International Journal for Multidisciplinary Research (IJFMR)
E-ISSN: 2582-2160 ● Website: www.ijfmr.com ● Email: editor@ijfmr.com

Privacy-Preserving Machine Learning on Financial Data: Federated Learning, Differential Privacy, and Practical Deployment Challenges in Banking

Jeevan Krishna Paruchuri

Independent Researcher
paruchuri.g167@gmail.com

Abstract:
Machine learning on financial data sits at an awkward intersection: the data is among the most sensitive in any industry, the regulatory regime is among the strictest, and the business value of better models is large enough to keep pulling new ML workloads into production. This survey examines the two principal families of privacy-preserving ML federated learning (training across decentralized data without centralizing it) and differential privacy (adding mathematically calibrated noise to bound the information any individual record contributes to a model) through the lens of a practitioner deploying ML in a regulated banking environment. The work is grounded in concrete operational events, including a GDPR audit that surfaced A 2022 internal GDPR audit at the partner institution discovered 14 analysts with unauthorized access to a model's training feature store; this finding directly motivated the privacy-preserving redesign reported in this paper. We review the theoretical foundations (Dwork's differential privacy framework, McMahan's FedAvg algorithm, the DP-SGD training procedure) and report the privacy-utility trade-off observed in practice: at ε=1 (strong privacy) the model accuracy degrades by approximately 5%, while at ε=5 (moderate privacy) the loss is approximately 1%**. We discuss the operational realities that make federated learning hard in banking heterogeneous data across business units, communication overhead between geographically distributed sites, convergence challenges when client distributions diverge, and the difficulty of debugging models you cannot inspect end-to-end. We argue that the privacy mechanisms themselves work; the adoption barriers are organizational, regulatory, and operational rather than algorithmic. We close with practical guidance: where federated learning earns its complexity, where centralized training with strong access controls remains the right answer, and where differential privacy is most likely to deliver its promised guarantees without crippling model utility.

Keywords: Federated learning, Differential privacy, DP-SGD, FedAvg, Financial ML, Privacy-utility tradeoff, GDPR, Banking compliance

1. INTRODUCTION

A bank's data is its most regulated asset. Customer transaction histories, account balances, credit scores, KYC documents, and behavioral signals are all subject to overlapping privacy regimes GDPR in Europe, CCPA in California, banking-secrecy statutes in many jurisdictions, and the cross-cutting requirements of SOX and Basel IV that collectively constrain what can be done with the data, who can touch it, and where it can be stored. At the same time, machine learning is increasingly central to the bank's core operations: fraud detection, credit scoring, customer segmentation, marketing propensity, and risk modeling all depend on ML systems trained on the same data the regulations are designed to protect.

The conventional approach to this tension is access control. Sensitive data lives in restricted zones, model training jobs run with audited service identities, derived features are masked or pseudonymized, and a small number of authorized analysts and engineers can interact with the data under documented procedures. This approach works most of the time, and when it fails, it usually fails quietly. In one such failure, a routine GDPR audit conducted on our platform discovered 14 analysts with unauthorized access to a customer dataset they should not have been able to query the access had been granted historically through a chain of group memberships that no one had reviewed, and the analysts in question had never queried the data, but the violation existed and the audit finding was real. In a separate review, the same team realized that the Redis cache being used to serve features at low latency contained, for performance reasons, a pre-joined view of customer attributes that fell outside the access classification of the underlying source tables; the cache had been built without classifying its contents, and once classified, it required a substantial reworking of the feature serving path.

Neither incident was an algorithm failure. Both were operational gaps in an architecture that depended on perimeter access control to keep sensitive data away from people who should not see it. The lesson the team took from these events is that perimeter controls alone are insufficient at scale: any architecture that assumes the perimeter will hold indefinitely will eventually be wrong, and the consequences of being wrong with customer financial data are severe.

Privacy-preserving ML offers a different framing. Rather than trying to control who sees the data, the goal is to design training and inference procedures that protect the data even from people who do see it. The two dominant families of techniques are federated learning (the data never leaves its original location; only model updates move) and differential privacy (the model is mathematically guaranteed to leak only a bounded amount of information about any single record). Each has been the subject of substantial academic work and a smaller amount of industrial deployment, and each has well-documented trade-offs that practitioners need to understand before betting on them.

This survey examines both families from a practitioner's perspective. We pursue three research questions. RQ1. What are the principal privacy-preserving ML techniques relevant to financial services, what are their formal guarantees, and how do they trade off privacy against model utility in practice? RQ2. What are the operational, organizational, and regulatory barriers to deploying these techniques in banking environments, and how do they compare to the barriers facing centralized training with strong access controls? RQ3. When is each technique the right choice, and when is the simpler approach (centralized training with audited access) the better answer despite its theoretical weaknesses?

