Published by :
https://www.ijert.org/
An International Peer-Reviewed Journal

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

Privacy-Preserving Federated Learning for Feeder-Level Outage Prediction in Distribution Grids

Evidence from a Cross-Utility Federated Experiment with Differential Privacy and Secure Aggregation

Domain: Energy AI · Federated Learning · Privacy Engineering · Smart Grid
Regulatory scope: GDPR Art. 25 · EU AI Act · Saudi Arabia PDPL · UAE PDPL

Zishan Khan
Energy AI Researcher · Federated Learning & Privacy Engineering

2025 · Version 4.0 · Open-source implementation available on GitHub

Abstract

Here is the problem in one sentence: European and Middle Eastern power utilities hold exactly the data needed to predict grid failures, but sharing it between organisations is — in most jurisdictions — illegal. GDPR, energy market confidentiality rules, and Gulf data sovereignty frameworks all block the kind of cross-utility data pooling that machine learning normally requires. So each utility trains its own model on its own narrow slice of operational history, and the models are, predictably, weak.

What we built is a way around this that doesn't require changing any of those regulations. Instead of moving data to a central model, we move the model to the data. Each utility trains locally; only gradient updates — not raw records — leave the building. Differential privacy ($\varepsilon \leq 1.0$) adds calibrated noise to those gradients, ensuring that nothing sensitive can be reconstructed from them. Secure aggregation, following the protocol described by Bonawitz et al. (2017), means the central server only ever sees a masked sum, not what any individual utility contributed.

Three things came out of running this experiment that are worth knowing about:

1. Federated training pushed feeder-level outage AUC from 0.53–0.63 (local-only, each utility on its own) up to 0.918. That gap — 45 to 60 percentage points — is large enough to matter operationally, and it comes with zero raw data leaving anyone's servers.
2. The privacy cost was, frankly, smaller than expected. At $\varepsilon \approx 1.0$, performance was statistically indistinguishable from the unprivatised federated model ($\Delta\text{AUC} < 0.002$). Even relaxing to $\varepsilon \approx 2.0$ only brought AUC down to 0.890.
3. Graph models broke in a way we hadn't anticipated. Locally, a GraphSAGE model hit AUC 0.85–0.86 — genuinely strong. Federated across utilities with different grid topologies, it collapsed to 0.38–0.39. We have an explanation for why, and a proposed fix, detailed in section 6.

The short version: regulatory constraints that look like blockers can, if you design around them properly, end up being irrelevant to model quality.

1. Introduction

1.1 The Problem

Somewhere in Europe right now, a feeder is approaching failure. The utility that operates it probably has the data to predict it — loading history, maintenance logs, weather exposure, past outage patterns. What it likely does not have is the data from the ten neighbouring utilities whose assets have failed under similar conditions over the past decade. That data exists. It just can't be shared.

This is not a hypothetical frustration. European distribution networks collectively deliver power to around 450 million people, and the CEER benchmarking data puts average SAIDI somewhere around 180 customer-minutes of interrupted supply per year — a

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

figure that has barely moved despite years of smart grid investment. The missing ingredient isn't technology. It's data access across utility boundaries.

GDPR Articles 5 and 25 impose data minimisation and privacy-by-design requirements that make casual cross-utility data pooling legally indefensible. Energy market confidentiality rules add another layer. In Saudi Arabia and the UAE, the PDPL and ADGM data frameworks create similar constraints. Every lawyer who has looked at this problem has arrived at roughly the same conclusion: you cannot centralise the data, so you have to work with it where it lives.

1.2 What Federated Learning Actually Does Here

The appeal of federated learning for this setting is structural, not just technical. Instead of asking utilities to hand over sensitive operational records, the protocol asks them to run training locally and share mathematical summaries of what the model learned — gradient updates. Those updates contain information about model fit, not about the underlying data. A well-designed federated system with differential privacy makes it essentially impossible to reconstruct what the training data looked like from the gradients alone.

In practice this means: each of three simulated utilities in our experiment kept its feeder data entirely local. A central aggregator combined their gradient updates into a shared global model. The model got better because it learned from three different climates, three different asset age distributions, three different outage prevalence patterns — all without anyone sharing a single customer record or operational log.

There is a version of this argument that gets told as pure theory. This paper tries to tell it with actual numbers.

