2024 IEEE Power & Energy Society Innovative Smart Grid Technologies Conference (ISGT) | 979-8-3503-1360-4/24/$31.00 ©2024 IEEE | DOI: 10.1109/ISGT59692.2024.10454202

HIL Testbed-based Auto Feature Extraction and Data Generation Framework for ML/DL-based Anomaly Detection and Classification

Aditya Akilesh Mantha, *Member, IEEE,*, Arif Hussain, *Member, IEEE,* Gelli Ravikumar, *Member, IEEE,*
Department of Electrical and Computer Engineering, Iowa State University, Ames, IA, United States, 50010
Email: aditya98@iastate.edu, hussain@iastate.edu, gelli@iastate.edu

***Abstract*—In smart grid operations, sufficient data generation, including events such as faults, cyberattacks, and perturbations, is one of the key requirements for accurate Machine Learning (ML) and Deep Learning (DL) based anomaly detection and classification. However, the public datasets in power systems pose data sufficiency, bias, and uncertainty challenges. Moreover, synthetic datasets may only capture some feature vectors in power systems and complex dynamics, resulting in sub-optimal performance of ML/DL models. In this paper, we proposed the Auto Feature Extraction and Data Generation (AFEDG) framework for ML and DL models in smart grid applications such as anomaly detection and classification. We developed a Python-based Application Programming Interface (API) to automate the extraction of feature vectors and data generation processes by integrating virtual sensors. The API dynamically modulates parameters and signals within the power system model, ensuring comprehensive coverage of all event scenarios and generating unbiased, balanced datasets. The proposed API is designed to seamlessly integrate with power systems, exhibiting scalability and adaptability regardless of the number of buses within the system. We implemented the proposed AFEDG framework on a Hardware-in-the-Loop (HIL) Cyber-Physical System (CPS) testbed to generate large-scale real-time datasets. In our case study, we used the Python API to generate electric fault datasets on the modified SSN-distribution bus grid model in OPAL-RT, integrating with RT-LAB and MATLAB.**

***Index Terms*—feature vectors extraction, Python API, real-time data, RT-LAB, Smart Grid.**

I. Introduction

ML and DL are important in power systems applications because they are adaptive and data-driven, enabling sophisticated analysis and decision-making in dynamic and complex scenarios. These applications include anomaly detection, classification, predicting energy demand, and optimizing energy distribution. The accuracy, robustness, and reliability of ML and DL models are directly related to the quality of the data they are trained on. The readily available and utility datasets pose unique challenges [1], particularly regarding security and privacy. The smart grid is also exposed to cyberattacks [2], and data breaches could lead to privacy intrusions and even compromise the grid integrity. Several data generation methodologies have been proposed in the last decade using generative adversarial networks [3], [4] for smart grids. A computationally efficient method to create data sets by considering the security aspects of the power system is proposed in [5].

In the traditional and state-of-the-art approaches, we typically use a populated dataset as shown in Fig. 1a, which might not consider any logic, may contain desired features, and may not provide all the required features for training the ML or DL algorithm to achieve the desired objective. Moreover, for any new data inclusion, we must generate the new data sets and train the model repeatedly. This process is cumbersome, ineffective, and a computational burden. So, there is a need for a robust framework for data generation that replicates the diverse events encountered in power systems, from transient disturbances to longer-term system dynamics. We tried to incorporate the required or desired feature vector of an application into the API so that it executes the simulation and generates the data set. The data set follows the desired feature vectors and ensures the right data set for the required ML and DL algorithms. Also, automating the data acquisition and generation process allows it to apply to all the models, irrespective of the power system. To address the above challenges, we proposed an Auto-Feature Extraction and Data Generation (AFEDG) framework for ML and DL-based smart grid applications by integrating Python-API with a real-time simulator. The key contributions of our research are:

