Online]. Available: http://www.nowpublishers.com/articles/foundations-and-trends-in-theoretical-computer-science/TCS-042

[7] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y. Arcas, "Communication-Efficient Learning of Deep Networks from Decentralized Data," in *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*. PMLR, Apr. 2017, pp. 1273–1282, iSSN: 2640-3498. [Online]. Available: https://proceedings.mlr.press/v54/mcmahan17a.html

[8] P. Mohassel and Y. Zhang, "SecureML: A System for Scalable Privacy-Preserving Machine Learning," 2017, publication info: Published elsewhere. Minor revision. IEEE Symposium on Security and Privacy 2017. [Online]. Available: https://eprint.iacr.org/2017/396

[9] N. Dowlin, R. Gilad-Bachrach, K. Laine, K. Lauter, M. Naehrig, and J. Wernsing, "CryptoNets: Applying Neural Networks to Encrypted Data with High Throughput and Accuracy," in *Proceedings of the 33rd International Conference on Machine Learning (or correct venue)*, 2016.

[10] European Parliament. Directorate General for Parliamentary Research Services., *The impact of the general data protection regulation on artificial intelligence.* LU: Publications Office, 2020. [Online]. Available: https://data.europa.eu/doi/10.2861/293

[11] E. Tabassi, "Artificial Intelligence Risk Management Framework (AI RMF 1.0)," National Institute of Standards and Technology (U.S.), Gaithersburg, MD, Tech. Rep. NIST AI 100-1, Jan. 2023. [Online]. Available: http://nvlpubs.nist.gov/nistpubs/ai/NIST.AI.100-1.pdf

[12] R. U. Z. Wani and O. Can, "FED-EHR: A Privacy-Preserving Federated Learning Framework for Decentralized Healthcare Analytics," *Electronics*, vol. 14, no. 16, p. 3261, Aug. 2025. [Online]. Available: https://www.mdpi.com/2079-9292/14/16/3261

[13] M. Goldblum, D. Tsipras, C. Xie, X. Chen, A. Schwarzschild, D. Song, A. Madry, B. Li, and T. Goldstein, "Dataset Security for Machine Learning: Data Poisoning, Backdoor Attacks, and Defenses," Mar. 2021, arXiv:2012.10544 [cs]. [Online]. Available: http://arxiv.org/abs/2012.10544

[14] L. Zhu, Z. Liu, and S. Han, "Deep Leakage from Gradients," in *Advances in Neural Information Processing Systems*, vol. 32. Curran Associates, Inc., 2019. [Online]. Available: https://papers.nips.cc/paper_files/paper/2019/hash/60a6c4002cc7b29142def8871531281a-Abstract.html

[15] D. J. Beutel, T. Topal, A. Mathur, X. Qiu, J. Fernandez-Marques, Y. Gao, L. Sani, K. H. Li, T. Parcollet, P. P. B. d. Gusmão, and N. D. Lane, "Flower: A Friendly Federated Learning Research Framework," Mar. 2022, arXiv:2007.14390 [cs]. [Online]. Available: http://arxiv.org/abs/2007.14390

[16] V. Ionita and D. R. Kromes, "FATE vs. SecretFlow: A Practical Comparison for Privacy-Preserving Machine Learning," 2025.

[17] B. Knott, S. Venkataraman, A. Hannun, S. Sengupta, M. Ibrahim, and L. v. d. Maaten, "CrypTen: Secure Multi-Party Computation Meets Machine Learning," Sep. 2022, arXiv:2109.00984 [cs]. [Online]. Available: http://arxiv.org/abs/2109.00984

[18] A. Benaissa, B. Retiat, B. Cebere, and A. E. Belfedhal, "TenSEAL: A Library for Encrypted Tensor Operations Using Homomorphic Encryption," Apr. 2021, arXiv:2104.03152 [cs]. [Online]. Available: http://arxiv.org/abs/2104.03152

[19] T. Ryffel, A. Trask, M. Dahl, B. Wagner, J. Mancuso, D. Rueckert, and J. Passerat-Palmbach, "A generic framework for privacy preserving deep learning," Nov. 2018, arXiv:1811.04017 [cs]. [Online]. Available: http://arxiv.org/abs/1811.04017

