Federated Learning for Fault and False Data Injection Attack Detection in the IEEE 9-bus System

1st Najla M. Aljuaid
Dept. of Electrical & Computer Engineering
University of Florida
Gainesville, FL, United States
najlaaljuaid@ufl.edu

2nd Dhruv S. Kushwaha
Dept. of Electrical & Computer Engineering
University of Florida
Gainesville, FL, United States
dhruv.kushwaha@ufl.edu

3rd Roghieh Abdollahi
NexaPower Solutions
Greer, SC, United States
R.Biroon@nexa-power.com

4th Zoleikha Biron
Dept. of Electrical & Computer Engineering
University of Florida
Gainesville, FL, United States
z.biron@ece.ufl.edu

***Abstract*—With the advent of the digitization of power grid systems, fault and cyber-attack detection have been a challenging problem in the field. Due to stealthy cyber-attacks and their similar effects on the grid's voltage and frequency, it is increasingly difficult for statistical methods to distinguish between faults and cyber-attacks. While deep learning (DL)–based approaches have shown promise, their high computational requirements and scalability limitations pose significant challenges for large-scale grid systems. To this end, we propose a novel federated learning (FL) approach that decentralizes the detection procedure by enabling clients (or grid zones) to perform fault and cyber-attack detection. We consider the IEEE 9-bus system modeled in SIMULINK. Fault and false data injection (FDI) attacks are injected into the vulnerable bus for the system based on PQ-sensitivity. The results show that the proposed FL algorithm successfully distinguishes faults and cyber-attacks, achieving competitive performance compared to baseline centralized DL approaches while reducing data centralization requirements and improving scalability.**

***Index Terms*—False Data Injection Attack, Fault, Federated Learning**

I. Introduction

Power systems stability is essential for ensuring the continuous and reliable delivery of electricity to consumers. However, due to the geographically distributed nature of power distribution systems, there are two major classes of abnormalities that threaten the stability and reliability of power systems: physical faults and cyber-attacks. This is primarily due to several vulnerable points spread over a large area, making it unfeasible and expensive to constantly monitor [1]. Among cyberthreats, False Data Injection Attacks (FDIAs) are especially dangerous because they manipulate measurement data used in system monitoring and state estimation, while potentially remaining undetected by conventional protection mechanisms. Although both physical faults and cyber-attacks can degrade system performance and threaten grid reliability, the inability to accurately distinguish malicious attacks from normal faults may allow adversarial behavior to persist within the system, resulting in long-term operational and security risks.

Existing studies have applied machine learning and deep learning techniques to detect physical faults and cyber-attacks in smart grids. Methods such as Support Vector Machines (SVM), and k-Nearest Neighbors (KNN), offer effective, practical, and scalable solutions for detecting cyber-attacks in smart grids. Deep learning methods such as Deep Neural Networks (DNNs), and Convolutional Neural network (CNN) have shown strong capability in identifying complex and subtle anomalies associated with FDIAs [2]–[4].

In parallel, fault detection and classification in power systems have been extensively studied over the past decades. Traditional signal-processing techniques, such as wavelet transform analysis, have been widely used to identify types of the faults and locations through voltage and current signal analysis [5]. More recently, deep learning-based approaches have demonstrated promising performance in detecting and classifying different fault conditions in benchmark power systems, such as the IEEE 9-bus network [6]. However, most of these approaches rely on centralized training architectures that require the aggregation of large amounts of data from distributed grid components, raising concerns regarding data privacy, communication overhead, and scalability.

To address these limitations, recent studies have explored the FL technique as a privacy-preserving distributed learning framework for anomaly detection in smart grids [7], [8]. FL enables collaborative model training without sharing raw local data, thereby preserving data confidentiality and reducing centralized data dependency. More than merely detecting

This work is supported by the National Science Foundation (NSF) under Grant No. 2441485

the cyber-attacks, identifying the location of attack injection, namely "Isolation", is essential for power systems to mitigate the impacts of faults and attacks. However, relatively limited research has focused on distinguishing cyber-attacks from physical faults in power systems. Existing studies have explored several data-driven and model-based approaches for this purpose. For example, ML-based frameworks using classifiers such as Random Forest and SVM to distinguish FDIA from fault events in Intelligent electronic devices-based energy systems [9], while FALCON utilized deep neural networks for integrated fault and cyberattack classification and localization in power distribution systems [10]. However, these methods still depend heavily on centralized architectures and large-scale deep learning models, which introduce challenges related to scalability, computational complexity, and data privacy in distributed smart grid environments. To address the privacy limitations of centralized learning, FL-based approaches have also recently emerged for cyberattack and disturbance discrimination in smart grids. For instance, Husnoo *et al.* proposed a semi-asynchronous FL framework to address communication latency and straggler effects in distributed smart-grid environments [11]. However, this study mainly focus on communication-efficient aggregation or publicly available industrial datasets rather than realistic power-system operating conditions with dynamic load behavior.