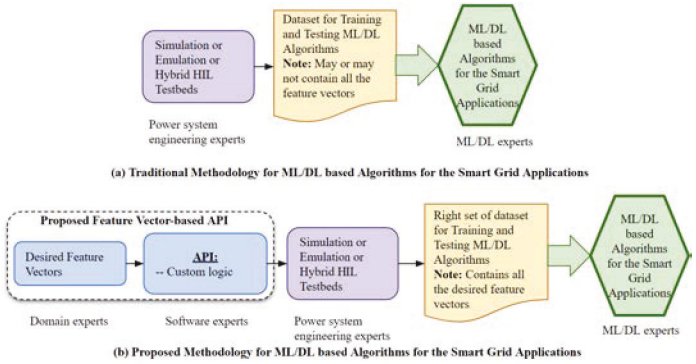

Fig. 1. Traditional vs Proposed AFEDG for ML/DL based Applications

- Proposed an auto feature extraction and data generation framework to address the need for high-quality, diverse, and balanced data sets for ML/DL-based models in smart grid applications using Python API.
- Developed and designed a Python-API integrated with RT-Lab and Simulink to generate large-scale real-time datasets using the proposed AFEDG framework.
- A detailed case study is demonstrated for various fault events in CPS using the Monte Carlo simulation method for a modified SSN-distribution grid system available within the OPAL-RT environment.

II. Proposed AFEDG Framework

The ML and DL models changed the operations and control of CPS. The interaction of computational devices and sensors allows seamless communication between cyber components and physical processes. Due to the unpredictable nature of events within CPS, these models must be equipped to rapidly detect, diagnose, and respond to anomalies in real-time. The Fig. 3 illustrates the proposed AFEDG framework implemented on the CPS testbed. The models are loaded into the OP5600 real-time simulator, which emulates the behavior of the power system model via IP/TCP protocols. The OP5600 real-time simulator is integrated with virtual sensors and actuators. These sensors, such as Phasor Measurement Units (PMUs) and Intelligence Edge Devices (IEDs), are communicated via the IEEE-61850 protocol. Data from PMUs converge at substation Phasor Data Concentrator (PDC) and communicate with virtual PMUs in OPAL-RT via IEEE C37.118 protocol.

A. Taxonomy of Feature Vectors in Cyber-Physical Systems

The power grid integrated with IEDs, Distributed Energy Resources (DERs), and AI (Artificial Intelligence) transform it into a sophisticated CPS. This integration brought a change in operation, communication, and management. As a result, the grid became more adaptive, predictive, and responsive. The cyber and physical components in the system influence the overall grid operation due to various events in the grid, as shown in Table I. The ability to capture and analyze these feature vectors is essential in understanding the system's behavior and maintaining its robustness and reliability. We

TABLE I
Feature Vectors In Cyber-Physical System

Events	Types	Equations
Faults	1. Single line to ground fault 2. Double line fault 3. Double line to ground fault 4. Triple line fault 5.Triple line to ground fault	Fault Current: $I_f = V_f / Z_f$ Where I_f is the fault current and Z_f is the fault impedance.
Cyberattacks	1. Denial of Service (DoS) attacks 2. IT attacks 3. Malware attacks	From [6] represented as $y^*(t) = (1 + \beta_{\text{case}}) \times y(t)$ $y^*(t)$ comprised signal β_{case} is cyber attack events $y(t)$ represents signal of normal condition
Perturbations	1. Small perturbations 2.Large perturbations	The small-signal stability: δ" = [P_m - P_e -D*δ']/M
Voltage Fluctuations	1. Short-term voltage fluctuations 2. Long-term voltage fluctuations	$\Delta V = I*Z$, where ΔV is the voltage fluctuation, I is the current through the line, and Z is the impedance of the line.

can generate comprehensive datasets encompassing a wide range of power system feature vectors such as faults, cyber attacks, and perturbations datasets using the proposed Python API for automated feature extraction. These datasets enhance the ML/DL models in smart grid applications such as anomaly detection and classification, thereby contributing to the reliable and efficient operation of the smart grid.

B. CPS: Actuators, Controllers, and Sensors

The computational devices are integrated into CPS for seamless communication with physical processes. Within the modernized power grid, actuators, controllers, and sensors work together to balance decision-making, action execution, and system monitoring. The proposed API provides the ability to interact with the different sensors and configure the API to generate datasets with distinct sensor characteristics. The sensors, such as Current Transformers (CTs), Potential Transformers (PTs), and PMUs, provide accurate and reliable information. PTs and CTs are represented in a vector $\mathbf{V} = [V_1, V_2, ..., V_n]^T$, $\mathbf{I} = [I_1, I_2, ..., I_n]^T$, where V_i, I_i refers to the voltage and current measured at the bus 'i' with 'i' = 1, 2, . . . , n respectively.