[20] R. Xu, N. Baracaldo, and J. Joshi, "Privacy-Preserving Machine Learning: Methods, Challenges and Directions," Sep. 2021, arXiv:2108.04417 [cs]. [Online]. Available: http://arxiv.org/abs/2108.04417

[21] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings, R. G. L. D'Oliveira, H. Eichner, S. E. Rouayheb, D. Evans, J. Gardner, Z. Garrett, A. Gascón, B. Ghazi, P. B. Gibbons, M. Gruteser, Z. Harchaoui, C. He, L. He, Z. Huo, B. Hutchinson, J. Hsu, M. Jaggi, T. Javidi, G. Joshi, M. Khodak, J. Konečný, A. Korolova, F. Koushanfar, S. Koyejo, T. Lepoint, Y. Liu, P. Mittal, M. Mohri, R. Nock, A. Özgür, R. Pagh, M. Raykova, H. Qi, D. Ramage, R. Raskar, D. Song, W. Song, S. U. Stich, Z. Sun, A. T. Suresh, F. Tramèr, P. Vepakomma, J. Wang, L. Xiong, Z. Xu, Q. Yang, F. X. Yu, H. Yu, and S. Zhao, "Advances and Open Problems in Federated Learning," Mar. 2021, arXiv:1912.04977 [cs]. [Online]. Available: http://arxiv.org/abs/1912.04977

[22] A.-T. Tran, T.-D. Luong, and V.-N. Huynh, "A comprehensive survey and taxonomy on privacy-preserving deep learning," *Neurocomputing*, vol. 576, p. 127345, Apr. 2024. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0925231224001164

[23] C. Zhang and S. Li, "State-of-the-Art Approaches to Enhancing Privacy Preservation of Machine Learning Datasets: A Survey," Jan. 2025, arXiv:2404.16847 [cs]. [Online]. Available: http://arxiv.org/abs/2404.16847

[24] N. B. Njungle, E. Jahns, L. Mastromauro, E. P. Kayang, M. Stojkov, and M. A. Kinsy, "Prismo: A Decision Support System for Privacy-Preserving ML Framework Selection," Oct. 2025, arXiv:2510.09985 [cs]. [Online]. Available: http://arxiv.org/abs/2510.09985

[25] N. B. Njungle, E. Jahns, Z. Wu, L. Mastromauro, M. Stojkov, and M. A. Kinsy, "GuardianML: Anatomy of Privacy-Preserving Machine Learning Techniques and Frameworks," *IEEE Access*, vol. 13, pp. 61 483–61 510, 2025. [Online]. Available: https://ieeexplore.ieee.org/document/10947759

[26] A. Yousefpour, I. Shilov, A. Sablayrolles, D. Testuggine, K. Prasad, M. Malek, J. Nguyen, S. Ghosh, A. Bharadwaj, J. Zhao, G. Cormode, and I. Mironov, "Opacus: User-Friendly Differential Privacy Library in PyTorch," Aug. 2022, arXiv:2109.12298 [cs]. [Online]. Available: http://arxiv.org/abs/2109.12298

[27] N. Papernot, M. Abadi, Ú. Erlingsson, I. Goodfellow, and K. Talwar, "Semi-supervised Knowledge Transfer for Deep Learning from Private Training Data," Mar. 2017, arXiv:1610.05755 [stat]. [Online]. Available: http://arxiv.org/abs/1610.05755

[28] H. Xu, S. Shrestha, W. Chen, Z. Li, and Z. Cai, "DP-FedLoRA: Privacy-Enhanced Federated Fine-Tuning for On-Device Large Language Models," Sep. 2025, arXiv:2509.09097 [cs]. [Online]. Available: http://arxiv.org/abs/2509.09097

[29] K. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, H. B. McMahan, S. Patel, D. Ramage, A. Segal, and K. Seth, "Practical Secure Aggregation for Privacy Preserving Machine Learning," 2017, publication info: Preprint. MINOR revision. [Online]. Available: https://eprint.iacr.org/2017/281

[30] K. Laine, "Simple Encrypted Arithmetic Library 2.3." 2017, microsoft Research / SEAL documentation.