Main Contributions: This paper presents a privacy-preserving FL approach for distinguishing FDIA from physical faults in power systems under decentralized operating environments. Unlike conventional centralized approaches, the proposed method enables collaborative model training without requiring raw data sharing among distributed grid nodes. In addition, dynamic load variations are incorporated into the IEEE 9-bus simulation environment to create a more realistic representation of practical smart-grid operating conditions during both fault and cyber-attack scenarios.

The main contributions of this work are as follows:

- **A FL-based framework is developed to distinguish between physical faults and FDIA while preserving local data privacy.**
- **Dynamic load conditions are integrated into the IEEE 9-bus to improve the realism and variability of generated fault and attack datasets.**
- **A comparative investigation of CNN and DNN architectures is conducted under federated and centralized settings.**

The following outline describes the rest of this paper: Section II describes the overview and system architecture, Section III described the dataset, attack, and fault injections. Section IV provides a description of the data collection procedure and preprocessing, Section V describes the proposed approach, and Section VI provides details on experimental setup and discussions on results. Finally, Section VII discusses the conclusions of this work and future research directions.

II. Problem Statement

Modern smart-grid systems rely on telemetered operational measurements transmitted through communication networks for monitoring, state estimation, and control applications. Consequently, compromised communication channels or monitoring infrastructure may allow adversaries to manipulate transmitted measurements without requiring direct physical access to protected substation equipment. Among such cyber threats, FDIAs are particularly challenging because maliciously altered measurements may resemble genuine physical disturbances or faults, making accurate distinctions difficult under dynamic operating conditions.

In this work, the considered FDIA scenario focuses on the corruption of transmitted active-power associated with the dynamic load measurements connected to Bus 7 in the IEEE 9-bus system. The attackers often assume compromise of communication channels between field devices and the control center rather than physical tampering of current/potential transformer hardware. This assumption is consistent with practical smart-grid cybersecurity studies involving state estimation and supervisory monitoring applications. In addition to cyber-attacks, physical faults remain a major source of uncertainty in power systems. Transmission-line faults, such as line-to-line and line-to-ground short circuits, introduce abrupt disturbances in voltage, current, frequency, and power-flow measurements. Since FDIA events may produce measurement variations similar to those caused by genuine physical faults, distinguishing between fault-induced disturbances and maliciously manipulated telemetry measurements becomes a challenging detection problem in modern smart-grid environments.

Figure 1 shows the IEEE 9-bus benchmark system as the case study in this paper. The system contains three generators and nine buses, where buses 5, 7, and 9 are connected to dynamic loads. Unlike conventional static-load implementations, this study incorporates time-varying dynamic load behavior to emulate changing operating conditions encountered in practical power systems. The adopted dynamic-load model is intended to provide a controlled proof-of-concept environment capable of generating representative operating variability for evaluating the proposed FL framework.

To emulate realistic measurement uncertainty, stochastic measurement noise is added to the telemetry signals associated with the dynamic loads. The noise represents sensor inaccuracies and communication disturbances commonly encountered in practical monitoring systems.

For the FL algorithm, the IEEE 9-bus system is partitioned into three distributed clients, where each client locally stores measurements associated with a set of buses as follows:

1) Client 1: Buses 2, 8, 9
2) Client 2: Buses 3, 6, 7
3) Client 3: Buses 1, 4, 5

III. Anomaly Modeling and Implementation

The attackers aim to maximize the negative impacts on the system stability; therefore, they often inject faults/attacks on

TABLE I: Fault Parameters

Parameter	Value
Fault Distance	100 km
Fault Types	AB, AG, AC, BC
System Frequency	60 (Hz)
Fault Resistance	0.001 Ω

sensitive buses with a higher impact on system stability. To identify such buses in the system, an influence index of the voltage–active power sensitivity, $S_c(j)$, is defined to quantify the importance of buses in the power system network; for details, refer to our previous study [12]. $S_c(j)$ quantifies the impact of active-power injection at bus j on all bus voltages.

$$S_c(j) = \sum_i \left| \frac{\partial |V_i|}{\partial P_j} \right| \tag{1}$$

For the IEEE 9-bus system, the numerical results show that bus 7 is ranked first in influence among the PQ buses. Thus, bus 7 is simultaneously sensitive to system-wide active-power variations and has a strong impact on the overall system. Therefore, we choose bus 7 as the attack and fault injection location bus in this study, as shown in Fig. 1.