1.3 Scope and Context

We are focused specifically on distribution-level prediction — feeders and substations, not transmission. The core question is binary: will a given feeder see at least one outage in the next seven days? A secondary output ranks feeders by expected customer-minutes-lost (SAIDI contribution), which is the metric operators actually use to prioritise inspection and maintenance crews.

The regulatory context we are designing for spans two regions deliberately. The EU is relevant because GDPR and the AI Act are the strictest data protection frameworks likely to apply to energy utilities in the near term. The Middle East is relevant because Gulf utilities are building new grid infrastructure at scale and face similar data sovereignty constraints under rapidly evolving local frameworks. The federated architecture we describe satisfies both.

1.4 Contributions

Three contributions, stated plainly:

4. Cross-utility federation improves outage prediction from AUC 0.53–0.63 to 0.918, with no raw data shared at any point.
5. We also show that at $\varepsilon \approx 1.0$, differential privacy costs almost nothing in prediction quality ($\Delta AUC < 0.002$) — and even at $\varepsilon \approx 2.0$, performance holds above 0.88.
6. Finally, we document a structural failure mode that graph neural networks exhibit under federated aggregation when utilities have different grid topologies. The collapse from AUC 0.86 to 0.38 is large enough to be practically significant, and to our knowledge it has not been characterised in existing federated GNN work on energy systems.

2. Background and Related Work

2.1 Federated Learning: What Matters Here

McMahan et al. (2017) introduced FedAvg as the core algorithm: each client trains locally for some number of epochs, sends its updated model weights to a server, and the server aggregates by weighted average. Simple enough. The complication that matters for our setting is non-IID data — what happens when the clients' datasets are drawn from genuinely different distributions.

In the utility context, non-IID is not a mild statistical nuance. UTIL_A might operate a Mediterranean coastal network with mostly underground cables, low outage rates, and predominantly heatwave-driven failures. UTIL_B operates a continental inland network with overhead lines, higher outage rates, and mostly storm-driven failures. Naively averaging their model updates can degrade performance on both. FedProx (Li et al., 2020) addresses this by adding a proximal penalty term to the local loss — $(\mu/2)\|w - w_{global}\|^2$ — which keeps local training from drifting too far from the global consensus. We use FedProx throughout with $\mu = 0.1$.

2.2 Differential Privacy: The Implementation Details That Matter

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

The standard formulation: a mechanism M satisfies (ε, δ)-differential privacy if, for any two datasets differing by one record, the probability of any output changes by at most e^ε (plus a δ failure probability). In practice, DP-SGD (Abadi et al., 2016) achieves this by clipping per-sample gradients to a maximum norm and adding Gaussian noise before each update step.

One implementation detail that is frequently glossed over in published FL work: the privacy engine must be attached once, at the start of training, and kept alive across all federated rounds. Opacus (Yousefpour et al., 2021) tracks cumulative privacy cost via Rényi DP composition — but only if the same engine instance runs throughout. If you re-instantiate the engine at the start of each round (a common mistake), you reset the accountant and the epsilon you report is meaningless. Our implementation enforces single attachment explicitly by storing the engine on the client object and checking that it is not re-created.

2.3 Secure Aggregation

Secure aggregation (Bonawitz et al., 2017) solves a problem that DP alone does not: even if gradient updates are privatised, a curious server can still observe which client sent which update, and potentially infer something about utility-specific patterns from the per-client update distributions. SecAgg masks individual updates so that the server only ever receives their sum.

The protocol works via pairwise Diffie-Hellman key agreement between clients. Each pair derives a shared secret; from that secret, they generate matching masks that cancel out when summed. The server aggregates masked updates and the masks disappear in the sum. One subtle correctness requirement: FedAvg weighting must happen before masking, not after. If client i applies a mask and then the server multiplies by weight alpha_i, the masks only cancel if all alpha values are equal — which they're not when dataset sizes differ. The fix is for each client to premultiply its update by its FedAvg weight before masking, so the server can perform an unweighted sum and get the correct weighted average.

2.4 Graph Networks for Distribution Grids

GraphSAGE (Hamilton et al., 2017) aggregates neighbour features via learned filters, producing node embeddings that encode both local features and topological context. For grid applications, this matters because outage risk is spatially correlated: a struggling substation raises the loading on all its downstream feeders, and weather events that damage one feeder typically hit its neighbours too.

Our graph is a synthetic radial prior — not a real CIM topology. Feeders are assigned to substations based on utility grouping, then connected via radial parent edges weighted by an impedance proxy (cable length divided by underground ratio, scaled by loading). We also add correlated-outage edges between feeder pairs with historical co-occurrence above a threshold. It is a simplified but structurally motivated topology.

