Hardware-in-the-Loop Real-Time Transient Emulation of Large-Scale Renewable Energy Installations Based on Hybrid Machine Learning Modeling

Ruogu Chen, *Student Member, IEEE,* Tianshi Cheng, *Student Member, IEEE,* Ning Lin, *Member, IEEE,* Tian Liang, *Member, IEEE,* and Venkata Dinavahi, *Fellow, IEEE*

***Abstract*—For the integration of large-scale renewable energy resources into power grids, the complex and dynamic behavior of Inverter-Based Resources (IBRs), such as wind farms, photovoltaic (PV) arrays, and battery energy storage systems (BESSs), poses significant challenges. Traditional models often fall short of feasibly simulating these resources at scale. This paper introduces a hybrid machine learning approach, employing Multi-Layer Perceptrons (MLPs) and Gated Recurrent Units (GRUs), to effectively simulate IBRs. The hybrid models combine MLPs and GRUs to capture the transients of IBRs. An extensive dataset, including environmental data, load profiles, and fault instances, was used for training and validation. The source of this dataset was the computational Electromagnetic Transient (EMT) models of IBRs and validated results. A test system was developed to integrate a microgrid comprising batched ML-based IBR modules into a large-scale AC-DC system, which is based on the IEEE 118-bus system. The system is deployed on a Field-Programmable Gate Array (FPGA) board, highlighting the viability of real-time, hardware-accelerated emulations. The results show that the hybrid ML methodology accurately represents large-scale IBRs and predicts transient behaviors in integrated grids, offering crucial insights for the future planning, operation, and control of AC-DC grids, especially those with high renewable energy integration.**

***Index Terms*—Artificial neural networks, battery energy storage, electromagnetic transients, field programmable gate arrays, faster-than-real-time gated recurrent units, hardware-in-the-loop, inverter-based resources, machine learning, multi-layer perceptron, photovoltaic, transient behavior, wind farms**

NOMENCLATURE

ANN	Artificial Neural Network.
BESS	Battery Energy Storage System.
DFIG	Doubly-Fed Induction Generator.
EMT	Electromagnetic Transient.
FF	Flip-Flop.
FPGA	Field-Programmable Gate Array.
FTRT	Faster Than Real-Time.
GRU	Gated Recurrent Unit.
GSC	Grid Side Converter.
HIL	Hardware-in-the-Loop.
IBR	Inverter-Based Resources.
LSTM	Long Short-Term Memory.
LUT	Look UP Table.
MLP	Multi-Layer Perceptron.
MSELoss	Mean Square Error Loss.
PV	Photovoltaic.
RNN	Recurrent Neural Network.
RSC	Rotor Side Converter.
TS	Transient Stability.

This work is supported by the National Science and Engineering Research Council of Canada (NSERC), Mitacs, and RTDS Technologies Inc.

R. Chen, T. Cheng, and V. Dinavahi are with the Department of Electrical and Computer Engineering, University of Alberta, Edmonton, Alberta T6G 2V4, Canada. N. Lin is with Powertech Labs Inc., Surrey, British Columbia, Canada V3W 7R7. T. Liang is with RTDS Technologies Inc. Email: ruogu@ualberta.ca, tcheng1@ualberta.ca, ning3@ualberta.ca, tian.liang@ualberta.ca, dinavahi@ualberta.ca.

I. INTRODUCTION

THE integration of large-scale renewable energy resources into power grids is a crucial aspect of global efforts to mitigate climate change and transition toward sustainable and ecologically conscious energy consumption [1]. Representatives of these resources, including doubly-fed induction generator (DFIG) wind farms, photovoltaic (PV) arrays, and battery energy storage systems (BESSs), collectively termed Inverter-Based Resources (IBRs), are characterized by dynamic and intricate transient behaviors. These behaviors necessitate sophisticated control and operational strategies, particularly concerning transient behaviors resulting from variability in fault occurrences, diverse weather conditions, and fluctuations in load demand. Understanding these transient emulations of large-scale IBRs integrated into power networks is essential for grid planning and operation, and improving grid reliability and resilience.

Transient simulation methods for power systems generally fall into two categories: transient stability (TS) simulation and electromagnetic transient (EMT) simulation. Time-domain EMT simulation, in comparison to TS simulation, can portray transient behaviors down to sub-μs [2], making it a more suitable choice for simulating the transients of IBRs to capture detailed and faster transient behaviors [3], [4]. Traditionally, EMT simulations are conducted by solving discrete-time numerical equations from computational models. However, in an IBR group such as a wind farm, the variability in wind speed across hundreds of turbines can significantly affect the system's overall performance. This variability is particularly pronounced in mountainous regions due to wake and acceleration effects [5]. Large-scale models are essential to account for individual unit variations, but as these traditional models scale up, they encounter significant computational constraints [6].

To address these limitations, this study introduces a hybrid machine learning-based methodology for modeling large-scale IBRs, which circumvents the complex iterative solution of large sets of non-linear equations inherent in traditional EMT models. The proposed machine learning components are based on Artificial Neural Networks (ANNs). Previous work on ANN-based power system equivalent models [7], and the implementation of AI technologies in the long-term steady-state analysis of power systems [8]–[10], although insightful, lacks the ability to perform fast transient predictions for complex, highly non-linear renewable energy sources.

The proposed machine-learning-based IBR models utilize hybrid ANNs, including Multi-Layer Perceptrons (MLPs) and Gated Recurrent Units (GRUs). The choice of neural network type depends on the complexity and time-dependency of different IBR models. The GRU, a variant of the Recurrent Neural Network (RNN) and akin to the Long-Short-Term Memory (LSTM), excels in representing highly nonlinear and time-dependent components [11]. The MLP, a simpler feed-forward ANN, is more suitable for less time-dependent, non-linear models. These hybrid models for IBRs were trained using comprehensive datasets derived from validated simulations and other code-based computational models of IBRs [12], [13]. The datasets include historical weather data, grid inputs and loads, and instances of faults, offering diverse conditions and scenarios for model training and validation. The trained ML-based IBR models were then interfaced with an actual AC grid through microgrids.

Even though TS simulations are based on larger time-steps up to the level of several ms, they are adequate for simulating grid-level behaviors [12] and offer more computational efficiency compared to EMT simulations [14], [15]. Thus, a multi-time-step strategy was adopted to perform EMT-TS hybrid emulation over the ML-based models and the AC grid model, leveraging the advantages of both EMT and TS simulations. This hybrid simulation method has been widely used and has even been realized in commercial tools like PSCAD/EMTDC [16], [17]. However, previous studies of hybrid simulation have never used ML-based models. To enable real-time emulation of the entire AC-DC system, a Field-Programmable Gate Array (FPGA) board was deployed as the hardware accelerator, given its capacity for parallel computations [2]. FPGAs have been widely used as accelerators for neural networks representing EMT models [18]–[21]. However, previous machine-learning research EMT works focused on small-scale systems with small-scale traditional electrical and mechanical components such as motors and converter circuits. The application of FPGAs to more complex models like IBRs remains under exploration.