For this case study, we will consider four scenarios to cover different types of fault and cyber-attack at bus 7. ***Scenario 1:*** a line to line fault between phase A and phase B. ;***Scenario 2:*** a line to ground fault between phase A and ground G; ***Scenario 3:***a line to line fault between phase A and phase C; and ***Scenario 4:*** a line to line fault between phase B and phase C. Using these scenarios, a dataset is collected and used in FL algorithm to detect and isolate faults and cyber-attack in the IEEE 9-bus power system.

A. Fault Modeling

To emulate realistic transmission-line disturbances in the updated IEEE 9-bus case study with integrated dynamic loads [13], electrical faults were introduced using the *Three-Phase Fault* block in MATLAB/Simulink. Following standard transmission-system fault analysis practices, asymmetrical short-circuit faults were applied to the transmission line connecting Buses 7 and 8.

The faults scenarios considered in this study include line-to-line faults (AB, AC, and BC) and single-line-to-ground faults (AG). Fault events were activated during predefined non-overlapping time intervals to ensure that only one disturbance type occurred at a time, thereby preventing interference between transient responses associated with different fault conditions. The fault simulation parameters are summarized in Table I.

B. Cyber-attack Modeling

To generate attack measurements for the FDIA detection dataset, we implement a *proportional* measurements scaling attack module in Simulink. *Note: In this study, the attacker is assumed to possess sufficient knowledge of the IEEE 9-bus system topology and measurement structure to generate stealthy FDIA signals, consistent with prior FDIA literature.*

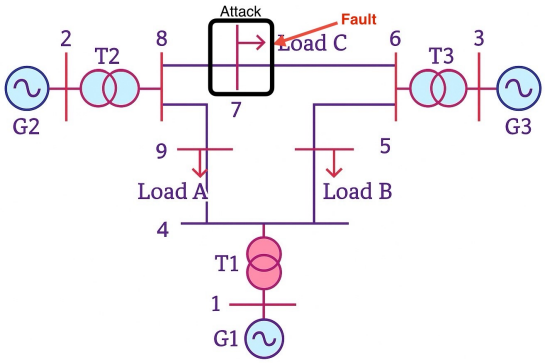

Fig. 1: IEEE 9-bus system with fault and attack location

Furthermore, the adversary is assumed to compromise communication channels between field measurement units and the control center/EMS rather than physically tampering with CT/PT hardware inside substations. The attack manipulates transmitted active power demand measurements associated with the dynamic load connected to the targeted bus. Let $z(t)$ denote the active power demand measurement of the dynamic load connected to the targeted bus. The attacked measurement $\tilde{z}(t)$ is constructed by injecting a pulse-shaped attack whose magnitude is proportional to the true measurement value (i.e., a percentage corruption):

$$\tilde{z}(t) = z(t) + \alpha(t) z(t) \tag{2}$$

The attack is activated by a binary pulse signal $p(t) \in \{0, 1\}$ generated using a Pulse Generator block. The pulse parameters define a periodic attack pattern with period T_p and duty cycle ρ, which determine the attack start/stop intervals. The pulse is scaled by a constant gain A to control the attack magnitude, yielding

$$\alpha_0(t) = A\, p(t) \tag{3}$$

where, $A = 0.15$ corresponds to 15% of $z(t)$.

To emulate timing misalignment and communication latency, the scaled pulse is passed through a transport delay block with delay T_d, as

$$\alpha(t) = \alpha_0(t - T_d), \qquad T_d = 10 \text{ s}. \tag{4}$$

IV. Data Collection and Pre-processing

Step 1-Data Collection: Running the simulation of the updated IEEE 9-bus system for all four scenarios, we will collect the required dataset for training and evaluation of the algorithm for detection and isolation. The collected dataset represents dynamic power flow metrics for the IEEE 9-bus system, including:

- **Active Power Measurements:** Measurements include generator active power outputs, load active power demand, and line active power flows associated with the IEEE 9-bus system.
- **Binary Labels:** representing 0 or 1 as occur or not occur for each of the faults and attacks.