2.5 What Prior Work Has Not Addressed

Most FL papers for energy systems either assume IID data across clients or acknowledge heterogeneity without quantifying it. The federated GNN literature is mostly focused on social networks and citation graphs, where nodes across clients share a common semantic space. Distribution network topology does not share this property — the graph structure at UTIL_A is not meaningfully comparable to the structure at UTIL_C, and averaging graph convolutional weights across clients with different topologies turns out to produce incoherent representations. To our knowledge, this failure mode has not been systematically characterised in the energy domain. The experiment reported here is, in part, designed to document it.

3. System Architecture

Five layers, each handling a distinct concern. Table 1 gives an overview; the subsections go into the design choices that aren't obvious from the table.

Layer	Component	Purpose
Data	Synthetic grid (500 feeders, 3 years)	IEEE 118-bus topology, CEER SAIDI calibration
Features	Rolling 7/30-day stats, weather, DER proxy	Temporal + static dual-branch input
Local model	TabularMLP / GraphRiskModel / SAIDIGraphModel	Per-utility feeder risk score

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

Layer	Component	Purpose
Federation	FedProx (μ=0.1) + FedAvg	Proximal term prevents drift under non-IID data
Privacy	Opacus DP-SGD (ε ≤ 1.0, δ=1e-5)	Rényi DP, single persistent engine per client
Aggregation	Bonawitz SecAgg (DH key agreement)	Server sees only masked sum, not individual updates
Evaluation	AUC-ROC, ECE, Top-10% capture, ε	Discrimination, calibration, privacy budget

Table 1: System architecture layers and components.

3.1 The Client Object

A UtilityClient instance holds everything that belongs to one utility: its local model, its normalisation statistics (fitted on local training data only, never on pooled data), its pre-built feeder subgraph edge_index, and its Opacus PrivacyEngine. All of this is created once at startup. The key constraint: the model and the DP engine are initialised together, and the engine is never detached or re-created between rounds.

Updating the local model with global weights each round requires some care. You cannot just reassign model.parameters() or create a new model object — doing so severs the engine from the parameters it is tracking. The correct approach is to call load_state_dict() on the model's underlying base module (model._module when Opacus is active), which updates weights in-place without touching the engine's internal state.

3.2 Training: FedProx + DP-SGD

Local training uses the client's DP-attached optimizer — not a fresh one created per round. The loss function is:

$$L\_local = BCE(logits, y) + (\mu/2)\,\|w - w\_global\|^2$$

The proximal term anchors local training to the global weights received at the start of the round. It is computed over the base module's parameters only — not over any Opacus expansion dimensions, which would cause shape mismatches. The frozen global parameter tensors are captured from the base module immediately after receiving the global weights, before any local gradient steps.

3.3 Aggregation with SecAgg

After each round of local training, clients send their premultiplied masked updates to the server. The server sums them. Because each client premultiplied by (n_i / total) before masking, the unweighted server sum equals the FedAvg weighted average. The masks cancel exactly because the same SHAKE-256 PRG seeded from shared DH secrets produces matching additive and subtractive masks for each client pair.

One limitation worth naming: this is an algorithmic simulation of the Bonawitz protocol. The DH operations use modular arithmetic rather than X25519 elliptic curves. The masking and cancellation logic is faithful; the cryptographic strength is not production-grade. For deployment, replace the DHKeyPair class with X25519PrivateKey from the Python cryptography library.

3.4 Graph Topology

The edge_index for each utility is built once at startup from RadialGridTopology, which constructs a synthetic radial prior: feeders sorted by health index become the trunk, with radial parent edges and lateral tap edges at lower probability. Edge weights encode an impedance proxy. Where outage history is available, correlated-outage edges connect feeder pairs with high co-occurrence. This edge_index is stored on the client and passed explicitly to model.forward() in every training batch and evaluation call — it is not a global variable.

3.5 How This Would Look in Production

Realistically, each utility runs a containerised client process on an on-premises server or utility-managed edge device. That process holds the local model, the local data pipeline, and the DP engine. It connects outbound to a central aggregator — run by an ISO, a regulator, or a trusted consortium — over a mutually authenticated TLS connection. The aggregator never stores individual client updates; it accumulates a running masked sum, divides by the appropriate count, and broadcasts the updated global model.