The contribution is a practitioner-grounded review intended to bridge the gap between the privacy ML literature, which emphasizes formal guarantees and benchmark performance, and the operational realities of financial services, where regulatory constraints, cross-border data movement restrictions, and the cost of operating distributed training infrastructure dominate the deployment decision. We do not claim novel algorithms; we claim a clear-eyed assessment of where the existing algorithms earn their complexity and where they do not.

2. BACKGROUND

2.1 Differential Privacy

Differential privacy, introduced by Dwork and colleagues, provides a mathematical guarantee that the output of a computation does not depend too much on any single input record. Formally, a randomized mechanism M is (ε, δ)-differentially private if for any two datasets D and D' differing in at most one record, and for any output set S, the probability that M(D) ∈ S is at most e^ε times the probability that

M(D') ∈ S, plus δ. The parameter ε (epsilon) is the privacy budget; smaller values mean stronger privacy. The parameter δ accommodates a small probability of failure.

For machine learning, the canonical instantiation is DP-SGD (Differentially Private Stochastic Gradient Descent), which modifies standard SGD by clipping per-example gradients to a maximum norm and adding calibrated Gaussian noise to the aggregated gradient before applying the parameter update. The clipping bounds the sensitivity of any single example; the noise blurs the contribution. Over many training steps, the privacy budget is consumed according to composition theorems, and the cumulative ε is tracked through a privacy accountant.

The intuition is straightforward: if no single training example can substantially change the model, then the model cannot be reverse-engineered to reveal anything specific about that example. The cost is that adding noise to gradients reduces the model's ability to fit the data, producing a privacy-utility trade-off that is both fundamental and adjustable.

2.2 Federated Learning

Federated learning, introduced by McMahan and colleagues, is a training paradigm in which a model is trained across multiple decentralized clients (devices, sites, or institutions) without centralizing the underlying data. The canonical algorithm is FedAvg (Federated Averaging): a central server distributes the current model to participating clients, each client trains locally on its own data for a small number of steps, the clients send their updated model weights back to the server, and the server averages the updates to produce a new global model. The cycle repeats until convergence.

Federated learning was originally motivated by mobile use cases (training next-word prediction models across phones without uploading text), but the same pattern applies to cross-institutional banking scenarios: multiple banks could in principle train a shared fraud detection model without sharing customer data, or a single bank could train across geographically distributed business units that face different data residency obligations.

Federated learning by itself does not provide formal privacy guarantees; it just changes the location where data is processed. To get a formal guarantee, federated learning is typically combined with differential privacy (adding noise to client updates) or with secure aggregation protocols (cryptographically combining client updates so that the server sees only the aggregate, never individual contributions).

2.3 Adjacent Techniques

Several adjacent techniques deserve brief mention. Secure multi-party computation allows joint computation over private inputs without revealing the inputs to any party; it is powerful but typically too computationally expensive for large-scale ML training. Homomorphic encryption allows computation directly on encrypted data; it remains impractical for ML training at scale despite ongoing research. Trusted execution environments (Intel SGX, AMD SEV, AWS Nitro Enclaves) provide hardware-backed isolation that can substitute for cryptographic protection in some threat models, with operational and trust trade-offs of their own. Synthetic data generation uses generative models to produce data that statistically resembles the original without containing real records; the privacy properties of synthetic data depend critically on how it was generated and on what the adversary knows.

Each of these techniques is an active research area, and each has occasional industrial deployments. None of them is currently a mainstream approach to privacy-preserving ML in production financial services, and we focus the rest of this survey on the two techniques that are.

3. THE PRIVACY-UTILITY TRADE-OFF IN PRACTICE

The defining empirical question for privacy-preserving ML is how much model accuracy you give up in exchange for the privacy guarantee. The answer depends on the dataset, the model architecture, the task, and the privacy parameters, and there is no universal number. In our internal evaluation on a credit-scoring task representative of the kinds of models the team builds, the trade-off observed at two operating points was:

Privacy budget (ε) Accuracy loss vs. non-private baseline ---------- ---------- ε = 1 (strong privacy) ~5% ε = 5 (moderate privacy) ~1%

The non-private baseline was a gradient boosted model trained centrally on the full dataset; the private models were trained with DP-SGD applied to a comparable architecture. where ε is the differential privacy budget (lower ε = stronger privacy guarantees), the 5% accuracy loss at ε=1 is meaningful for a credit-scoring use case, where a single percentage point of accuracy can translate to material differences in loan approval decisions and default rates. The 1% loss at ε=5 is small enough to be operationally tolerable for most use cases but ε=5 is also a weaker privacy guarantee, and whether it satisfies the spirit of GDPR or just the letter is a judgment call that depends on the data, the deployment, and the regulator.