C. Data Polling and Sampling rate

In CPS, each device is grounded on standard communication protocols to exchange data. Communication protocols such as IEEE C37.118, Modbus, DNP3, and IEC-61850 are used standards for exchanging data with high-speed retrieval and data granularity. In PMUs, IEEE C37.118 is used for high-speed data transfer and to poll the data rates ranging from 10 to 60 samples per second, effectively capturing the system's dynamics. However, these complicated details and complex dynamics of power systems may not be captured in synthetic datasets. This distinction becomes more prominent in ML/DL applications in Smart Grid. The OPAL-RT facilitates communication with physical sensors using protocols like DNP3, MODBUS, and IEC-61850. Data granularity depends on the sampling rate and influences the accuracy and effectiveness of system modeling, monitoring, and analysis. According to the Nyquist-Shannon sampling theorem, the rate must be at least twice the highest frequency to fully reconstruct the signal.

$$f_s \geq 2f_m \tag{1}$$

D. Proposed Python API for AFEDG

The proposed API is designed to improve data generation with auto feature extraction for data-driven power systems applications. This API automates the process of data generation and feature selection, which is integrated with virtual sensors and other relevant data sources. The features are selected according to the application and configured in API to extract data with required feature vectors. The feature extraction process at bus i involves selecting relevant data from the virtual sensor outputs to capture the unique characteristics of each event. The extracted features at bus i are denoted as $X^{(i)}_{\text{event}}(t)$

$$X^{(i)}_{\text{event}}(t) = F^{(i)}_v(V^{(i)}(t), I^{(i)}(t), \ldots) \tag{2}$$

where:

- $F^{(i)}_v(V^{(i)}(t), I^{(i)}(t), \ldots)$ is the function of voltage $V^{(i)}(t)$, current $I^{(i)}(t)$, and other relevant variables such as reactive power, active power, and frequency at bus i.

The API communicates with the real-time simulator through the RT-LAB and controls the simulation parameters and

979-8-3503-1360-4/24/$31.00 ©2024 IEEE

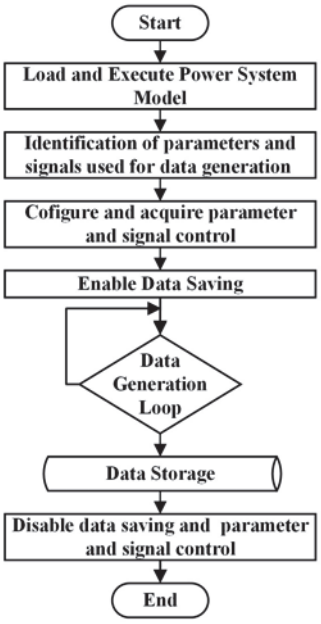

Fig. 2. Flow diagram of Python API

outputs. Fig. 2 illustrates API's workflow and enumerates the sequential steps involved in API's communication. The power system model is loaded into the real-time simulator and executed, during which the API retrieves a list of parameters and signals associated with the power system model. We must acquire control of these parameters and signals as shown in Fig. 5 to dynamically control the feature vectors using API. According to the application, specific parameters & signals are accessed and configured in API are shown in Fig. 6. The API generates the data with desired feature vectors and retrieves the data in the required format. The API overcomes data sufficiency, uncertainty, and biased challenges as follows:

1) Data Sufficiency: On the one hand, the Python API's capability to dynamically adjust and control real-time simulation parameters ensures a consistent flow of comprehensive datasets. Moreover, the controlled data generation loop is designed to produce large-scale real-time datasets. The API ensures that data is continuously generated, covering all possible scenarios and edge cases, thereby ensuring sufficiency. Also, data sufficiency depends on the complexity of the ML/DL model and application.