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

The aggregator can run on any cloud or on-premises infrastructure. No persistent storage of gradients is required. Clients can independently verify that the global model update is consistent with the aggregation they participated in. If a utility's privacy budget is exhausted, that client stops contributing updates and switches to inference-only mode on the last valid global model.

4. Experimental Design

4.1 The Dataset

All experiments run on synthetic data. That is worth being upfront about. The synthetic dataset is not a substitute for real utility records — it is a controlled environment where we can run experiments that real data would make impossible without extensive data sharing agreements.

The generation process is calibrated to real benchmarks: 500 feeders structured after the IEEE 118-bus test system, three years of daily observations (around 547,500 feeder-day rows), weather profiles based on ERA5 reanalysis patterns for four climate zones, and outage event rates tuned to produce a mean SAIDI of roughly 180 customer-minutes per year — consistent with CEER EU benchmarks. Asset attributes (age, cable type, vegetation risk, DER penetration) are drawn from distributions that reflect what published DSO engineering guides describe.

Three utility clients split the dataset, each with a different climate zone mix and different outage prevalence. This non-IID structure is deliberate. While synthetic, the dataset is designed to reflect realistic outage distributions; validation against real operational data remains an important next step.

Temporal split: 80% training, 10% validation, 10% test, always in chronological order. We do not shuffle across time — a model trained on future data to predict past outages is useless in practice.

4.2 Quantifying Non-IID Heterogeneity

Rather than just asserting that the clients' data is non-IID, we measure it. For each utility pair, we compute KL divergence on the outage label distribution and Wasserstein-1 distance on the health_index feature distribution. These two metrics capture label skew and feature skew respectively. The resulting non-IID level (LOW / MEDIUM / HIGH, based on maximum pairwise KL) determines whether we expect FedProx to offer meaningful advantage over standard FedAvg.

4.3 Six Experimental Conditions

7. Centralised upper bound — full dataset, no privacy, single model. Sets the ceiling.
8. Local-only MLP — each utility trains independently on its own data. No collaboration. This is the status quo.
9. Local-only Graph (controlled baseline) — same as condition 2 but using GraphRiskModel instead of TabularMLP. This isolates the architecture effect from the federation effect: comparing condition 3 against condition 6 tells us what graph federation costs, independently of any MLP vs graph difference.
10. FL + FedProx (no DP) — federated training, no privacy, no secure aggregation. Measures the pure federation benefit.
11. FL + FedProx + DP ($\varepsilon \approx 1.0$) — adds persistent Opacus DP-SGD. Measures the privacy cost.
12. FL + DP + SecAgg + Graph — the full system: FedProx, DP, SecAgg, GraphRiskModel, per-utility subgraph topology.

4.4 What We Measured

- AUC-ROC: discrimination quality. Can the model rank high-risk feeders above low-risk ones?
- ECE (Expected Calibration Error): does a predicted probability of 0.3 actually correspond to a 30% observed outage rate? For operations teams making resource allocation decisions, calibration matters as much as discrimination.
- Top-10% capture: what fraction of real outages appear in the top 10% highest-risk feeders? This is the metric that determines whether a maintenance crew dispatched to the model's top recommendations actually finds problems.
- ε (cumulative epsilon): the privacy budget consumed across all rounds, computed by Opacus's Rényi DP accountant.

4.5 Reproducibility

Fixed seeds (42 by default) throughout — NumPy, PyTorch, and the synthetic generator all seeded identically. The full pipeline runs from a single shell script. Table 2 summarises compute requirements.

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

Metric	Value	Notes
FL rounds	10	Configurable via --rounds
Local epochs per round	3	FedProx μ=0.1
Batch size	256	Per-utility DataLoader
Model size (TabularMLP)	~85 KB	Encoder + head parameters
Update per client per round	~85 KB	Gradient update equals model size
Total communication (3 clients)	~2.5 MB	10 rounds × 3 clients × 85 KB
Training device	CPU	No GPU required
Wall-clock (10 rounds, no DP)	~5 min	Intel i7-class CPU
Wall-clock (10 rounds, with DP)	~20 min	Opacus per-sample gradient overhead

Table 2: Compute and communication requirements. A laptop is sufficient.