To emulate realistic measurement uncertainty in practical power-system monitoring and telemetry, additive measurement

noise was introduced into the active power measurements associated with the dynamic loads at buses 5, 7, and 9. The noise was generated using the Simulink *Band-Limited White Noise* block [14]. The block parameters were set to *Noise Power* $1\times10^{-}8$ and sampling time 1×10^{-3}, s, which corresponds to a noise standard deviation of approximately 0.3%. The noise was applied multiplicatively to the sinusoidally modulated active power signal, i.e.,

$$y(t) = u(t)\big(1 + n(t)\big), \tag{5}$$

where $u(t)$ denotes the sinusoidally modulated active power signal and $n(t)$ is the injected noise process. The final dataset contains $40,000$ samples in total. Among them, 8,000 time instances correspond to injected fault conditions and an additional 8,000 time instances correspond to FDIA-attacked measurements.

Step 2- Data Pre-processing: In the next step, we will consider two networks for the FL algorithm:(i) a CNN-based model; and (ii) a DNN-based model. For both models, the following steps are performed:

1) **Single-label target:** The original 'fault' and 'attack' target columns are combined into a single label with three classes:
   a) **0:** No fault or attack.
   b) **1:** Fault.
   c) **2:** Attack.
2) **Feature partitioning (FL clients):** The input features are divided among three FL clients according to the bus-to-client mapping shown in Section II.
3) **Train-Test split:** The datasets samples are split into 70% training, 15% validation, and 15% test sets using stratified sampling.
4) **Normalization:** Features are normalized using a RobustScaler fitted on the training set only, and the same transform is applied to validation and test sets to avoid data leakage.

To provide temporal context for the CNN, we segment the feature sequence into overlapping windows of length L. Let τ denote the total number of time samples. For each starting index $s \in \{0, \dots, \tau - L - 1\}$, we form an input window as:

$$\mathbf{X}^{(s)} = \big[x_s, x_{s+1}, \dots, x_{s+L-1}\big] \in \mathbb{R}^{L\times F}$$

and assign the window label using the last time step within the window:

$$y^{(s)} = y_{s+L-1}.$$

This procedure produces a 3D array of shape `(num_samples, L, num_features)` for the inputs and a corresponding 1D label vector of length `num_samples`. After window construction, Steps 2–4 (feature partitioning, stratified splitting, and normalization) are applied to the windowed dataset. In the DNN pre-processing, each row of the dataset is treated as an independent sample. We implement Steps 1–4 at the beginning of this section and obtain a 2D input array of shape `(num_samples, num_features)` for the input and a corresponding 1D label vector of length `num_samples`.

V. Federated Learning Framework

A. Federated Training Procedure - FedAvg Algorithm

The key idea of the proposed method is to follow the FL paradigm, which allows multiple clients to collaboratively train an FDIA detection model without centralizing raw data, thereby preserving data privacy. Rather than transmitting local datasets to a central server, each client trains the model locally and shares only model updates, such as gradients or weights, with the server. The central server then aggregates these updates to obtain an improved global model and broadcasts the updated weights back to the clients. This aggregation strategy is known as Fedratd Averaging (FedAvg) [15] [16].

Let us consider a FL system with K clients, where client k, $k \in \{1, \dots, K\}$, holds a private local dataset $\mathcal{D}_k = \{(x_i, y_i)\}_{i=1}^{n_k}$. The objective is to learn the global model parameter, namely, w, by minimizing a weighted empirical risk across clients as described in (6)-(7).

$$\min_{w} \; F(w) = \sum_{k=1}^{K} \frac{n_k}{\sum_{j=1}^{K} n_j} F_k(w), \tag{6}$$

$$F_k(w) = \frac{1}{n_k} \sum_{(x,y)\in\mathcal{D}_k} \ell(f_w(x), y)\,, \tag{7}$$

where, $f_w(\cdot)$ denotes the detector model and $\ell(\cdot,\cdot)$ is the training loss.