2) Uncertainty and Biased : On the other hand, uncertainty and bias can distort simulation results, leading to incorrect conclusions. API tackles this issue by varying certain parameters during the data generation phase while keeping others fixed. The variable parameters should be selected such that there should be a significant change from one data point to the next data point. This ensures that the generated datasets represent a broad spectrum of real-world conditions and are not biased towards any particular state. This systematic sequence of steps enables the Python API to regulate the real-time simulator and virtual sensors effectively, generating comprehensive, event-specific datasets.

III. Testbed-based Implementation and Simulation

In this section, we discussed the process of implementing the proposed AFEDG framework on testbed-based hardware in the loop (HIL) simulation to generate fault datasets. Fig. 3 describes the architecture of our co-simulation testbed consisting of Cyber-Physical simulation components. We considered the 4-bus distribution grid model available on OPAL-RT, as shown in Fig. 4. The power system model is implemented on the Cyber-Physical testbed setup [7] available at the PowerCyber lab at Iowa State University.

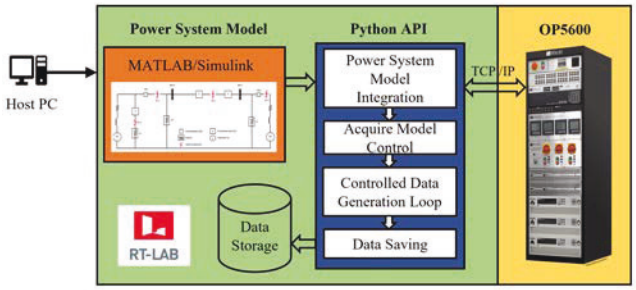

Fig. 3. Hardware-in-Loop simulation testbed architecture for proposed AFEDG framework

A. Hardware Setup

1) RT-LAB: We implemented our model on RT-LAB version 2023. 1. It is a distributed real-time platform that allows users to test dynamical models built in MATLAB/Simulink environment for HIL simulation.

2) ARTEMiS: It is a real-time power system solver that provides high stability for discrete-time state-space models [8]. It facilitates efficient parallel computing of electric circuits with multiple CPU cores.

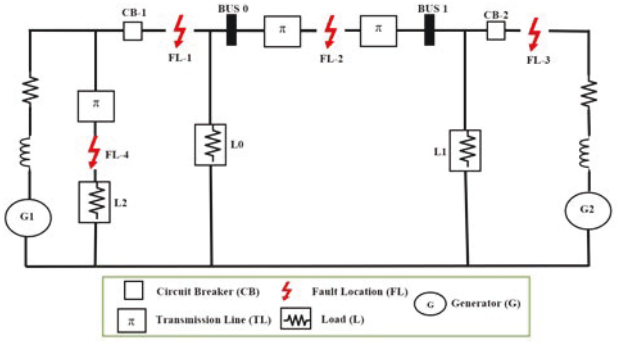

Fig. 4. Single line diagram of modified power system model

B. Integrating Power System Model to RT-LAB using Python API

We modified the existing 4-bus distribution grid model available on the OPAL-RT. The power system model is divided into the Master Subsystem (SM) and the Console subsystem (SC). The signal communication is done across two subsystems with the help of the *opComm* block. The model has utilized the *'opElectricFault'* block to create faults in the power system model at different locations. We isolated *'opElectricFaultSelector'* block and implemented it in the SM subsystem to gain control over fault parameters. We used a Python-based script to integrate MATLAB/Simulink with RT-LAB using RT-LAB predefined libraries. Upon completing the

979-8-3503-1360-4/24/$31.00 ©2024 IEEE

Simulink model, we built it in RT-LAB and loaded it into the OP-5600 real-time simulator utilizing the available target subsystem. After successful loading, the model was executed. We used the Monte Carlo simulation method [9] in Python API to generate the data and control the simulation of feature vectors in power systems. The API automates the feature vector extraction process as shown in Fig. 6 by acquiring control of parameters and signals within the power system model as specified.