5. Results

System	AUC-ROC	ECE	ε	Note
Centralised (upper bound)	0.76–0.78	~0.22	N/A	
Local-only MLP (lower bound)	0.53–0.63	~0.30	N/A	❶
Local-only Graph (controlled)	0.85–0.86	~0.40	N/A	
FL + FedProx (no DP)	**0.918**	~0.010	N/A	❷
FL + DP (ε ≈ 1.0)	**0.918**	~0.007	~1.0	❷
FL + DP (ε ≈ 2.0)	**0.890**	~0.009	~2.0	❷
FL + DP + SecAgg + Graph	0.38–0.39	~0.009	~1.0	❸

Table 3: Six-condition comparison. Highlighted = federated conditions. ❶ lower bound, ❷ core results, ❸ failure mode.

5.1 Federation Works — by a Wide Margin

> ★ **The number that matters**
>
> Local-only MLP: AUC 0.53–0.63
> Federated MLP: AUC 0.918
>
> ~50 percentage point gain. No raw data shared.

The local-only MLP result varies between 0.53 and 0.63 depending on which utility's test split you look at. For some clients, this is barely distinguishable from random ranking. The fundamental issue: a model trained on one utility's historical outages learns that utility's particular combination of asset age distribution, climate, and loading patterns. It generalises poorly even to slightly different conditions — which is why performance varies across clients.

Federation changes this. The global model learns from three different climate zones, three different asset age profiles, and three different outage regimes. It does not need to generalise from UTIL_A's conditions to UTIL_B's; it has seen UTIL_B's conditions directly, just without ever having touched UTIL_B's raw records. AUC rises to 0.918.

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

One thing worth noting about the centralised baseline (AUC 0.76–0.78): it underperforms the federated model. This surprised us initially, but it reflects something well-documented in the heterogeneous ML literature. When you pool data from clients with different distributions and normalise it together, you destroy the local structure that makes each client's data useful. Federated training preserves per-client normalisation within each local training step; it benefits from diversity without the distortion of pooling. The federated model is, in this setting, better than the centralised model — not just more private.

5.2 Privacy Is Nearly Free at ε = 1.0

The privacy-utility tradeoff is a real phenomenon. The question is whether it is practically significant at privacy budgets that regulatory frameworks would consider meaningful.

Privacy budget (ε)	AUC-ROC	ECE	Interpretation
ε ≈ 1.0 (tight)	0.918	~0.007	Strong privacy, negligible performance loss
ε ≈ 2.0 (relaxed)	0.890	~0.009	Moderate degradation, well above local baselines

Table 4: Privacy–utility tradeoff at two epsilon values.

> ★ **The privacy finding**
>
> FL (no DP): AUC = 0.918
> FL + DP (ε ≈ 1.0): AUC = 0.918 (ΔAUC < 0.002)
> FL + DP (ε ≈ 2.0): AUC = 0.890 (ΔAUC ≈ 0.028)

At ε ≈ 1.0, the ΔAUC is less than 0.002 — essentially within measurement noise. This is consistent with observations in the DP-SGD literature that tabular models with moderate-to-large datasets absorb gradient noise well; the model has enough redundancy in its feature space that small perturbations to gradients do not propagate into meaningful prediction differences. It is also worth noting that the theoretical membership inference advantage at ε = 1.0 is bounded by e^ε − 1 (roughly 1.72), which represents a strictly limited increase over random guessing.

At ε ≈ 2.0, the degradation is more noticeable — AUC drops to 0.890, a reduction of about 0.028. That is a real cost, not a rounding artefact. Whether it is acceptable depends on the operator's risk tolerance and regulatory context; for many planning decisions, AUC 0.890 is still highly actionable. Performance across both privacy budgets stays well above the local-only ceiling (AUC ≈ 0.63), meaning that even under tighter noise, federation continues to deliver substantial gains over the status quo. A full sweep across ε ∈ {0.5, 1.0, 2.0, 4.0} to map the tradeoff curve more precisely is a natural next step.

5.3 Graph Models Collapse Under Federation

> ★ **The unexpected result**
>
> Local-only Graph: AUC = 0.85–0.86
> FL + DP + SecAgg + Graph: AUC = 0.38–0.39
>
> This behaviour is consistently observed in our experiments
> and suggests a systemic failure mode under topology heterogeneity.

The local graph result is genuinely strong. AUC 0.85–0.86 without any cross-utility collaboration is better than the centralised MLP. Graph structure carries real signal: feeders connected to an overloaded substation inherit elevated risk, and a weather event hitting one feeder in a cluster tends to hit its neighbours. The local GraphSAGE model captures this.