The main contributions of this paper can be summarized as follows:

- The development and training for hybrid neural network modeling of large-scale renewable energy sources and energy storage systems, while the independent characteristics of individual resources are still preserved.
- The introduction of a novel real-time hardware-in-the-loop (HIL) transient emulation of the EMT-TS hybrid large-scale ML-based renewable energy installations on the Xilinx® UltraScale+™ FPGA by efficient hardware implementation of the ML-based IBR models. The hardware-accelerated test system achieved an equivalent faster-than-real-time (FTRT) ratio of 3.33, demonstrating substantial improvements in computational efficiency and response time over traditional simulation frameworks. This capability is pivotal for the future planning and operational strategies of AC-DC grids with large-scale renewable installations.

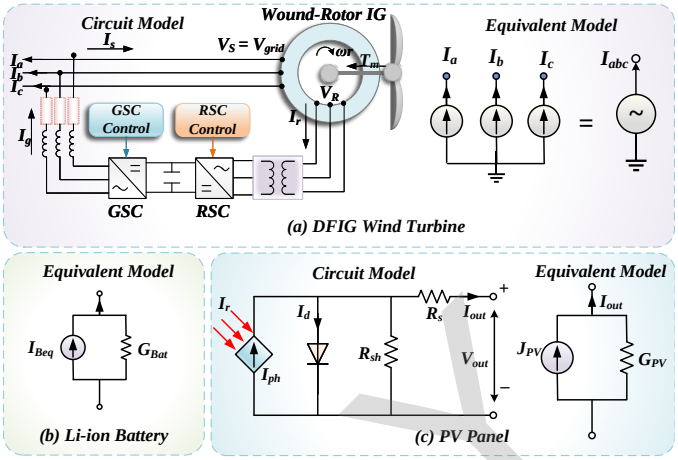

Fig. 1. Traditional computational IBR models: (a) DFIG wind turbine, (b) Li-ion battery, and (c) PV panel.

The paper is organized as follows: Section II details the traditional IBR models used as ML training targets and the hybrid ANNs employed for different renewable energy sources, including the detailed training process. Section III describes the setup of the HIL transient emulation system and the hardware implementation of the ML-based IBR models integrated with the AC-DC grids. Section IV presents test scenarios and results. Section V offers the conclusion.

II. Machine Learning Based Modeling of Renewable Energy Sources

A. Computational Models for Renewable Energy Sources

This section presents the traditional models used in this study that the machine learning model will learn to behave. These models are based on research works validated by commercial simulation software such as PSCAD/EMTDC.

1) Wind Turbines: The wind turbine model utilized in this study is the DFIG wind turbine from MATLAB Simulink [22]. Fig. 1(a) shows the fundamental structure of the DFIG model. This model comprises a three-phase wound rotor induction generator, with stator windings directly connected to the grid. The rotor windings are linked to the grid via a back-to-back voltage source converter. Independent control of the grid side converter (GSC) and rotor side converter (RSC) allows for separate regulation of active and reactive power, thereby optimizing the extraction of wind energy under varying wind intensities [23]. During operation, the generator speed dynamically adjusts to changes in wind speed, contributing to improved power quality and grid compatibility.

2) Battery Storage System: The battery model is based on [12], which can be represented by a Thévenin equivalent representation, consisting of a voltage source V_{Bat} and an internal resistance Z_B. The essential components of V_{Bat} include: The base voltage, E_0. Polarization effects, represented as E_{pol}. The exponential voltage, E_{exp}. Voltages due to charging, E_{chg}, and discharging, E_{dsc}, processes.

Mathematically, V_{Bat} is expressed as:

$$V_{\text{Bat}} = E_0 + E_{\text{pol}} + E_{\text{exp}} + S_c E_{\text{chg}} + (1 - S_c) E_{\text{dsc}}, \quad (1)$$

where S_c signifies the battery's charging status, equating to 1 during the charging phase.

However, in the context of EMT simulation, the nodal voltages are to be solved rather than the mesh currents. Therefore, it is necessary to convert the Thévenin equivalent circuit to its Norton equivalent circuit. Assuming the internal resistances of all batteries under study are converted into the conductance and then grouped as a vector $\mathbf{G}_{\text{B}} = [G_{\text{B1}}, G_{\text{B2}}, \ldots]$, the current contribution of the batteries can be expressed as

$$\mathbf{I}_{\text{B,eq}} = V_{\text{Bat}} \circ \mathbf{G}_{\text{B}}, \quad (2)$$

where the vector $\mathbf{I}_{\text{B,eq}} = [I_{\text{B,eq1}}, I_{\text{B,eq2}} \ldots]$.

The Norton equivalent circuit for the battery model is shown in Fig. 1(b).

3) Solar PV Cells: The PV cell model is adopted from [13]. A large-scale PV array is comprised of strings of PV panels connected in parallel, while a single string consists of series-connected single PV units. Fig. 1(c) shows the equivalent circuit of a single-diode PV unit, where the equivalent current source is represented by:

$$I_{ph} = \frac{S_{irr}}{S^*_{irr}} \cdot I^*_{ph} \left(1 + \alpha_T \cdot (T_K - T^*_K)\right), \quad (3)$$

which has variables with the superscription $*$ as references, S_{irr} as the solar irradiance, α_T the temperature coefficient, and T_K the absolute temperature. In addition, the model is also comprised of the shunt and series resistors R_p and R_s, respectively.

With all of its components represented by current sources and conductors or resistors, the PV unit can be converted into the most concise two-node Norton equivalent circuit, as shown in Fig. 1(c).

B. Hybrid ANN Modeling for Renewable Energy Sources

1) Gated Recurrent Unit: The Gated Recurrent Unit was introduced in [24] and has been known for its efficiency in capturing temporal dynamics. In a standard GRU cell, the computations for the gates and the resulting hidden state update are encapsulated by the following equations:

$$z_t = \sigma(W^z_{ih} x_t + b^z_{ih} + W^z_{hh} h_{(t-1)} + b^z_{hh}), \quad (4)$$
$$r_t = \sigma(W^r_{ih} x_t + b^r_{ih} + W^r_{hh} h_{(t-1)} + b^r_{hh}), \quad (5)$$
$$n_t = \tanh(W^n_{ih} x_t + b^n_{ih} + r_t \circ (W^n_{hh} h_{(t-1)} + b^n_{hh})), \quad (6)$$
$$h_t = (1 - z_t) \circ n_t + z_t \circ h_{(t-1)}. \quad (7)$$

In these expressions, z_t denotes the output of the update gate, which determines the extent to which the GRU cell's state is updated. The output of the reset gate, r_t, modulates the influence of the previous state by allowing the cell to selectively forget certain aspects. The candidate hidden state, n_t, is calculated as a combination of the current input and the previous state, with the reset gate's output regulating this integration. The final output of the GRU cell at time t, h_t, emerges as a mixture of the previous state and the candidate state, with the update gate's output dictating the balance. The terms W_{ih} and W_{hh} represent the input-to-hidden and hidden-to-hidden weight matrices, respectively, while b_{ih} and b_{hh} are their respective bias vectors.