The training window, $T : t \in \{1, 2..., T\}$, is coordinated by the central server and at each sampling time t , the server determines the global model parameters w^t and selects a subset of participants. The set of participating clients may vary depending on the specific scenario. For example, in our setting, all clients participate in every training round. Accordingly, the server broadcasts the global model parameters w^t to all clients in S_t. Using the obtained w^t, each client k initializes its local model and performs E local epochs of mini-batch optimization using the Adam optimizer on its local dataset $\mathcal{D}_k$. After local training, client k obtains updated parameters w_k^{t+1} and sends them to the server.

After receiving the updated models from all clients in S_t, the server aggregates them using the FedAvg. The aggregation is performed as a sample-count-weighted average of client parameters:

$$w^{t+1} = \sum_{k\in S_t} \frac{n_k}{\sum_{j\in S_t} n_j} w_k^{t+1}. \tag{8}$$

where n_k is the number of local training samples (windows) at client k. This process is repeated for T rounds. Throughout training, the raw client dataset $\mathcal{D}_k$ remains local and only model parameters are communicated between clients and the server. The complete FedAvg procedure is summarized in Algorithm 1.

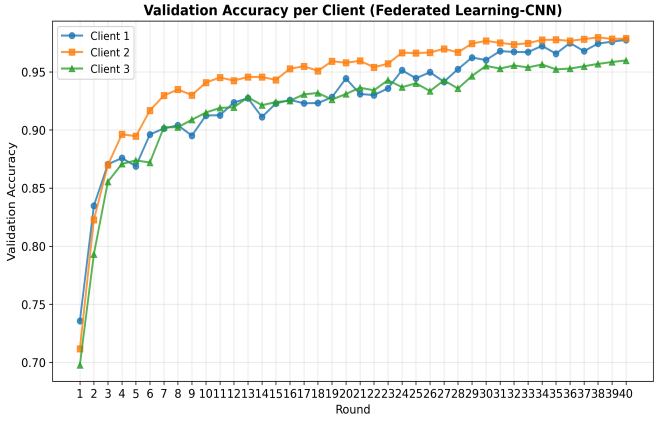

Fig. 2: Validation Accuracy per client-CNN

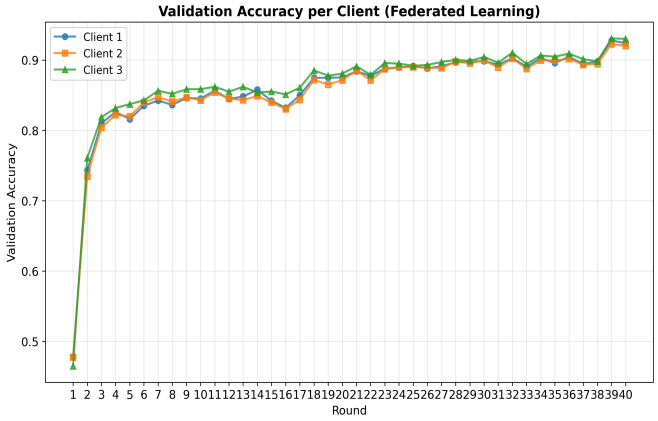

Fig. 3: Validation Accuracy per client-DNN

B. FDIA Detector Architecture: CNN-Based model

The CNN detector consists of three Conv1D layers (64–128–128 filters) with Batch Normalization, ReLU activation, and dropout. Global average and max pooling outputs are concatenated and passed through fully connected layers for three-class classification (normal, fault, attack).

C. FDIA Detector Architecture: DNN-Based model

In addition to the convolutional detector, we implement a fully connected DNN as an FDIA detector.

The DNN detector consists of three fully connected hidden layers (256–128–64), each followed by ReLU activation, batch normalization, and dropout. Dropout rates p, p, and 0.5p are used in the first, second, and third hidden layers, respectively. A final linear layer maps the learned representation to three output classes.

VI. Experimental Setup and Results

To evaluate the performance of the proposed distributed FL algorithm in distinguishing faults and FDIA, we consider four cases: (i) FL-CNN, (ii) FL-DNN, (iii) CNN, and (iv) DNN. The FL-CNN and FL-DNN cases correspond to training the CNN and DNN networks using the FedAvg algorithm, respectively. Cases (iii) and (iv) represent the conventional centralized training of CNN and DNN models, where the