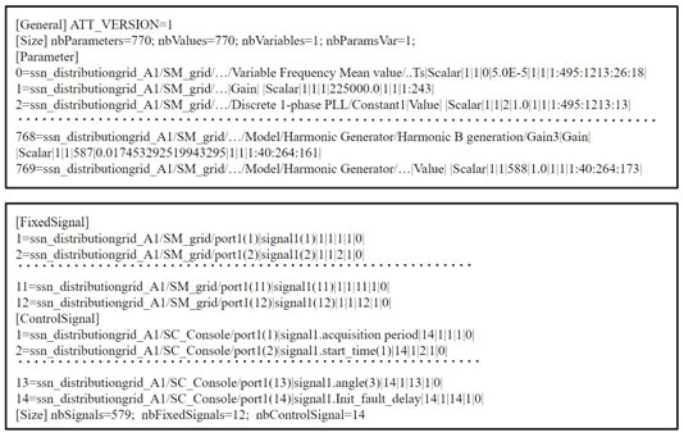

Fig. 5. System parameters and Signals

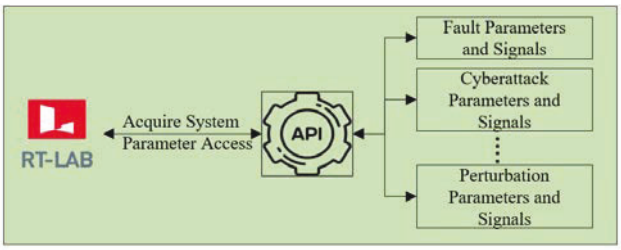

Fig. 6. Proposed Python API automating feature vectors

In this model, we have 770 parameters and 579 signals, of which 12 are fixed and 14 control signals, as shown in Fig. 5. We used the *'SetAcquisitionWrite'* function to save data in RT-LAB and dynamically controlled it by Python API. It provides more flexibility as users can write scripts to set up data logging, start and stop logging at specific times, or even perform custom processing on the data as it's being captured. The data was saved in a .mat file and subsequently preprocessed in MATLAB to elucidate the nature and characteristics of the generated data.

IV. Case Study

This section demonstrates fault datasets generated using our proposed AFEDG framework. We focused on generating fault datasets because they are frequent anomalies and greatly impact system parameters in the power system. We have configured the API to capture the voltage and current values for fault detection and classification. We generated the 11 faults: single line-to-ground faults (SLG), line-to-line faults (LL), double line-to-ground faults (LLG), triple line faults (LLL), and triple line-to-ground faults (LLLG). We implemented fault scenarios using the Monte Carlo simulation method [9] controlled via a Python API. In which we controlled fault type, fault location, fault start time, fault duration, and fault firing angle. We have chosen fault parameters for our simulation based on the studies presented in [10], [11], as detailed in Table II. To better understand the datasets, we plotted time-domain plots showcasing voltage and current variations of generated datasets. We are collecting three-phase voltages and currents of Bus0 and Bus1, including their respective time stamps.

TABLE II
Configuration of Parameters for Fault Datasets Simulation

Parameter	Configuration
Fault Type	(A/B/C)G, AB, AC, BC, (AB/AC/BC)G, ABC, or ABCG
Fault Resistance	0.1, 1, 5, 100
Fault Location	4 Locations
Firing Angle	30°,45°,90°,135°