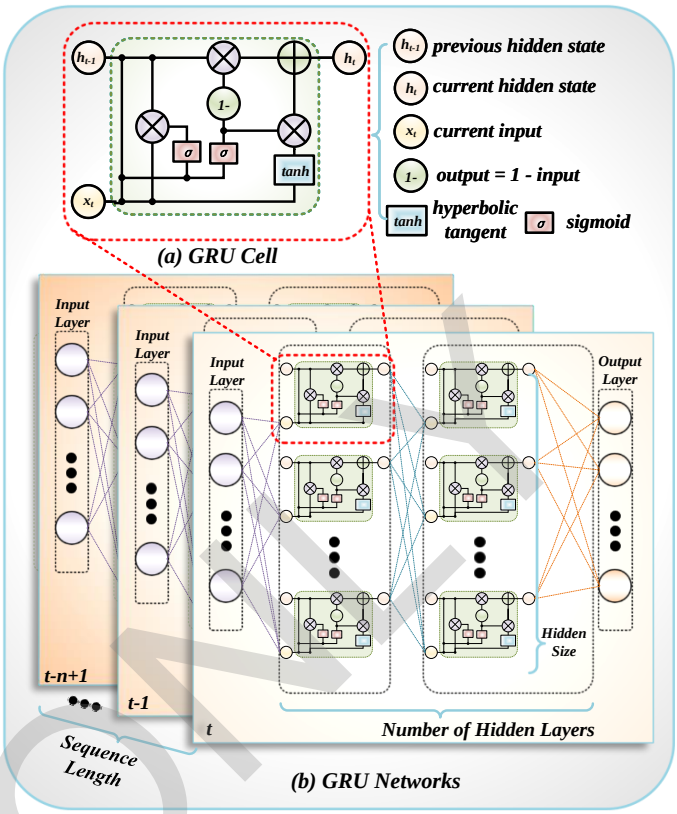

Fig. 2. GRU structures: (a) cell, and (b) network.

The calculation process is visualized in the simplified GRU cell diagram shown in Fig. 2(a). A typical GRU network is comprised of a large amount of individual GRU cells, which is illustrated in Fig. 2(b).

2) Multi-Layer Perceptron: The MLP is a type of artificial neural network commonly used for various machine learning tasks [25]. It consists of multiple layers of interconnected neurons, each layer contributing to the transformation of input data to produce meaningful outputs. The output of a neuron can be computed using the following equations:

$$z_j = \sigma\left(\sum_{i=1}^{n} w_{ji} x_i + b_j\right), y_k \quad = \sigma\left(\sum_{j=1}^{m} v_{kj} z_j + c_k\right), \quad (8)$$

where z_j is the activation of neuron j in a hidden layer, y_k is the output of neuron k in the output layer, σ is an

This article has been accepted for publication in IEEE Journal of Emerging and Selected Topics in Industrial Electronics. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/JESTIE.2024.3434364

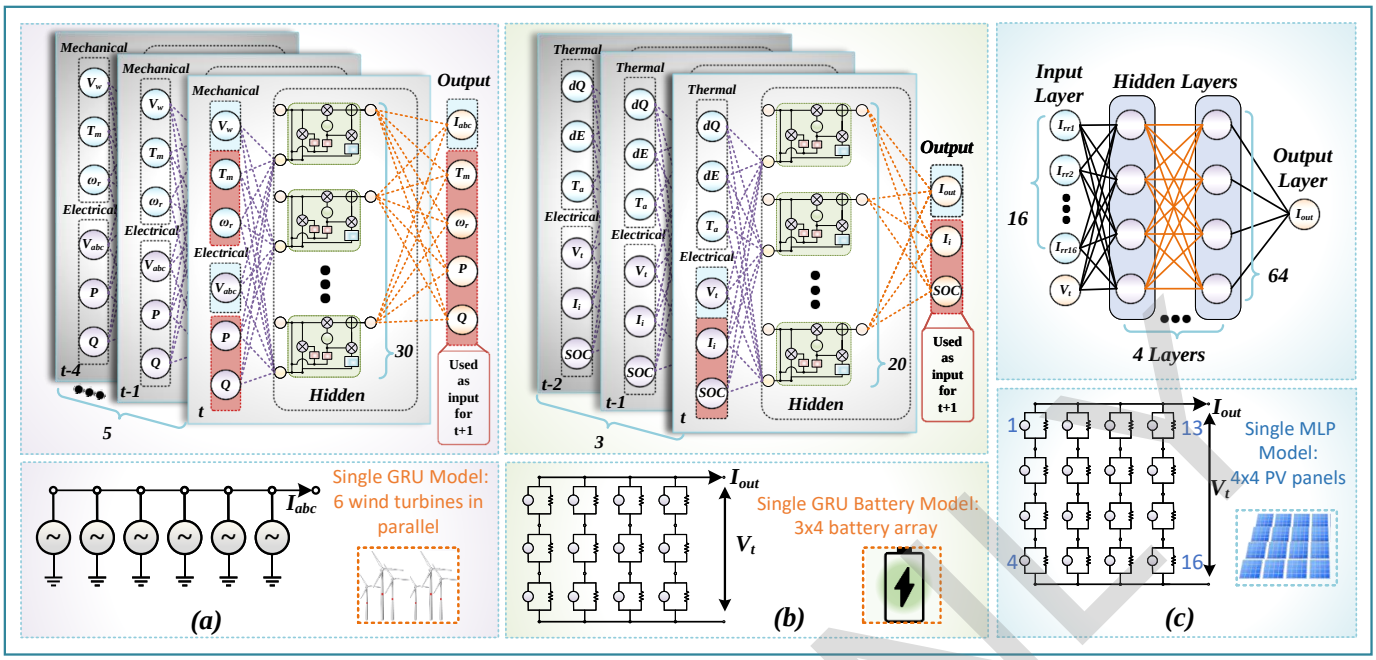

Fig. 3. ML-based IBR neural networks (top) and their equivalent circuits (bottom): (a) GRU wind turbine model, (b) GRU battery model, and (c) MLP PV model.

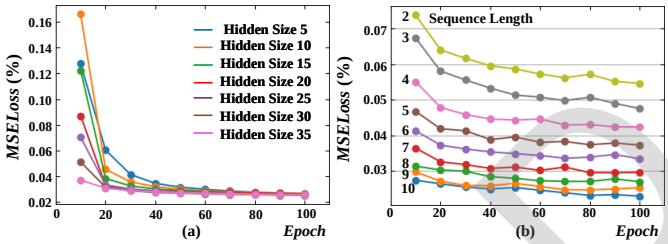

Fig. 4. DFIG parameter tuning: (a) training error under different hidden sizes, and (b) MSELoss vs training epochs, under different sequence lengths.

activation function (e.g., $sigmoid$ or $ReLU$), w_{ji} are the weights connecting input nodes to hidden layer neurons, v_{kj} are the weights connecting hidden layer neurons to output nodes, and b_j and c_k are the bias terms for the hidden and output layers, respectively. Since MLP has already been widely applied in various research aspects for many years, the detailed network-level process is omitted.