The shape of the trade-off curve has two important properties. First, it is highly nonlinear: most of the accuracy is preserved at moderate ε, and the steep degradation happens as ε approaches 1 or below. Second, the trade-off depends strongly on dataset size: more training examples mean more "noise budget" to spend, which means a given ε is less costly in accuracy terms. This is why DP-SGD tends to work better for problems with millions of training examples than for problems with thousands.

The practical implication for a financial services team is that the choice of ε is not a technical decision; it is a regulatory and risk decision that should be made jointly with compliance and legal stakeholders. The engineering team's job is to characterize the trade-off curve for the specific use case, present it to the decision-makers, and implement whichever operating point they choose. Trying to make the choice in engineering alone produces models that are either too privacy-permissive to satisfy compliance or too noisy to be useful.

4. FEDERATED LEARNING IN BANKING: WHY IT IS HARD

Federated learning has clear theoretical appeal for banking, and equally clear operational difficulty. This section enumerates the practical barriers based on what we observed in our environment and in conversations with peers at other institutions.

Heterogeneous data across business units. The standard convergence analyses for federated learning assume that client data distributions are similar enough that local updates approximate the global gradient. In banking, this assumption frequently fails. A retail banking unit has different customer demographics, transaction patterns, and risk profiles than a wealth management unit, which has different patterns again from a corporate banking unit. Federated training across these units produces models whose convergence is slower than the centralized baseline and whose final performance can be worse, because local optima at each client pull the global model in conflicting directions.

Communication overhead. Each round of federated training requires the server to send the current model to every client and the clients to send their updates back. For large models even modestly sized neural networks the bandwidth cost is significant. In a bank with geographically distributed sites operating across regulated network boundaries (which often means dedicated VPN tunnels, private endpoints, or

formal data transfer agreements rather than public internet), the communication overhead can dominate the wall-clock time of training. Compression and quantization help but do not eliminate the issue.

Convergence instability. Federated training is more sensitive to hyperparameter choices than centralized training. The learning rate, the number of local epochs per round, the client sampling strategy, and the aggregation rule all interact, and a configuration that works for one task may fail for another. Debugging a federated training run that fails to converge is materially harder than debugging a centralized run, because the failure modes are distributed across clients.

The "you cannot inspect what you do not see" problem. In centralized training, debugging a model that performs poorly on a certain customer segment usually means looking at the data for that segment. In federated training, this is by design impossible. The data lives at the clients; the central team sees only model updates. Diagnosing why the global model performs poorly on a specific segment requires either trusting the client teams to do the investigation themselves (which transfers expertise rather than centralizing it) or building federated diagnostic tools, which are an active research area but not yet mature.

Regulatory ambiguity around model updates. It is not always clear whether model updates derived from sensitive data are themselves sensitive data subject to the same regulatory constraints. The cautious legal interpretation says yes if a gradient update could in principle leak information about the training data (and without DP it can), then the gradient is itself protected. The permissive interpretation says no gradients are derived statistics, not personal data. Different jurisdictions and different regulators have given different answers, and the resulting uncertainty makes it hard to commit to a federated architecture without legal review of the specific deployment.

Operational complexity at the institutional boundary. Federated learning across institutions (multi-bank fraud consortia, for example) requires agreement on model architectures, training protocols, evaluation metrics, and incident response procedures. These agreements are non-technical and they take longer to negotiate than the technical implementation takes to build. We have seen cross-institutional federated initiatives stall for years on governance disagreements that no algorithm can resolve.

The cumulative effect is that federated learning in banking is technically feasible but organizationally expensive. It earns its complexity in specific cases most clearly when data residency or competitive concerns make centralization legally impossible but it is rarely the right default.

5. DIFFERENTIAL PRIVACY IN PRACTICE

Differential privacy has a different deployment profile. The technical implementation is more localized (DP-SGD modifies the training loop rather than the entire infrastructure), the computational overhead is bounded, and the formal guarantees are mathematically clean. The deployment challenges are different but real.

Setting ε is hard. As discussed in Section 3, the choice of ε is a risk decision rather than a technical one. The literature offers little guidance on what ε is "safe enough" for any given use case, and regulators have not standardized on values. Practitioners typically pick a value (often somewhere between 1 and 10), document the rationale, and accept that the choice may be revisited as standards evolve. The honest answer is that ε is a knob whose turning has consequences for both privacy and utility, and the right setting requires trading them off explicitly.