Algorithm 1 Federated Averaging (FedAvg) for Training the Global Detector

Server performs: K: number of clients; B: mini-batch size; E: local epochs; η: learning rate; R: number of global rounds
Initialize global model w^0
for $t = 0, 1, 2, \ldots, T-1$ **do**
  $S_t \leftarrow \{1,\ldots,\mathrm{K}\}$ ▷ all clients participate
  for all clients $k \in S_t$ **do**
    $w_k^{t+1} \leftarrow \textsc{ClientUpdate}(k, w^t)$
    Client returns (n_k, w_k^{t+1})
  end for
  $n \leftarrow \sum_{k \in S_t} n_k$
  $w^{t+1} \leftarrow \sum_{k \in S_t} \frac{n_k}{n} w_k^{t+1}$ ▷ FedAvg aggregation
end for
Output: trained global model w^T

training dataset contains data from all buses, rather than being distributed across clients as in the first two FL-based cases. All simulations are conducted in Python 3.11 using PyTorch [17] as backend.

The CNN and DNN architectures follow the configurations described in Section V. Common training parameters include Adam optimizer (learning rate 5×104), batch size 128, ReduceOnPlateau scheduling, and CrossEntropyLoss.

The common hyperparameters for batch size - 128, learning rate scheduler - ReduceOnPlateau, and loss function as CrossEntropyLoss. The centralized DL approaches are each trained for 250 epochs. The FL parameters are: (i) Number of rounds - 40; (ii) Number of local epochs - 6. We use an Alienware M18 R2 with Intel i9-149000HX with a Nvidia GeForce RTX 4080 using Ubuntu 22.04 to conduct all simulations.

In this study, we evaluate classification performance across three scenarios: "None" (no fault, no attack), "Fault," and "Attack." The selected metrics account for the inherent class imbalance present in the fault and attack scenarios. Performance comparisons are conducted for both the CNN and DNN models using the following evaluation metrics:(i) Precision, (ii) Recall, (iii) F1 score - macro and weighted, (iv) Accuracy, (v) False positive rate (FPR), and (vi) True positive rate (TPR). The results are tabulated and presented in Table II. Due to space constraints, we abbreviate the labels as follows: N - None; F - Fault; A - Attack. The results in Table II illustrate minimal performance deviation in FL-based approaches compared to centralized DL approaches (CNN and DNN). The overall difference in CNN architectures for centralized and FL is 2.8% and in DNN architectures is 2.7%. This performance drop is compensated by low feature space per client in FL, which boosts the overall training time by roughly ~ 4 minutes. We further observe from Table II low FPR and high TPR rates, particularly in CNN cases, which imply the effectiveness of the use of temporal features obtaining by creating a sliding window on the dataset.

[1]N - None, F - Fault, A - Attack

TABLE II: Performance metrics comparison for baseline DL and federated learning

Model Type	Metrics[1]																
	Precision			Recall			F1 score				FPR			TPR			Accuracy
	N	F	A	N	F	A	N	F	A	Macro	N	F	A	N	F	A	Macro
CNN	0.999	0.999	0.995	0.998	0.996	1.000	0.998	0.997	0.998	0.9976	0.002	0.0002	0.0013	0.9981	0.9958	1.00	99.80%
DNN	0.976	0.768	0.957	0.894	0.935	0.998	0.933	0.843	0.977	0.9004	0.033	0.069	0.011	0.894	0.935	0.998	92.25%
FL-DNN	0.989	0.690	0.938	0.838	0.969	0.998	0.907	0.806	0.967	0.8721	1.180	0.712	0.939	0.837	0.969	0.998	89.55%
FL-CNN	0.976	0.946	0.974	0.974	0.941	0.984	0.975	0.944	0.979	0.9655	0.0359	0.0132	0.0065	0.9745	0.9414	0.9844	97.00%

The validation accuracy during training for each client in the FL-CNN model is shown in Figure 2, while the validation accuracy for each client in the DNN model is presented in Figure 3. Each round denotes updating the client models over 6 epochs, and the aggregated server model is updated based on the FedAvg algorithm discussed before. The validation accuracies for FL-CNN and FL-DNN are given by Fig 2 and 3, respectively. We observe convergence for each client in a decentralized manner, showing the efficacy of the proposed approach.

VII. Conclusions

This study proposes an effective and decentralized method for distinguishing between faults and attacks using a FL strategy. The results demonstrate that the models are capable of detecting both faults and attacks with high accuracy. The study was conducted using the IEEE 9-bus system. Future research will extend the proposed FL framework to larger benchmark systems, including the IEEE 39-bus and IEEE 118-bus networks, in order to investigate scalability, heterogeneous client behavior, communication overhead, and robustness under more realistic smart-grid operating conditions.