# Appendix A — WORKED EXAMPLE: PRIVACY ACCOUNTING FOR DP-SGD

This appendix provides a concise, reproducible recipe for computing privacy loss $(\varepsilon, \delta)$ for DP-SGD training using Rényi Differential Privacy (RDP) / moments-accountant accounting. It includes (i) notation and conceptual sketch, (ii) a practical worked recipe, (iii) the **actual aggregated results** from our runs (three seeds per noise multiplier $\sigma$), and (iv) minimal Opacus / TensorFlow-Privacy skeletons to reproduce the accounting.

## A.1 DP-SGD: MECHANISM AND NOTATION

We use the standard DP-SGD setup (Abadi et al., 2016):

- Dataset size: $N$.
- Batch size: $b$.
- Sampling probability: $q = \frac{b}{N}$.
- Number of epochs: $E$.
- Total gradient steps: $S = \lceil \frac{N}{b} \rceil \cdot E$.
- Per-example clipping norm: $C$.
- Gaussian noise multiplier: $\sigma$ (noise added has variance $\sigma^2 C^2$).
- Target $\delta$ (commonly chosen $\leq 1/N$; we use $\delta = 10^{-5}$ in examples and experiments).

A randomized mechanism $\mathcal{M}$ is $(\varepsilon, \delta)$-differentially private if for all neighbouring datasets $D, D'$ and measurable sets $S$:

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

DP-SGD composes per-step privacy loss across training steps. The RDP accountant computes per-step Rényi divergences and composes them additively across $S$ steps; the composed RDP is then converted to $(\varepsilon, \delta)$ by numerical optimization over the Rényi order $\alpha$.

## A.2 RDP $\rightarrow (\varepsilon, \delta)$ (conceptual)

Let $D_\alpha$ be the order-$\alpha$ Rényi divergence accumulated across all steps. Then for any $\alpha > 1$,

$$\varepsilon(\alpha) = \frac{D_\alpha - \log \delta}{\alpha - 1}.$$

The final $\varepsilon$ is $\min_{\alpha > 1} \varepsilon(\alpha)$. Practical libraries (Opacus, TensorFlow-Privacy) perform this optimization numerically—prefer the library routines rather than hand-optimizing.

## A.3 WORKED RECIPE (practical)

1. Choose dataset/model (toy example: MNIST, $N = 60{,}000$).
2. Fix hyperparameters: batch $b$, epochs $E$, clip $C$, try several $\sigma$-values.
3. Compute $q = b/N$ and $S = \lceil N/b \rceil \cdot E$.
4. Run DP-SGD using Opacus (PyTorch) or TF-Privacy (TensorFlow). Use the library's privacy accounting API to obtain $\varepsilon$ for chosen $\delta$.
5. Record rows: $\sigma, C, \varepsilon, \delta,$ test-acc, runtime, seed in `results/eps_results_final.csv`.

## A.4 EXPERIMENT METADATA (this work)

The aggregated runs reported in the repository used:

- Dataset: MNIST, $N = 60{,}000$ (train).
- Batch $b = 128$, Epochs $E = 15 \Rightarrow$ Steps $S = 7035$ (as recorded in CSV).
- Clip norm $C = 1.0$.
- Seeds: {42, 43, 44} (three independent runs per $\sigma$).
- Target $\delta = 1 \times 10^{-5}$.
- Opacus version: 1.5.4; PyTorch: 2.9.1+cpu (as recorded in CSV metadata).

## A.5 AGGREGATED RESULTS (MEAN ± STD ACROSS SEEDS)

Table 4 shows the aggregated (per-$\sigma$) accounting and utility numbers computed from `results/eps_results_final.csv` that accompany this paper. For baseline (non-private) runs $\varepsilon$ is not applicable.

**Notes on the table.**

- The baseline rows are the non-private runs (no Opacus privacy engine attached) and therefore have no $\varepsilon$ (marked '—').
- The large $\varepsilon$ at $\sigma = 0.25$ stems from the relatively small noise multiplier combined with the sampling / steps schedule (these numbers are reproduced directly from the Opacus accountant output saved in `results/eps_results_final.csv`).
- Accuracy means and standard deviations were computed over the three seeds reported in the CSV.