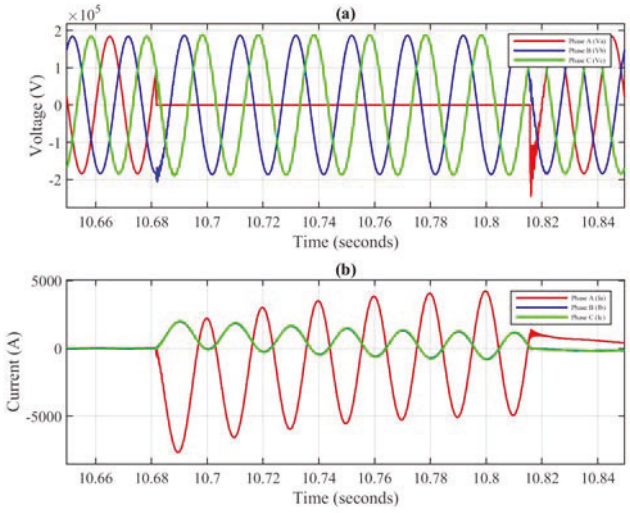

Fig. 7. SLG Fault: Three-Phase (a) Voltages (b) Currents at BUS0

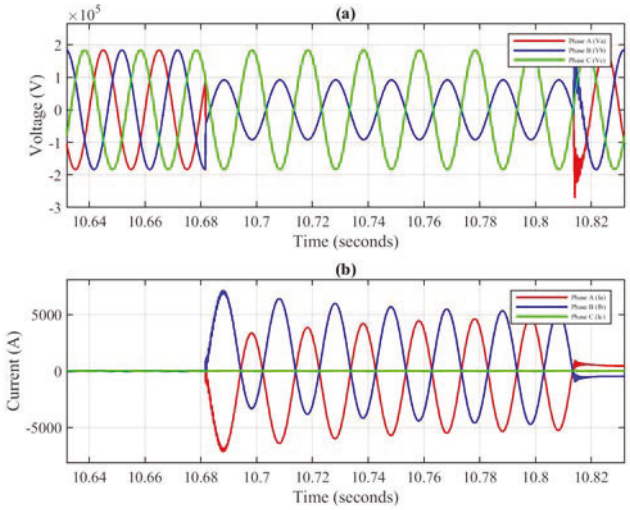

Fig. 8. LL Fault: Three-Phase (a) Voltages (b) Currents at BUS0

Figs. 7-10 illustrates the real-time signals generated datasets corresponding to diverse fault scenarios within the system.

979-8-3503-1360-4/24/$31.00 ©2024 IEEE

These scenarios encompass the full spectrum of power systems: SLG, LL, LLG, and LLLG. The simulation parameters are maintained across all scenarios of all faults at fault location 1 with a fault duration of 0.125 seconds, a fault firing angle of 30 degrees, and a total simulation duration of 5 seconds. In these figures, the fault occurs at a 10.680-second time stamp till 10.805 seconds, maintaining 0.125 seconds as specified. When the fault happens, there is a voltage dip to zero, as shown in part (a) of Figs. 7-10, while there is a drastic change in the current as shown in part (b) Figs. 7-10.

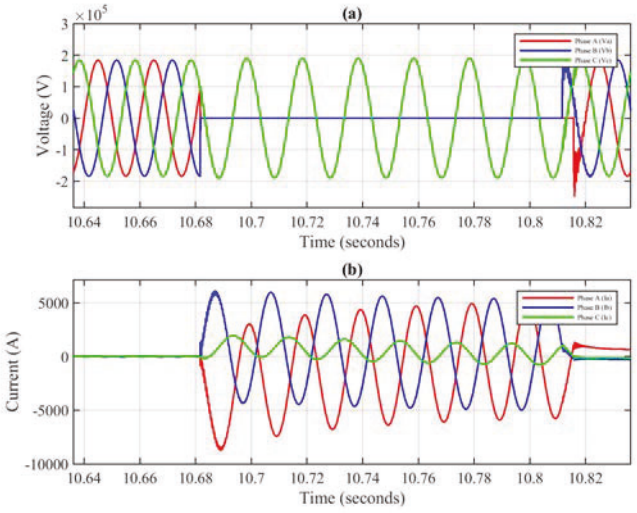

Fig. 9. LLG Fault: Three-Phase (a) Voltages (b) Currents at BUS0

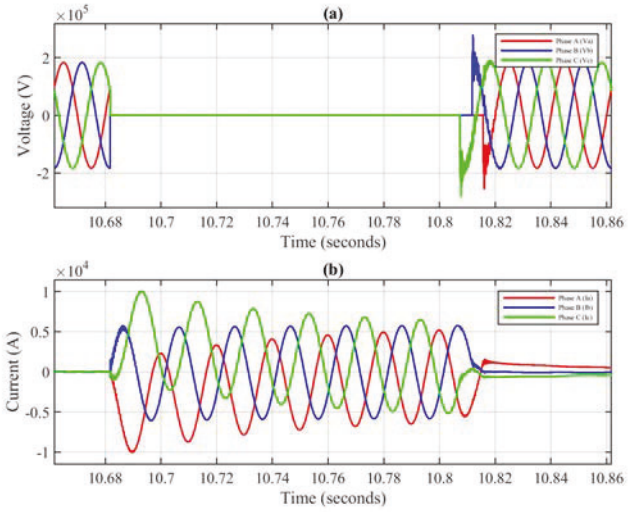

Fig. 10. LLLG Fault: Three-Phase (a) Voltages (b) Currents at BUS0

V. Future Work

Our future research directions are generating datasets for cyberattacks, perturbations, and voltage fluctuations using the proposed framework. Further, we aim to evaluate and validate the performance of ML/DL models using datasets generated by our framework for various grid models.