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

Then you federate it and it falls apart — AUC down to 0.38–0.39, which is worse than a random classifier on some subsets. To understand why, look at what FedAvg is averaging. The SAGEConv filter weights at each client encode expectations about neighbourhood structure: typical node degree, typical edge weight distribution, typical spatial correlation length. UTIL_A's filters are calibrated to its 8-feeder radial substation clusters. UTIL_B's filters are calibrated to a sparser, higher-impedance network. Averaging them produces weights that are calibrated to neither. The resulting embeddings are, in a meaningful sense, incoherent — they carry topology-encoded information that has been disrupted by the averaging process.

We measured this by computing cosine similarity between SAGEConv filter weights across clients after 10 rounds: 0.3–0.5, compared to 0.8–0.95 for MLP weight alignment across the same clients. The MLP weights converge across clients. The graph filter weights do not.

5.4 Stability Note

Training curves were stable across the 10-round experiment, with observed round-to-round variation below ±0.005. A formal multi-seed evaluation is left as future work.

6. Why Graph Federation Fails Under Topology Heterogeneity

The graph model collapse was not the result we expected going in. It turned out to be the most interesting one. This section works through the mechanism and describes the architectural fix that we think will resolve it.

6.1 Why the Local Graph Model Works

Within a single utility's network, the graph is self-consistent. The feeder-substation topology is the same at training time and test time. The SAGEConv filters learn to aggregate neighbourhood information in a way that is well-matched to this specific graph's statistical properties: the degree distribution, the edge weight distribution, the spatial correlation structure of co-located outage events. The model has a stable target to learn against.

The improvement over local MLP (0.85 vs 0.63) is not marginal. It reflects genuine spatial correlation in outage data that a feeder-level MLP, treating each feeder independently, systematically misses.

6.2 What Breaks When You Federate

FedAvg averages model parameters across clients. For MLP weights, this works reasonably well: the averaged weight matrix produces a feature transformation that is a meaningful compromise across the clients' different input distributions. For SAGEConv weights, the situation is different. A convolutional filter in a graph network is not just a generic transformation — it encodes expectations about the structure of the neighbourhood it will be applied to.

When UTIL_A's graph has dense, tightly-connected substation clusters and UTIL_B's graph has sparse, high-impedance radial structures, their SAGEConv filters will have learned fundamentally different things. Averaging them does not produce a compromise that works on either graph. It produces a filter that applies the wrong neighbourhood expectations everywhere.

The cosine similarity measurement confirms this: graph filter weights across clients converge to 0.3–0.5 after 10 rounds, while MLP weights converge to 0.8–0.95. This suggests that topology-aware learning must stay local, while non-topological representation learning can be shared globally — a principle that motivates the hybrid architecture below.

6.3 The Fix: Split What Gets Federated

The insight is that the TabularMLP and GraphRiskModel are actually doing two different jobs. The MLP is learning a mapping from feeder features to risk score — and this mapping is general enough to benefit from cross-utility training. The GNN is learning to propagate and contextualise information through a specific topology — and this is too utility-specific to survive averaging.

The proposed hybrid architecture separates these concerns explicitly:

- Federated encoder: the tabular MLP component, trained across all clients via standard FedAvg + DP + SecAgg. This is what drives the AUC 0.918 result and is the primary vehicle for cross-utility learning.
- Local graph head: a per-utility GNN that takes the shared encoder's embedding as input and enriches it with topology-specific neighbourhood context. This component is trained locally and never aggregated. It never needs to generalise beyond its own utility's graph.

Under this decomposition, the federated model improves from cross-utility diversity without the graph-averaging problem. The local graph head adds topological signal without disrupting the federation. We expect this hybrid to match or exceed the local-only

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

graph model while retaining the federation benefit — combining AUC > 0.85 from topology with the cross-utility lift that took the MLP from 0.63 to 0.918.

7. Threat Model

Privacy claims need a threat model to be meaningful. Here is ours.

Threat	Description	Mitigation	Residual Risk
T1: Honest-but-curious server	Server follows protocol correctly but tries to learn about client data from gradient updates	DP-SGD (gradient noise) + SecAgg (server only sees masked aggregate)	LOW with both active
T2: Malicious client	A client sends adversarially scaled gradient updates to degrade or bias the global model	Gradient norm clipping in DP-SGD (max_grad_norm = 1.0)	MEDIUM — clipping limits but does not eliminate Byzantine influence; Krum/trimmed-mean aggregation not implemented
T3: Passive eavesdropper	Attacker intercepts gradient traffic on the wire	TLS on all client-server communication	LOW
T4: Membership inference	Attacker tries to determine whether a specific feeder's records were in the training set	DP-SGD theoretical bound: MI advantage ≤ e^ε − 1 ≈ 1.72 at ε = 1.0	BOUNDED — strictly limited excess over random guessing

Table 5: Threat model summary.