3) GRU Based DFIG Wind Turbine Model: The DFIG wind turbine system is a complex system, divided into mechanical and electrical components to aid the GRU model in understanding the wind turbine's operational principles. The mechanical component includes wind speed, V_w, DFIG torque, T_m, and rotor speed, w_r. The electrical component comprises the three-phase AC voltage from the grid, V_{abc}, identical to the DFIG stator voltage, V_s, and the output active and reactive power, P and Q. The trained GRU model digitally represents six DFIG wind turbines in parallel, as shown in the bottom part of Fig. 3(a), offering computational efficiency for large-scale wind farm deployment without compromising detail.

The training data was generated from offline simulation in Simulink. Since GRU uses a sequence of time-series data as inputs, the continuous nature of input signals cannot be violated when generating the data. In this case, each set of parameters produces a continuous data series within a single Monte Carlo test execution, rather than adjusting parameters during the run time. The wind speed data forms a normal distribution with a mean of 10 m/s and a standard deviation of 3 m/s, which covers both the normal and extreme working conditions of wind farms [5]. In addition to the wind speed variation, fault scenarios are also simulated. An external symmetrical fault is added to the training set. The short circuit resistance is 0.01Ω and the fault duration is 100 milliseconds. The fault was applied at the grid connection port of the wind farm and multiple instances of this fault were randomly inserted into the training set, which takes up 5% of the dataset.

To facilitate EMT simulation, the GRU-based DFIG wind turbine was trained as a three-phase current source to output the three-phase grid current I_{abc}. In order to balance the individual characteristics and computational efficiency, six parallel-connected turbines are seen as a bundle for training the ML-based model, which is depicted in the lower part of Fig. 3(a). GRU networks update input data x_t at each time-step. However, internal turbine parameters like torque T_m and rotor speed w_r are not directly measurable in real-time emulation. To enable continuous emulation, internal features are recursively processed, serving as inputs to the GRU while also being generated as outputs for subsequent time-steps. This is illustrated in the top part of Fig. 3(a), where the features in the red box indicate recursive features.

The model architecture was carefully selected to balance computational efficiency and performance. The MSELoss (Mean Squared Error Loss), which calculates the mean of the squares of the differences between predicted and actual values, is widely used to measure the performance of a regression

model. The formula for MSELoss is given by:

$$\text{MSELoss} = \frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2 \tag{9}$$

where n is the number of samples or data points, $\hat{y}_i$ is the predicted value for the i^{th} sample, and y_i is the actual value for the i^{th} sample.

As shown in Fig. 4(a), for varying hidden sizes, the MSELoss converges to 0.02% after 100 epochs. A hidden size of 30 was chosen to optimize resource usage. Fig. 4(b) shows that MSELoss decreases with increasing sequence length for a constant hidden size. However, a longer sequence also indicates the storage of more historical data, which requires more resources, and error reduction is less significant for lengths beyond five. Therefore, the DFIG wind turbine GRU model, shown in Fig. 3(a), consists of a single GRU hidden layer with a hidden size of 30 and a sequence length of 5.

Challenges like overfitting [26] and gradient vanishing/exploding [27] may arise during training. To address these, the $ReLU$ activation function was employed, and a dropout rate of 0.2 was implemented, giving each neuron a 20% chance of being excluded in each training iteration to ensure model generalization. The training epoch number is set to 1000 to further reduce the error. Other hyperparameters, such as learning rate and batch size, were tuned to 0.001 and 1000, respectively, for optimal training outcomes. The Adaptive Moment Estimation (Adam) optimizer [28] was used to minimize training loss.

The trained model behavior is validated by two symmetrical faults with 80 and 200 ms of fault duration, which is 20% less than and 100% more than those in the training set. The results are shown in Fig. 5(a) and (c). The trained model behavior under varying wind speeds was also tested. The wind speed validation set includes two step changes, one from 8 m/s to 13 m/s, the other from 10 m/s to 5 m/s, which never appeared in the training set. Although this step change is not realistic, it can fully demonstrate the transient performance of the trained model. The results are shown in Fig. 5(b) and (d). It can be observed that the GRU-based model can accurately capture the transient behaviors of the traditional model even under unprecedented scenarios when being trained.

4) GRU Based Li-ion Battery Storage Model: The Li-ion battery model, being a strongly time-dependent system, is aptly modeled using a GRU network. The model's features are categorized into electrical and thermal components. The electrical part encompasses the battery's output voltage V_t, output current I_{out}, input current I_i, and the state of charge SOC. The thermal aspect includes the ambient temperature T_a, the rate of change of the battery's voltage with temperature $\frac{dE}{dT}$, and the capacity's temperature dependency $\frac{dQ}{dT}$. The GRU-based battery model represents a 3x4 array, as shown in the bottom part of Fig. 3(b). Similar to the DFIG model training, SOC and I_i are recursive features, fed back into the model as depicted in the top part of Fig. 3(b).

The training methodology for the battery model mirrors that of the DFIG wind turbine model. The finalized GRU-based battery model structure comprises a single layer with a hidden size of 20 and a sequence length of 3. This architecture is

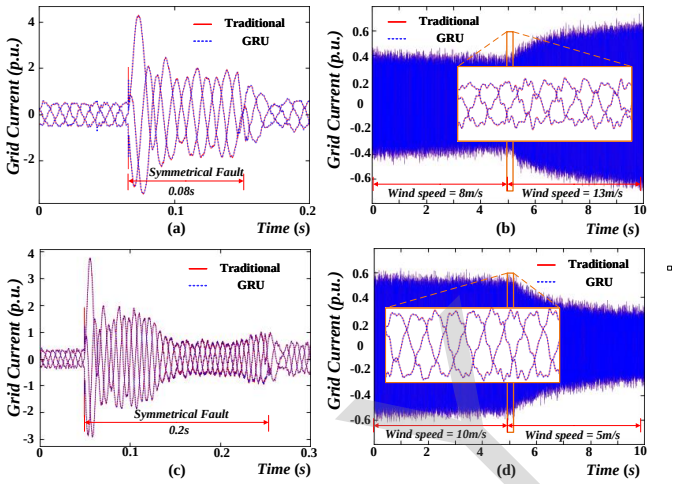

Fig. 5. DFIG training results on output current: (a) three-phase-to-ground fault with 80 ms duration, (b) varying wind speed from 8 m/s to 13 m/s, (c) three-phase-to-ground fault with 200 ms duration, and (d) varying wind speed from 10 m/s to 5 m/s.

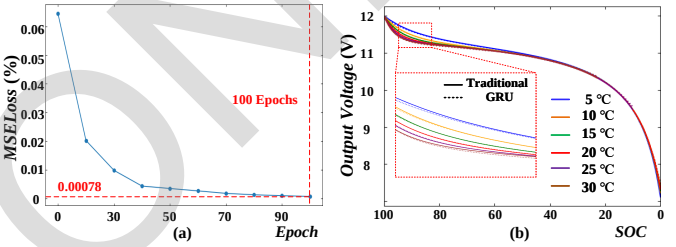

Fig. 6. Battery training results: (a) training error, and (b) charging characteristics under different temperatures.