Table 4: DP-SGD aggregated accounting (MNIST; three seeds per $\sigma$; batch=128; epochs=15; steps=7035; $\delta = 1 \times 10^{-5}$).

| **Noise** $\sigma$ | $\varepsilon$ (mean) | Test acc (mean) | Test acc (std) | runs |
|---|---|---|---|---|
| baseline | — | 0.9244 | 0.0013 | 3 |
| 0.25 | 118.6396 | 0.8408 | 0.0021 | 3 |
| 0.50 | 8.1861 | 0.8410 | 0.0020 | 3 |
| 0.75 | 1.8250 | 0.8409 | 0.0021 | 3 |
| 1.00 | 0.9152 | 0.8407 | 0.0019 | 3 |
| 1.50 | 0.4869 | 0.8401 | 0.0021 | 3 |
| 2.00 | 0.3373 | 0.8401 | 0.0012 | 3 |
| 3.00 | 0.2114 | 0.8395 | 0.0018 | 3 |
| 4.00 | 0.1549 | 0.8390 | 0.0013 | 3 |

- Steps $S$, epoch count, batch size, and $\delta$ are recorded in `results/eps_results_final.csv` and were used by the accountant to compute the shown $\varepsilon$.

# A.6 INCLUDING THE PRIVACY-UTILITY FIGURE

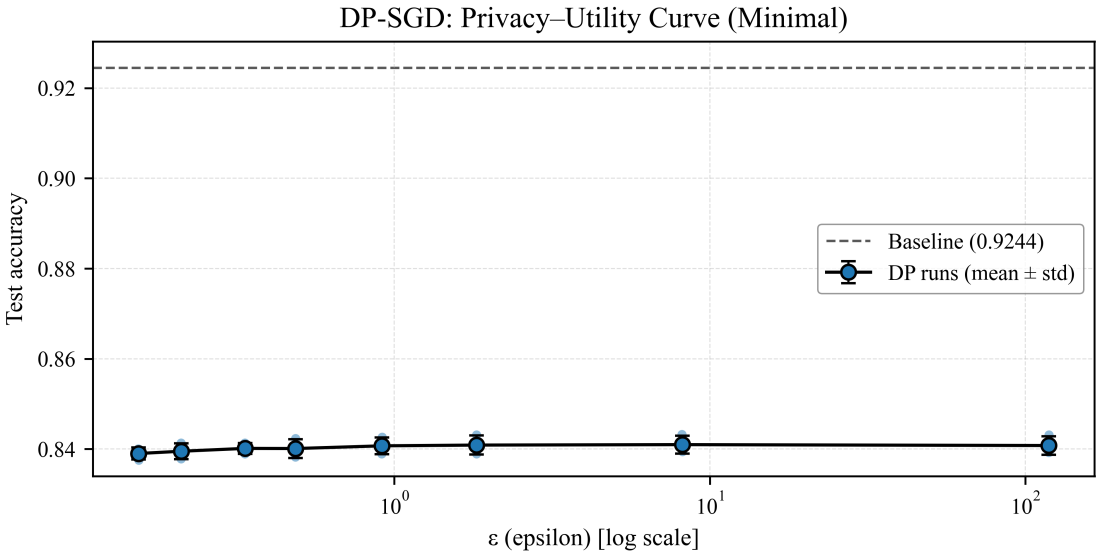


Figure 4: DP-SGD Privacy–Utility Curve (MNIST; 15 epochs; batch=128). Horizontal axis is the privacy budget $\varepsilon$ (log scale). The non-private baseline accuracy appears in the main paper figure.

**Notes on the figure:** ensure the file `epsilon_vs_accuracy.png` (or PDF) is uploaded to one of the paths listed in . If you have an SVG, convert it to PDF or PNG first.

# A.7 MINIMAL REPRODUCIBLE CODE (skeletons)

Place these snippets inside the repository notebook `notebooks/privacy_accounting.py` or `notebooks/privacy_accounting.ipynb` to reproduce privacy accounting and to re-generate the `eps_results_final.csv` file.