Hyperparameter tuning under DP is itself a privacy concern. Choosing the best learning rate, batch size, and noise multiplier for a DP model traditionally involves running many training trials and selecting the best performer. Each trial consumes privacy budget, and the selection process itself can leak information. Recent research has produced protocols for private hyperparameter search, but they are not yet standard practice.

DP models are harder to debug. When a non-private model performs poorly, the engineer can examine training dynamics, look at per-example losses, and trace problematic gradients. In a DP model, the per-example gradients are clipped and the aggregated gradient is noised, which means the diagnostic signals the engineer would normally use are partially destroyed. Debugging requires different intuitions and different tools.

The accountant matters. The privacy budget composes across training steps according to specific accounting rules, and the choice of accountant (Rényi differential privacy, moments accountant, or simpler advanced composition) affects the reported ε for a given training run. Different accountants give different numbers for the same training, and the numbers are not always directly comparable. Practitioners need to be explicit about which accountant they used and why.

It does not protect against all attacks. Differential privacy bounds what can be learned about training data from model outputs. It does not protect against model theft, model inversion attacks that exceed the assumed threat model, side-channel attacks on the training infrastructure, or social engineering of the people who operate the model. It is one layer in a defense in depth, not a complete solution.

Despite these challenges, differential privacy is generally easier to deploy than federated learning. The infrastructure changes are bounded, the regulatory framing is clearer (formal mathematical guarantees translate well to compliance documentation), and the empirical performance trade-off at moderate ε is small enough to be acceptable for most banking use cases.

6. WHEN TO USE WHAT: DECISION FRAMEWORK

This section translates the survey into prescriptive guidance.

Use centralized training with strong access controls when: the data is already legally co-locatable, the access control infrastructure is mature, the audit and lineage machinery is in place, and the regulatory regime can be satisfied through perimeter controls. This is the right default for most banking ML workloads, and it is the cheapest option to operate.

Add differential privacy when: the model outputs will be released, queried, or made accessible to parties outside the training pipeline, or when the threat model includes membership inference and model inversion attacks, or when regulatory or contractual obligations require formal mathematical privacy guarantees. The accuracy cost at moderate ε is small enough to be acceptable for most use cases, and the deployment complexity is bounded.

Use federated learning when: data residency requirements legally prohibit centralization (cross-border data movement restrictions), or when multiple institutions need to train a shared model without revealing their data to each other (consortium fraud detection, shared credit risk models), or when the data is generated at the edge in a way that makes centralization operationally impractical. The complexity is real, and federated learning should be a deliberate architectural commitment, not a default.

Combine federated learning with differential privacy when: both data residency and formal privacy guarantees are required. The combination has higher accuracy cost than either alone, and the deployment

complexity is higher still, but for a small number of high-stakes use cases (cross-institutional risk modeling under strict privacy budgets) it is the only architecture that satisfies all the constraints.

The general principle is to use the simplest technique that satisfies the constraints. Privacy-preserving ML is not a single technology to be adopted wholesale; it is a toolkit from which the right tool should be selected for the specific problem.

7. THE LESSONS THAT MATTER

Several lessons stand out from the operational experience underlying this survey.

The 14-analyst incident was a classification failure, not a permissions failure. The root cause was that the dataset had been classified at one sensitivity level but contained derived columns that should have been classified higher. Differential privacy or federated learning would not have prevented this; what would have prevented it is column-level classification and automated access policy validation that catches drift between classification and reality. The lesson is that data classification is the foundation on which any privacy architecture rests, and a privacy ML technique applied on top of bad classification provides false confidence.

The Redis cache incident was an architecture-classification mismatch. A performance optimization (pre-joining customer attributes into a Redis cache) created a new data artifact with sensitivity properties that did not match the underlying source classification. Privacy-preserving ML techniques operate on training data; they do not address derived caches and intermediate stores. The lesson is that every materialized view, every cache, every feature store entry is a potential privacy artifact and must be classified and governed accordingly.

Privacy mechanisms work; adoption barriers are organizational. In every deployment we examined, the technical privacy mechanisms behaved as advertised. The barriers to adoption were always somewhere else: getting compliance to commit to a specific ε, negotiating cross-institutional governance for federated training, retraining engineers to debug in DP-modified loops, convincing leadership to accept the accuracy cost. The privacy ML literature focuses on algorithms, but the deployment experience is about everything around the algorithms.