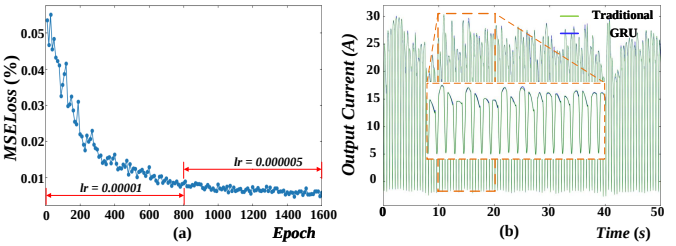

Fig. 7. PV training results: (a) training error under variable learning-rate training, and (b) output current profile under random irradiance.

illustrated in the top part of Fig. 3(b). Training losses are detailed in Fig. 6(a), where the MSELoss converges after 100 epochs, reaching a final loss of approximately 0.00078%. Fig. 6(b) displays the discharging characteristics under varying ambient temperatures, showcasing the high accuracy of the ML-based model.

5) MLP Based Solar PV Cell Model: Given the specific characteristics of PV arrays, where the current profile is not profoundly time-dependent, MLP is deemed a preferable choice. The rationale for this selection is that MLPs excel in capturing static nonlinear relationships between inputs and outputs, without emphasizing time-series or sequential data. This attribute makes them particularly well-suited for modeling the behavior of PV arrays.

The architecture of the MLP model is designed to represent

the $I-V$ characteristics of a 4x4 PV panel array, as shown in the bottom part of Fig. 3(c). This MLP structure, depicted in the top part of Fig. 3(c), comprises four hidden layers, each with a hidden size of 64. The primary inputs to the model include the terminal voltage V_t and the irradiance I_{rr} incident on each panel in the 4x4 array, while the output predicts the panel's output current I_{out}.

For the training process, the training dataset was generated by an optimized C++ simulation program based on the traditional PV model [13]. The Monte Carlo distribution was used to randomly generate the data from the traditional PV array simulation for training. The generated irradiance data forms a normal distribution with a mean of 1000 W/m^2 and a standard deviation of 300 W/m^2, simulating real-world specific irradiance conditions. The port voltage used for training is varied linearly from zero to the maximum operational voltage. A variable-learning-rate technique was employed, initially set at 0.00001 and reduced to 0.000005 to refine the training, as shown in Fig. 7(a). The high prediction accuracy of the output current profile under random irradiance conditions is illustrated in Fig. 7(b). The validation dataset used in Fig. 7(b) was also randomly generated and never appeared in the training dataset.

III. Implementation For ML-Based HIL Real-Time Transient Emulation of Renewable Energy Sources

A. Test System Topology

The test bench system is built upon real AC networks based on the IEEE 118-bus system. An IBR microgrid, comprising one wind farm, one solar farm, and one BESS station, is integrated. The detailed topology of the IBR microgrid is shown in Fig. 8(a). To enhance the complexity of the test case and fully utilize the FPGA resources, the wind farm includes 30 batches of trained GRU-based wind turbine bundles. As mentioned in Section.II.B.3), each of the GRU-based DFIG models contains six turbines, therefore, 30 such models represent totaling $30 \times 6 = 180$ turbines. Each 6-turbine string has a rated output power of 9 MW and thirty such strings are connected in parallel, yielding $30 \times 9 = 270$ MW of output power. The solar farm comprises 20 batches of 4x4 MLP-based PV panel arrays, totaling $4 \times 4 \times 20 = 320$ panels. In the solar farm, a solar string contains four batches of 4x4 PV arrays connected in series, each of such string is rated at $4 \times 3.125 = 12.5$ kW, and five of such series are connected in parallel, yielding a total output power of $5 \times 12.5 = 62.5$ kW. The BESS station, utilizing the ML-based model, is scaled to 50 MW. This IBR microgrid is connected to the 118-bus system at Bus-25 via a 25kV/138kV transformer, as displayed in Fig. 8(b).

In large-scale power systems like the IEEE 118-bus system, EMT-scale real-time simulation would be significantly expensive in terms of hardware resources and latency. Alternatively, TS simulation with a time-step of up to several milliseconds is tolerable in showing the system-level dynamics [12]. Therefore, in this study, the time-step for the 118-bus system is set to 5 ms to balance computational efficiency and system dynamic display. However, the ML-based IBR models are EMT models trained under 50 μs time-steps. To avoid time conflicts during simulation, a multi-time-step test bench system is formed. In other words, the equivalent time-step for the entire system equals the largest time-step of the sub-modules, which is 5 ms in this study. For each equivalent time-step calculation, the hybrid ML-based IBR models calculate for $\frac{5\times10^{-3}}{50\times10^{-6}} = 100$ steps in their own time scales. The data exchange rate between the ML models and the AC grid must also follow this rate.

B. Hardware Platform

The hardware platform used in this study is showcased in Fig. 8(c). The Xilinx® VCU118 board, equipped with the UltraScale+™ XCVU9P FPGA, was selected for its robustness in emulating complex systems. The VCU118 board provides an abundance of resources, including 4320 Block RAMs (BRAMs), 6840 DSP slices, 2364K flip-flops (FFs), and 1182K look-up-tables (LUTs), enabling the implementation of a sophisticated system.

Table I details the primary hardware resource utilization and latencies of both the individual renewable energy modules and the entire test bench system. Latencies are measured in μs and derived from the FPGA clock cycles, each set at 10 ns. Given that the ML-based IBR models are equivalent to their EMT counterparts with designed 50μs time-steps, the latency report suggests an FTRT emulation of the test bench system on this platform. The FTRT ratio for the 118-bus system is calculated as $\frac{5ms}{30\mu s} = 166.67$, while for the ML-based IBR models, the ratios are as follows: GRU-based Wind Farm: $\frac{50\mu s}{15\mu s} = 3.33$, GRU-based Battery: $\frac{50\mu s}{10\mu s} = 5$, MLP-based PV: $\frac{50\mu s}{6\mu s} = 8.33$.

TABLE I
Hardware Resource Utilization and Latencies

Model	BRAM	DSP	FF	LUT	Lat. (μs)
Individual ML-based IBR models					
GRU Wind Turbine	-	1.3%	0.42%	0.9%	15
GRU Battery	-	1.3%	0.42%	0.9%	10
MLP PV	-	2%	0.37%	0.9%	6
Test Bench System					
GRU Wind Farm	-	40%	12.6%	27%	15
GRU BESS	-	1.3%	0.42%	0.9%	10
MLP Solar Farm	-	40%	7.4%	18%	6
IEEE 118 System	-	12%	4%	15.4%	30
Total	-	93.3%	24.42%	61.3%	-
Available	4320	6840	2.36M	1.18M	-

C. FPGA Implementation of Hybrid ANN Models

The translation of machine learning (ML) models from high-level programming environments to FPGA platforms requires a series of methodical adaptations and optimizations to capitalize on the parallel processing capabilities of FPGAs. This study has employed a systematic approach to port the pre-trained hybrid ML-based models introduced in Section II from PyTorch frameworks to an FPGA-synthesizable C++ environment.