VI. Conclusion

The paper presented a comprehensive auto feature extraction and data generation framework for smart grid ML and DL-based anomaly detection and classification. We have integrated the proposed Python API with RT-LAB and Simulink to generate large-scale and real-time event-driven data sets. Using the proposed Python API, we controlled the feature vectors and virtual sensors to accurately capture the complex dynamics of various power system events. The dynamic modeling of system parameters by API generates sufficient, unbiased, balanced large-scale datasets for machine learning and deep learning models. We generated 704 fault datasets using the Monte Carlo simulation method with combinations of simulation parameters shown in Table. II. Further, API was compatible with any IEEE power system model, regardless of the number of buses. Overall, the successful validation of the AFEDG provides a robust framework for large-scale event-driven data generation in smart grid applications.

Acknowledgment

This research is funded partly by US NSF Grant # CNS 2105269, US DOE CESER Grant DE-CR000016, and Iowa Energy Center Grant #21-IEC-009.

References

[1] N. Kato, B. Mao, F. Tang, Y. Kawamoto, and J. Liu, "Ten challenges in advancing machine learning technologies toward 6g," *IEEE Wireless Communications*, vol. 27, no. 3, pp. 96–103, 2020.

[2] G. Ravikumar, B. Hyder, and M. Govindarasu, "Next-generation cps testbed-based grid exercise - synthetic grid, attack, and defense modeling," in *2020 Resilience Week (RWS)*, 2020, pp. 92–98.

[3] M. N. Fekri, A. M. Ghosh, and K. Grolinger, "Generating energy data for machine learning with recurrent generative adversarial networks," *Energies*, vol. 13, no. 1, 2020. [Online]. Available: https://www.mdpi.com/1996-1073/13/1/130

[4] C. Zhang, S. R. Kuppannagari, R. Kannan, and V. K. Prasanna, "Generative adversarial network for synthetic time series data generation in smart grids," in *2018 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm)*, 2018, pp. 1–6.

[5] A. Venzke, D. K. Molzahn, and S. Chatzivasileiadis, "Efficient creation of datasets for data-driven power system applications," *Electric Power Systems Research*, vol. 190, p. 106614, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0378779620304181

[6] G. Ravikumar and M. Govindarasu, "Anomaly detection and mitigation for wide-area damping control using machine learning," *IEEE Transactions on Smart Grid*, pp. 1–1, 2020.

[7] G. Ravikumar, B. Hyder, and M. Govindarasu, "Efficient modeling of hil multi-grid system for scalability concurrency in cps security testbed," in *2019 North American Power Symposium (NAPS)*, 2019, pp. 1–6.

[8] C. Dufour, J. Mahseredjian, and J. Belanger, "A combined state-space nodal method for the simulation of power system transients," in *2011 IEEE Power and Energy Society General Meeting*, 2011, pp. 1–1.

[9] J.-N. Paquin, J. Belanger, L. A. Snider, C. Pirolli, and W. Li, "Monte-carlo study on a large-scale power system model in real-time using emegasim," in *2009 IEEE Energy Conversion Congress and Exposition*, 2009, pp. 3194–3202.

[10] J. J. Q. Yu, Y. Hou, A. Y. S. Lam, and V. O. K. Li, "Intelligent fault detection scheme for microgrids with wavelet-based deep neural networks," *IEEE Transactions on Smart Grid*, vol. 10, no. 2, pp. 1694–1703, 2019.

[11] A. Hussain, C.-H. Kim, and S. Admasie, "An intelligent islanding detection of distribution networks with synchronous machine dg using ensemble learning and canonical methods," *IET Generation, Transmission & Distribution*, vol. 15, no. 23, pp. 3242–3255, 2021. [Online]. Available: https://ietresearch.onlinelibrary.wiley.com/doi/abs/10.1049/gtd2.12256