References

[1] A. S. Musleh, G. Chen, and Z. Y. Dong, "A survey on the detection algorithms for false data injection attacks in smart grids," *IEEE Transactions on Smart Grid*, vol. 11, no. 3, pp. 2218–2234, 2019.

[2] F. M. Shakiba *et al.*, "Application of machine learning methods in fault detection and classification of power transmission lines: A survey," *Artificial Intelligence Review*, vol. 56, no. 7, pp. 5799–5836, 2023. [Online]. Available: https://research.ebsco.com/linkprocessor/plink?id=3ac682c8-fa82-3235-b2fb-19b410e551f5

[3] D. Mukherjee, "A novel strategy for locational detection of false data injection attack," *Sustainable Energy, Grids and Networks*, vol. 31, p. 100702, 2022.

[4] A. Anwar, A. N. Mahmood, and M. Pickering, "Modeling and performance evaluation of stealthy false data injection attacks on smart grid in the presence of corrupted measurements," *Journal of Computer and System Sciences*, vol. 83, pp. 58–72, 2017.

[5] K. Kiruthika, G. Vishal, N. Vivek, and J. Mature, "Fault detection in ieee 9 bus system using matlab & simulink," *Research Square*, 2025.

[6] S. Jiriwibhakorn and S. Kanwal, "Time series-based fault detection and classification in ieee 9-bus transmission lines using deep learning," *IEEE Access*, vol. 13, pp. 118 135–118 146, 2025.

[7] R. Shrestha, M. Mohammadi, S. Sinaei, A. Salcines, D. Pampliega, R. Clemente, A. L. Sanz, E. Nowroozi, and A. Lindgren, "Anomaly detection based on lstm and autoencoders using federated learning in smart electric grid," *Journal of Parallel and Distributed Computing*, vol. 193, p. 104951, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0743731524001151

[8] Y. Li, X. Wei, Y. Li, Z. Dong, and M. Shahidehpour, "Detection of false data injection attacks in smart grid: A secure federated deep learning approach," *arXiv preprint arXiv:2209.00778*, 2022.

[9] B. M. R. Amin, M. J. Hossain, A. Anwar, and S. Zaman, "Cyber attacks and faults discrimination in intelligent electronic device-based energy management systems," *Electronics*, vol. 10, no. 6, 2021. [Online]. Available: https://www.mdpi.com/2079-9292/10/6/650

[10] M. Ganjkhani, M. Gilanifar, J. Giraldo, and M. Parvania, "Integrated cyber and physical anomaly location and classification in power distribution systems," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 10, pp. 7040–7049, 2021.

[11] M. A. Husnoo, A. Anwar, H. T. Reda, N. Hosseinzadeh, S. N. Islam, A. N. Mahmood, and R. Doss, "Fedisa: A semi-asynchronous federated learning framework for power system fault and cyberattack discrimination," in *IEEE INFOCOM 2023 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, 2023, pp. 1–6.

[12] D. S. Kushwaha, N. M. Aljuaid, R. Pandey, R. Biroon, and Z. A. Biron, "Stealthy false data injection attack detection and localization using reduced sparse transformer neural network," *Journal of Dynamic Systems, Measurement, and Control*, vol. 148, no. 2, p. 021007, 11 2025. [Online]. Available: https://doi.org/10.1115/1.4070031

[13] J. Pettikkattil, "Ieee 9 bus," https://www.mathworks.com/matlabcentral/fileexchange/45936-ieee-9-bus, 2025, mATLAB Central File Exchange. Retrieved November 15, 2025.

[14] The MathWorks, Inc., *Band-Limited White Noise — Simulink Block*, The MathWorks, Inc., 2026, online; accessed 7 February 2026. [Online]. Available: https://www.mathworks.com/help/simulink/slref/bandlimitedwhitenoise.html

[15] A. K. Sahu, T. Li, M. Sanjabi, M. Zaheer, A. Talwalkar, and V. Smith, "Federated optimization in heterogeneous networks," *arXiv: Learning*, 2018.

[16] B. Yurdem, M. Kuzlu, M. K. Gullu, F. O. Catak, and M. Tabassum, "Federated learning: Overview, strategies, applications, tools and future directions," *Heliyon*, vol. 10, no. 19, p. e38137, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2405844024141680

[17] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga *et al.*, "Pytorch: An imperative style, high-performance deep learning library," *Advances in neural information processing systems*, vol. 32, 2019.