**Opacus (PyTorch) – skeleton**

```
# pip install torch torchvision opacus
import torch
from torch import nn, optim
from opacus import PrivacyEngine
from torchvision import datasets,transforms
from torch.utils.data import DataLoader
import time, numpy as np

# Hyperparams
N = 60000
batch = 128
epochs = 15
sigma = 1.0
clip = 1.0
delta = 1e-5
seed = 42

# Data loader
transform = transforms.Compose([transforms.ToTensor()])
train_ds = datasets.MNIST('./data', train=True, download=True, transform=transform)
test_ds  = datasets.MNIST('./data', train=False, download=True,
    transform=transform)
train_loader = DataLoader(train_ds, batch_size=batch, shuffle=True)
test_loader  = DataLoader(test_ds, batch_size=1024, shuffle=False)

# Model
model = nn.Sequential(nn.Flatten(), nn.Linear(28*28,256),
    nn.ReLU(),nn.Linear(256,10))
opt = optim.SGD(model.parameters(),lr=0.01)
criterion = nn.CrossEntropyLoss()
steps = int(np.ceil(N / batch) * epochs)
sample_rate = batch / N

# Make private
privacy_engine = PrivacyEngine()
model, opt, privacy_train_loader = privacy_engine.make_private(
    module=model,
    optimizer=opt,
    data_loader=train_loader,
    noise_multiplier=sigma,
    max_grad_norm=clip,
```

```
)

# Train
start = time.time()
for _ in range(epochs):
    for x,y in privacy_train_loader:
        opt.zero_grad()
        loss = criterion(model(x), y)
        loss.backward()
        opt.step()
runtime = time.time() - start

# Get epsilon
print(f"sample_rate={sample_rate:.6f},␣steps={steps}")
eps = None
try:
    eps = privacy_engine.get_epsilon(delta=delta)
except Exception:
    try:
        eps=privacy_engine.accountant.get_epsilon(delta)
    except Exception:
        eps = float("nan")
print(f"Computed␣epsilon␣(delta={delta}):␣{eps:.6f}")
```

### TensorFlow-Privacy (accountant helper) – skeleton

```
# pip install tensorflow tensorflow-privacy
from tensorflow_privacy.privacy.analysis import rdp_accountant

N = 60000
batch = 128
epochs = 15
sigma = 1.0
delta = 1e-5
q = batch / N
steps = int((N / batch) * epochs)
orders = [1 + x/10. for x in range(1, 100)]

rdp = rdp_accountant.compute_rdp(
    q=q,
    noise_multiplier=sigma,
    steps=steps,
    orders=orders
)
eps, opt_order = rdp_accountant.get_privacy_spent(orders, rdp, target_delta=delta)
print(f"Epsilon:␣{eps:.6f}␣at␣optimal␣order␣{opt_order}")
```

## A.8 PRACTICAL NOTES AND RECOMMENDED DEFAULTS

- **Choice of $\delta$:** a common rule is $\delta \leq 1/N$. For MNIST ($N = 60{,}000$) using $\delta = 10^{-5}$ is reasonable for illustration; for sensitive datasets choose smaller $\delta$ (e.g., $10^{-6}$ or $1/N^2$).
- **Clip norm $C$:** typical values are 0.5–1.0. Too-small $C$ can harm optimization; tune $C$ with the learning rate.
- **Noise multiplier $\sigma$:** larger $\sigma$ reduces $\varepsilon$ (tighter privacy) but harms utility. Try 0.5–2.0 as starting points.
- **Repeat runs:** run $\geq 3$ seeds and report mean$\pm$std for accuracy and optionally for $\varepsilon$ if accountant variance arises across seeds or masking modes.
- **Runtime logging:** record wall-clock time to reflect computational cost; this helps practitioners weigh privacy–utility–cost trade-offs.

## A.9 REPRODUCIBILITY CHECKLIST (for the repo)

- **Environment:** include a 'requirements.txt' with exact versions (we record Opacus/-Torch versions in CSV).
- **Determinism:** full bitwise reproducibility on GPU is difficult; run the accounting on CPU for deterministic privacy results or explicitly record device/seed and accept small run-to-run variation.
- **Privacy orders:** use a dense set of orders for RDP $\rightarrow (\varepsilon, \delta)$ conversion (the 'orders' list above works well).
- **Runtime logging:** record wall-clock time per run and total cost to help practitioners evaluate compute vs privacy trade-offs.

**Cite this appendix:** "See Appendix 8 for a reproducible DP-SGD accounting example and the companion repository for notebooks and raw results."