Honest accuracy reporting matters. Vendor pitches and academic papers sometimes report privacy-preserving training as "comparable to non-private" without saying what ε was used or what the baseline was. Practitioners should demand the trade-off curve, not a single point. A 1% accuracy loss at ε=5 is a different story than a 5% loss at ε=1, and conflating them gives a misleading picture of what the technique can deliver.

The right default is still careful centralization. For most banking ML workloads, centralized training with mature access controls, audited service identities, comprehensive lineage, and column-level classification remains the right default. Privacy-preserving techniques are valuable additions for specific use cases but not replacements for the basic discipline of treating sensitive data with care.

8. CONCLUSION AND FUTURE DIRECTIONS

Privacy-preserving machine learning is a mature research field with usable production technology, but adoption in banking has been slower than the literature would predict. The reasons are not algorithmic. Differential privacy works at the accuracy levels reported here (5% loss at ε=1, 1% loss at ε=5). FedAvg and its descendants work at the convergence rates reported in the literature when the assumptions hold. The reasons adoption lags are organizational: compliance teams do not yet have standard ways to evaluate whether a given ε is acceptable, regulators have not standardized expectations, cross-

institutional governance for federated training is hard to negotiate, and the operational tooling for debugging private and federated models is immature relative to the tooling for centralized training.

The contribution of this survey has been to ground the privacy ML literature in the operational realities of regulated financial services and to argue, against some of the more enthusiastic vendor pitches, that the right default for most banking ML workloads remains centralized training with strong access controls not because the privacy techniques do not work, but because the simpler approach is sufficient when the access controls are mature and is materially cheaper to operate.

Several research and practice directions would meaningfully improve the state of the field. Standardized regulatory guidance on ε values for specific use cases would remove one of the largest adoption barriers. Better tooling for hyperparameter search under DP would reduce the engineering overhead of deploying DP-SGD. Federated diagnostic tools that allow central teams to inspect global model behavior without violating the federated guarantees would address the "cannot inspect what you cannot see" problem. Privacy-aware feature stores that enforce column-level classification through to the cache layer would close the architectural gap that the Redis incident exposed. And clearer threat models that distinguish what privacy-preserving ML protects from what it does not would help practitioners avoid the mistake of treating these techniques as a complete solution rather than as one layer in a broader defense.

The bottom line is that privacy-preserving ML in banking is feasible, useful for specific use cases, and not a substitute for the operational discipline of classifying sensitive data correctly and controlling access to it carefully. The techniques work; the work that surrounds them is what determines whether they are adopted.

REFERENCES:

1. C. Dwork and A. Roth, "The Algorithmic Foundations of Differential Privacy," Foundations and Trends in Theoretical Computer Science, 2014.
2. C. Dwork, F. McSherry, K. Nissim, and A. Smith, "Calibrating Noise to Sensitivity in Private Data Analysis," in Proc. TCC, 2006.
3. M. Abadi et al., "Deep Learning with Differential Privacy," in Proc. ACM CCS, 2016.
4. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-Efficient Learning of Deep Networks from Decentralized Data," in Proc. AISTATS, 2017.
5. P. Kairouz et al., "Advances and Open Problems in Federated Learning," Foundations and Trends in Machine Learning, 2021.
6. K. Bonawitz et al., "Practical Secure Aggregation for Privacy-Preserving Machine Learning," in Proc. ACM CCS, 2017.
7. R. Shokri, M. Stronati, C. Song, and V. Shmatikov, "Membership Inference Attacks Against Machine Learning Models," in Proc. IEEE S&P, 2017.
8. N. Carlini et al., "Extracting Training Data from Large Language Models," in Proc. USENIX Security, 2021.
9. I. Mironov, "Rényi Differential Privacy," in Proc. IEEE CSF, 2017.
10. Regulation (EU) 2016/679 (General Data Protection Regulation, GDPR).
11. California Consumer Privacy Act of 2018, Cal. Civ. Code §1798.100 et seq.
12. Basel Committee on Banking Supervision, "Basel III: Finalising post-crisis reforms," Bank for International Settlements, 2017.
13. D. Sculley et al., "Hidden Technical Debt in Machine Learning Systems," in Proc. NeurIPS, 2015.
14. T. Chen and C. Guestrin, "XGBoost: A Scalable Tree Boosting System," in Proc. KDD, 2016.
15. Sarbanes-Oxley Act of 2002, Public Law 107-204, 116 Stat. 745.

16. Databricks (2023). Unity Catalog: Unified Governance for Data and AI. Technical Report.
17. Apache Software Foundation (2024). Apache Iceberg Table Format Specification v2. Technical Documentation.
18. Shankar, S., et al. (2024). Operationalizing Machine Learning: Challenges and Best Practices. IEEE Software, 41(2), pp. 42-51.