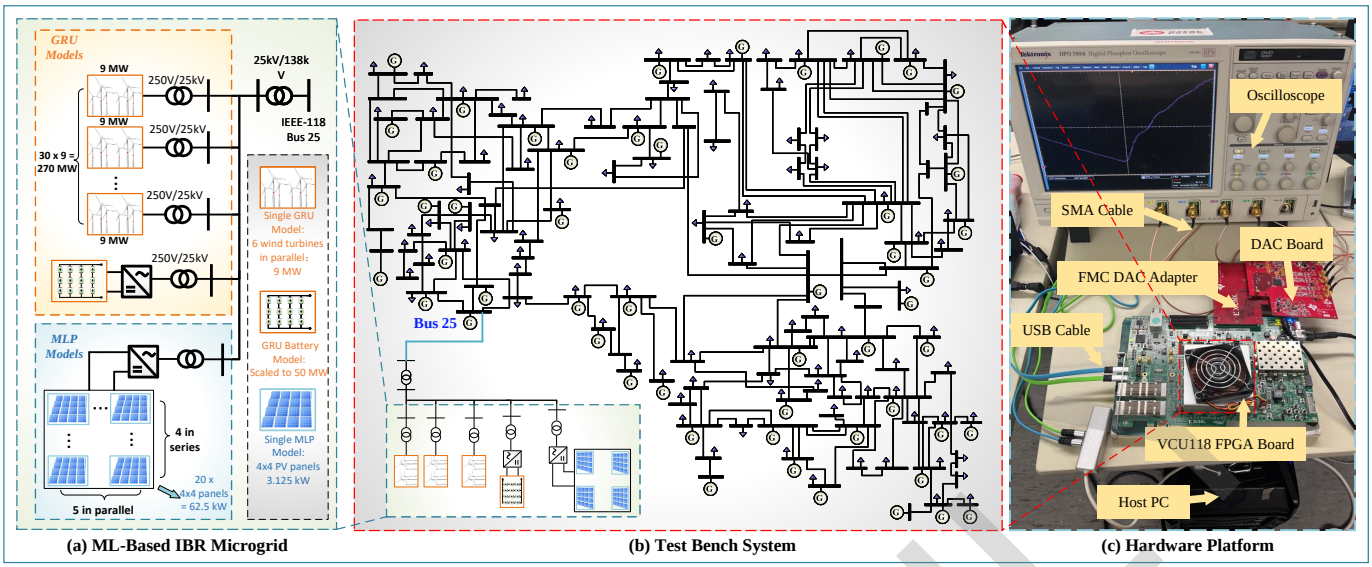

Fig. 8. (a) ML-based IBR microgrid; (b) IEEE 118-Bus system; and (c) hardware platform.

1) Model Adaptation and Quantization: Initially, the PyTorch-trained models were in float32 precision, which is both resource and time-consuming. The quantization process aims to convert the original float-point precision to fixed-point representation to meet the memory and computation efficiency requirements. Two distinct quantization approaches were adopted in this study: dynamic and static [29].

Dynamic quantization involves quantizing the weights of the model at rest and dynamically quantizing the activations at runtime. This method is particularly useful for models like GRU, where the recurrent nature of the computations means that the activations change dynamically with each input sequence.

$$Q_{dynamic}(x) = round\left(\frac{x}{\Delta_{dynamic}}\right) + Z_{dynamic}, \quad (10)$$

where x is the input value to be quantized, $\Delta_{dynamic}$ is the scale factor, dynamically calculated based on the range or distribution of activation data during runtime. This factor scales the input value to match the range of the fixed-point format. $Z_{dynamic}$ is the zero point in dynamic quantization, representing the fixed-point value that corresponds to the real number zero. It is computed dynamically, similar to $\Delta_{dynamic}$, to best fit the data during inference.

On the other hand, static quantization involves quantizing both the weights and activations of the model ahead of time, typically after the model has been trained. This method is more suited for feedforward networks like MLPs.

$$Q_{static}(x) = round\left(\frac{x}{\Delta_{static}}\right) + Z_{static}, \quad (11)$$

Similar to dynamic quantization, x is the input value to be quantized, Δ_{static} is the scale factor, and Z_{static} is the zero point. The term *static* denotes the fixed nature of quantization parameters (Δ_{static} and Z_{static}) once they are determined.

Using the GRU-based DFIG model as an example, the comparison of the resource utilization on a Xilinx® VCU118 FPGA board is shown in Fig. 9(a) and the model prediction output accuracy is shown in Fig. 9(b). It can be observed that the quantization process significantly decreases the hardware utilization of the ML-based models while keeping a considerably high accuracy. The dataset used here is the same validating set used in the training process introduced in Section II, where the wind speed changes from 8 m/s to 13 m/s at 5 seconds. The highest discrepancy point is even smaller than 0.01%. The other two models for the battery and PV panels have similar performances.

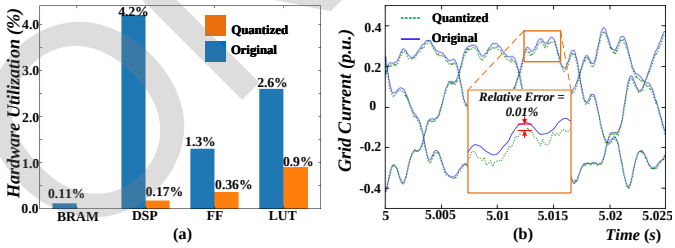

Fig. 9. Quantized model VS original model: (a) resource utilization, and (b) model accuracy.

2) FPGA-Specific Optimizations: Following the quantization process, the models were meticulously translated into C++ code based on the underlying mathematical principles of GRU and MLP as depicted in the aforementioned formulas. After comparison, the translated models retain the complete fidelity of the original models in PyTorch. Further FPGA-specific optimizations were performed in Vitis HLS to ensure efficient FPGA implementation.

The first crucial step was to address the representation of data types within the model, particularly accommodating the range of int8 types used in the quantized model. To achieve this, the `ap_int<8>` data type from the High-Level Synthesis (HLS) library was used.

The next point to be optimized is the activation functions

in neural networks. The activation functions used in this study are the $sigmoid$ function and $tanh$ function. One of the most straightforward and effective optimization methods for activation functions is to use the LUT (Look Up Table) instead of the default functions provided by the HLS library. The LUT method basically pre-calculates an input-output table of the target function using discrete input data in a certain range. The appropriate range for the LUTs was determined based on the potential input values the neural network might encounter during operation, while the resolution was chosen to balance the trade-off between accuracy and memory consumption. For the $sigmoid$ function, the range was set from -10 to 10 while for the $tanh$ function, the range was set from -6 to 6, which are spans that effectively capture the significant transitions of these functions. The resolution was selected with a finer granularity, with a step size of 0.001, thereby ensuring a high level of precision in the function approximations.