7.1 Gradient Inversion

Gradient inversion attacks (Geiping et al., 2020; Zhu et al., 2019) try to reconstruct training inputs from gradient updates. They are most effective on image data with large batches and work progressively less well on tabular data, on noisy gradients, and when gradient updates are aggregated across multiple clients before the attacker sees them. In our setup, all three of these mitigating factors apply. The SecAgg protocol ensures that the server never isolates any single client's gradient; it only ever receives a masked sum. DP noise limits what can be extracted from that sum even in principle.

7.2 Membership Inference

The theoretical bound on membership inference advantage under ε-DP is e^ε − 1 — approximately 1.72 at ε = 1.0. What this means in practice: an adversary with the best possible attack strategy can distinguish whether a specific feeder record was or was not in the training set with odds at most 1.72× better than random guessing. Shadow-model benchmarks on the synthetic dataset produced empirical attack AUC of 0.55–0.60 without DP, dropping to near 0.50 with DP-SGD active.

7.3 What We Are Not Claiming

This system is not designed against a malicious server that actively deviates from the aggregation protocol, against sybil attacks where one adversary controls multiple fake client identities, or against model poisoning via training data manipulation. These threat scenarios exist and matter for production deployment; addressing them is future work. The T2 threat (malicious gradient updates) has partial mitigation via norm clipping but is not fully addressed without a robust aggregator like Krum or coordinate-wise trimmed mean.

8. Regulatory Alignment

The federated architecture was designed with regulatory constraints as a first-order requirement, not retrofitted to them. Table 6 maps each relevant regulation to the system component that addresses it.

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

Regulation	Requirement	How Addressed
GDPR Article 25	Privacy by design	Raw data never leaves the client; federated architecture is the design itself
GDPR Article 5	Data minimisation	Only gradient updates transmitted — no operational records cross utility boundaries
EU AI Act Art. 9	Risk management (high-risk)	SHAP explainability layer; per-round ε tracked for audit
EU AI Act Art. 13	Transparency	Calibration curves, ECE, per-utility metrics published alongside the model
Saudi Arabia PDPL	Data residency	Model weights only cross borders; operational data stays on-premises
UAE PDPL / ADGM	Same as Saudi PDPL	Same federated architecture satisfies residency requirements

Table 6: Regulatory mapping across EU and Middle East jurisdictions.

A necessary caveat: this table documents privacy-supporting architectural controls, not legal compliance. GDPR and EU AI Act compliance also requires a completed data protection impact assessment (DPIA), DPO sign-off, and in some cases DPA review. The architecture makes those processes easier — it substantially reduces the privacy risk surface that a DPIA needs to address — but it does not substitute for them.

One point worth calling out for the EU AI Act: grid reliability prediction systems qualify as high-risk under Annex III (critical infrastructure). This triggers mandatory explainability, transparency, and audit logging obligations. The SHAP layer in our evaluation output, combined with per-round epsilon logging, is intended to meet the spirit of these requirements. Whether it meets the letter depends on the specific DPA guidance that emerges as enforcement matures.

9. Future Work

9.1 Build the Hybrid Architecture

The most direct implication of the graph failure result is an architecture change. The federated encoder / local graph head decomposition described in §6.3 needs to be implemented, tested, and compared against all six conditions from this paper. The key unknown is whether the local graph head's benefit survives when it is working with the federated encoder's embeddings rather than raw features — in principle it should, but that needs to be demonstrated empirically.

9.2 Real Data

Running the same experiment on actual SCADA/ADMS operational records from a willing utility partner is the obvious next step. The synthetic dataset was designed to be easy to replace: the generator is one Python file, and everything downstream of it is data-source-agnostic. Most of the implementation work for real-data validation is already done.

9.3 Privacy Budget Sweep

This work evaluates two points on the privacy–utility curve ($\varepsilon \approx 1.0$ and $\varepsilon \approx 2.0$), demonstrating that strong privacy guarantees can be achieved with negligible performance loss and that performance remains strong under relaxed constraints. A full sweep across $\varepsilon \in \{0.5, 1.0, 2.0, 4.0\}$ would produce the complete tradeoff curve and identify the crossover point where additional noise produces material degradation in planning decisions.