Additionally, the manipulation of for-loops is another significant step in FPGA-specified optimization. The two most commonly used optimization methods for for-loops are unrolling and pipelining [30].To balance the hardware resources usage and latencies, the for loops in the hybrid ML models are fully pipelined.

D. Interfacing FPGA-Implemented ML-based Models with AC Grid Emulation

The interfacing of FPGA-implemented machine learning models with the AC grid emulation is crucial for a seamless EMT-TS hybrid environment. The primary challenge stems from the temporal resolution disparity: the EMT-based wind farm and photovoltaic array models operate on a 50 µs time-step, while the 118-bus system's transient stability model uses a $5ms$ time-step. Moreover, both the hybrid ML models and the AC grid incur individual synthesis latencies on the FPGA, necessitating an additional data synchronization mechanism beyond the original multi-time-step ratio of 100.

This synchronization process, illustrated in Fig. 10, employs a buffering mechanism and control logic within the FPGA design. Each ML model independently computes at its intrinsic rate, storing outputs in buffers. A global clock governs the control logic, ensuring timely data exchange between the ML models and the AC grid model. With a multi-time-step ratio of 100, the AC grid module performs one calculation per system iteration, while the three ML-based modules complete 100 calculations, storing results in buffers. The equivalent system latency is thus $100 \times 15 = 1500\mu s$, dominated by the wind farm module with the highest latency. Data synchronization and exchange across the test bench system occur after this latency period. The input and out features of the ML-based IBR modules have been demonstrated in Section II. Some of the external features are to be exchanged with the AC grid, such as the input voltages. These features are directly read from the emulation bus data in the IEEE 118-bus system model during the data exchange stage. Some other external features including the wind speed, irradiance, and temperature are manually set according to different designs of the test scenarios, which will be introduced in the later section. Because of the

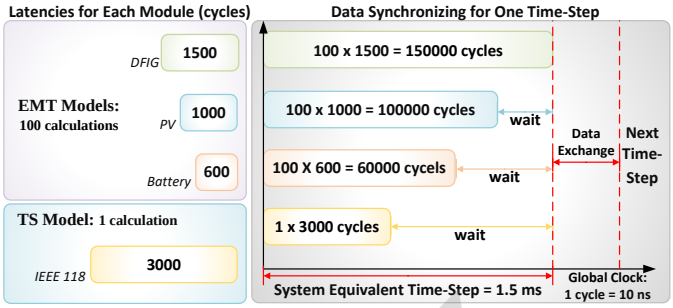

Fig. 10. FPGA data synchronization.

multi-time-step ratio of 100 between the ML-based models and the grid model, when the emulation is running, the bus data from the AC grid will be updated 100 times slower than the ML-based modules. That means those input features that are measured from the bus will be constant between two data exchange stages. Accordingly, during this period, the ML-based models keep iterating over those recursive features to show transient behaviors.

IV. Test Scenarios and Results

The FTRT multi-time-step HIL emulation is performed on the FPGA-based test bench system introduced in Section III. The test scenarios are mainly setting the previously mentioned ambient input features including wind speed, irradiance, and temperature to test the functionality and accuracy of the hybrid ML-based IBR modules under variable working conditions. The test cases mainly focus on the varying wind speed conditions for the wind farm model and the partial shading situation for the PV model. The testing results are compared with offline simulations using traditional computational models introduced in Section II. A.

A. Scenario 1: Step Change on Wind Speed

In this test case, the rated wind speed is set at 15 m/s. Given the variability of wind speeds in real-world scenarios, the turbines in the test wind farm are categorized into three groups. Each group is subjected to distinct wind speed inputs. The wind speed for each group remains constant for the first five seconds and then undergoes a step change to another constant value for the subsequent five seconds. While sudden wind speed transitions are atypical in natural environments, this methodology aptly demonstrates the transient dynamics of the ML-based wind turbine models. The respective wind speed settings for the groups are as follows: Group 1 experiences a change from 8 m/s to 13 m/s, Group 2 from 15 m/s to 10 m/s, and Group 3 from 10 m/s to 5 m/s, covering a broad spectrum of operational conditions for the turbines.

Fig. 11 illustrates that despite the step changes in wind speed, the alterations in rotor speed and active power output are not instantaneous but gradual. This gradual response is indicative of the time required by the DFIG wind turbine's motor to modify the rotor's kinetic energy and stabilize at a new steady state. The findings validate the performance of the ML-based wind farm model in comparison to traditional

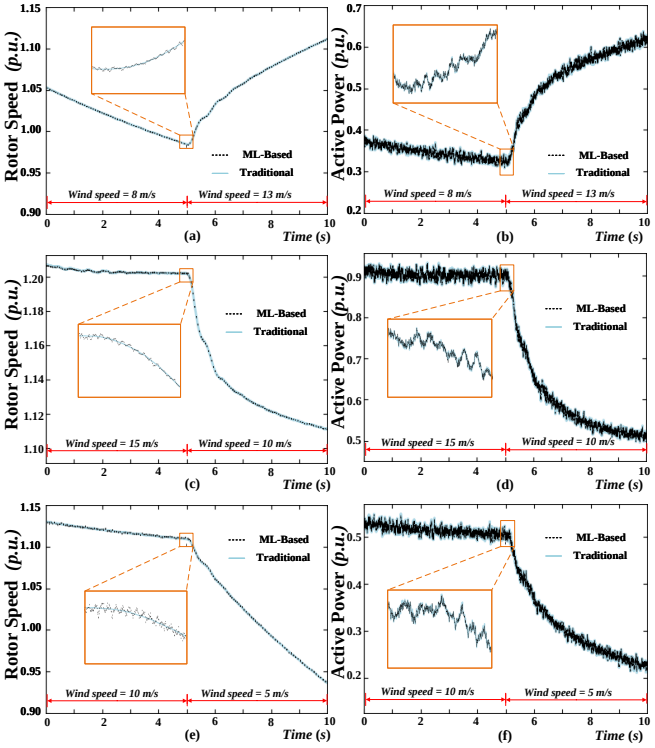

Fig. 11. Emulation results comparison of Scenario 1 between traditional and ML-based wind farm model under different wind speed conditions: (a) rotor speed for Group 1, (b) active power for Group 1, (c) rotor speed for Group 2, (d) active power for Group 2, (e) rotor speed for Group 3, (f) active power for Group 3.

computational models. Slight relative errors can be noted near the 5-second interval due to the sudden change in wind speed, yet these discrepancies are minimal (below 1%) and do not significantly impact the overall emulation fidelity.

B. Scenario 2: Partially Shaded on PV Panels

At the onset of the emulation, all solar panels were set to an irradiance level of 1000 W/m^2. However, at the 1-second mark, the irradiance levels for panels S1 and S16 were adjusted to 100 W/m^2 and 200 W/m^2, respectively to simulate a partially shaded scenario for the solar farm. Given the serial connection of the 4x4 solar array panels and the fact that S1 and S16 are located in different columns of the 4x4 PV array, it was expected that the overall power output would reduce to roughly half of its initial value. This is due to the characteristic of serial-connected panels, where the output is constrained by the panel with the minimum output current.