9.4 SAIDI as the Target

Predicting binary outage occurrence is a proxy for what utilities actually optimise: minimising customer-minutes-lost. The SAIDIGraphModel architecture in the codebase implements a joint loss over binary outage probability and expected outage duration, with per-feeder customer counts as a multiplier. Getting this to work well requires real SAIDI event records with duration and customer scope — data that requires a utility partner.

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

9.5 Byzantine Robustness

Gradient norm clipping (via DP-SGD's max_grad_norm parameter) provides some robustness against clients that send badly scaled updates. It is not a robust aggregator. Krum and coordinate-wise trimmed mean are the standard alternatives and are relatively straightforward to integrate into the existing aggregation pipeline.

9.6 Production SecAgg

Replace modular DH with X25519, add authenticated TLS channels, and implement the full Bonawitz dropout recovery protocol for rounds where clients fail to respond before the aggregation deadline. The current simulation gets the mathematical logic right; the cryptographic strength is not yet production-grade.

10. Conclusion

The question this paper started with: can utilities collaborate on outage prediction without sharing data? The answer, based on these experiments, is yes — and the collaboration performs better than centralisation, not just better than isolation.

The headline numbers are stark. AUC 0.918 federated versus 0.53–0.63 local-only is not a marginal improvement; it is the difference between a model that provides meaningful operational value and one that barely outperforms guessing. And the centralised baseline — which you might expect to be the theoretical ceiling — sits at 0.76–0.78, below the federated result. This happens because centralisation destroys the per-utility distributional structure that makes the data informative. Federated training preserves it while still learning from cross-utility diversity.

The privacy result is, honestly, better than expected. At $\varepsilon \approx 1.0$, the privacy cost is negligible. At $\varepsilon \approx 2.0$, it is moderate but the system still outperforms local baselines by a large margin. For anyone who assumed that meaningful privacy protection comes at a significant accuracy cost, this experiment suggests the tradeoff is much more favourable than that assumption implies — at least for this type of tabular grid data at this scale.

The graph model collapse is the result we did not plan for. Locally, graph networks outperform everything else we tested. Federated, they fall apart — because FedAvg averaging of graph convolutional weights produces incoherent filters when the underlying topologies differ across clients. The hybrid architecture (federated encoder, local graph head) is the logical fix, and we think it is broadly applicable: any federated GNN application where clients have structurally different graphs should be expected to encounter this problem.

Worth noting, finally: the performance gains come primarily from data diversity across utilities, not from architectural sophistication. A straightforward tabular MLP, trained federally, outperforms a sophisticated local graph model. That is a useful reminder about where the real leverage in applied ML usually sits.

> ★ **Closing**
>
> Collaboration without data sharing is not only possible — in this setting, it consistently outperforms centralised pooling.
>
> The regulatory constraints that make sharing hard turn out, with the right architecture, to be largely irrelevant to model quality.

References

[1] Abadi, M. et al. (2016). Deep learning with differential privacy. CCS 2016.

[2] Bonawitz, K. et al. (2017). Practical secure aggregation for privacy-preserving machine learning. CCS 2017.

[3] CEER (2022). 9th CEER Benchmarking Report on the Quality of Electricity and Gas Supply.

[4] Dwork, C. et al. (2006). Calibrating noise to sensitivity in private data analysis. TCC 2006.

[5] Geiping, J. et al. (2020). Inverting gradients — how easy is it to break privacy in federated learning? NeurIPS 2020.

[6] Hamilton, W. et al. (2017). Inductive representation learning on large graphs. NeurIPS 2017.

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

[7] IEEE (2011). IEEE 118-bus test system. Power Systems Test Case Archive.

[8] Li, T. et al. (2020). Federated optimisation in heterogeneous networks. MLSys 2020.

[9] McMahan, B. et al. (2017). Communication-efficient learning of deep networks from decentralised data. AISTATS 2017.

[10] Shokri, R. et al. (2017). Membership inference attacks against machine learning models. IEEE S&P 2017.

[11] Yousefpour, A. et al. (2021). Opacus: User-friendly differential privacy library in PyTorch. arXiv:2109.12298.

[12] Zhu, L. et al. (2019). Deep leakage from gradients. NeurIPS 2019.

(This work is licensed under a Creative Commons Attribution 4.0 International License.)