The results, illustrated in Fig. 12, reveal that the solar array maintained high accuracy with the standard irradiance inputs, exhibiting only a 0.2% relative error. In the partial shading condition, the output of the MLP-modeled solar array decreased from 62.5 kW to 31.25 kW. This change reflects a 4% relative error compared to the traditional model output. As introduced in Section II, the MLP-based PV model training data adhered to a normal distribution centered around 1000 W/m^2. Despite the elevated errors observed in the edge regions under partial shading, the output waveform maintained

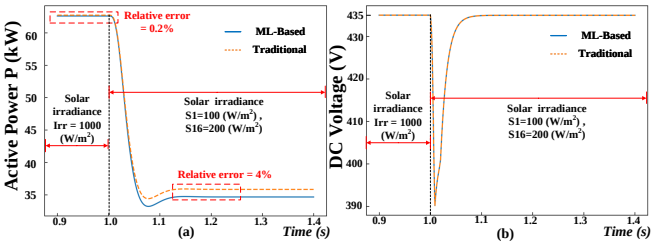

Fig. 12. Emulation results comparison of Scenario 2 between traditional and ML-based solar farm model under partially shaded conditions: (a) active Power, and (b) DC bus voltage of solar farm inverter.

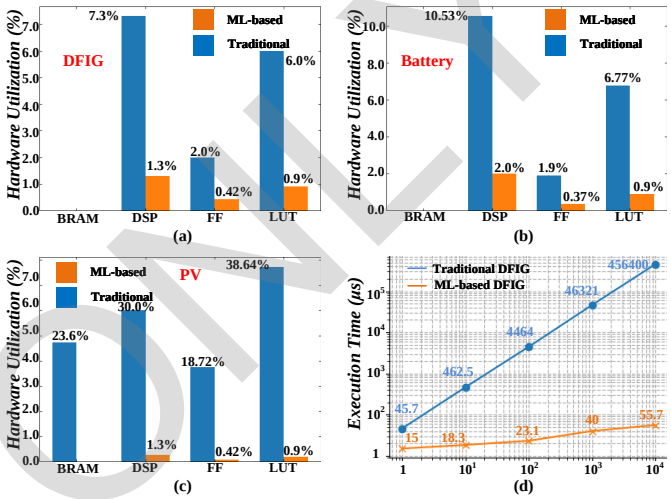

Fig. 13. Performance Evaluations: (a) FPGA resource utilization ML VS traditional DFIG, (b) FPGA resource utilization ML VS traditional battery, (c) FPGA resource utilization ML VS traditional PV, and (d) execution time per simulation step vs. number of wind turbines between traditional and ML-based DFIG model.

consistency with the expected behavior of the photovoltaic model, thereby underscoring the MLP model's robust generalization capabilities.

C. Performance Evaluation

To showcase the effectiveness of the proposed hybrid neural network-based IBR models, a comparative analysis was conducted with the traditional nonlinear IBR models which have been detailed in Section II. The evaluation was carried out on the same Xilinx® VCU118 board to assess the resource utilization of each approach. The scale of the models tested aligns with the specifications illustrated in Fig. 3, including a six-turbine DFIG model, a 3x4 battery array, and a 4x4 PV array. The comparison of FPGA resource utilization between traditional and ML-based IBR models is presented in Fig. 13(a) - (c), where the ML-based models demonstrate a marked reduction in hardware resource demands.

Furthermore, the time-domain performance of the traditional and ML-based models on FPGA was assessed, particularly focusing on the execution time for a single EMT simulation time-step, set at 50 μs. After testing, all three IBR models exhibit similar performance over their respective traditional models. Therefore, the DFIG wind turbine is used as an

example here. A range of wind turbine installations was considered to highlight the contrast between the scalability of the traditional and ML-based models. As the number of turbines increased from 1 to 10,000, distinct scalability trends emerged. Fig. 13(d) reveals that when the turbine count is below 10, both the traditional and the ML-based models perform comparably. However, for installation exceeding this threshold, the traditional model grows linearly in execution time while the ML-based model consistently delivers lower execution times. A speed-up rate surpassing 8193 times was observed when the system was scaled up to 10,000 wind turbines.

V. Conclusion

This work introduces an innovative approach leveraging machine learning for real-time, transient emulation, specifically tailored for renewable IBRs. Utilizing hybrid neural networks that capitalize on the temporal dependencies characteristic of traditional renewable energy models, the machine learning performance has been enhanced. The proposed ML-based IBR models are batch-processed in parallel, enabling the representation of system-level behaviors. Constructed on the framework of an actual AC grid, the test bench system's outcomes were validated by conventional models, revealing noteworthy advancements.

The advantages of the ML-based models presented in this paper are twofold: firstly, computational efficiency is significantly heightened. The machine learning-based approach circumvents the laborious, iterative resolution of discrete-time, non-linear equations inherent in traditional models, thus expediting the emulation process substantially. Secondly, the proposed model outperforms traditional models on FPGA in terms of execution time. The proposed ML-based models exhibit much higher scalability, especially for emulating large-scale renewable energy installations. Thirdly, the models have demonstrated high accuracy, as evidenced by the emulation results. The relative errors, when contrasted with traditional models, are minimal, rendering them negligible for practical purposes.

Nonetheless, there remains room for further optimization. On the one hand, the complexity of current ML-based models can still be increased to achieve higher accuracy in handling wider input data. On the other hand, the ML-based models are optimized for inputs within a specified range of rated values. Outlier data may lead to inaccurate predictions, a concern compounded by the potential for error accumulation inherent to the neural networks' mathematical structure. An improvement could be realized by integrating a conventional model within the ML-based model array. This hybrid approach would allow the traditional model to monitor and correct the ML models' relative errors when they exceed a predetermined threshold.

In summary, the ML-based models proffered in this research not only exhibit high computational efficiency and accuracy but also hold the promise of markedly enhancing the real-time emulation capabilities of renewable energy sources.

References

[1] X. Liang, "Emerging power quality challenges due to integration of renewable energy sources," *IEEE Trans. Ind. Appl.*, vol. 53, no. 2, pp. 855–866, March-April 2017.

[2] V. Dinavahi and N. Lin, *Real-Time Electromagnetic Transient Simulation of AC-DC Networks*. Hoboken, NJ, USA: Wiley-IEEE, 2021.

[3] S.-K. Kim, J.-H. Jeon, C.-H. Cho, E.-S. Kim, and J.-B. Ahn, "Modeling and simulation of a grid-connected PV generation system for electromagnetic transient analysis," *Solar Energy*, vol. 83, no. 5, pp. 664–678, 2009.

[4] J. Marchand, A. Shetgaonkar, J. L. Rueda Torres, A. Lekic, and P. Palensky, "EMT real-time simulation model of a 2 GW offshore renewable energy hub integrating electrolysers," *Energies*, vol. 14, no. 24, 2021. [Online]. Available: https://www.mdpi.com/1996-1073/14/24/8547

[5] X. Wang and S. Han, "Modeling of DFIG based wind farm considering temporal and spatial non-uniformity of wind speed in mountainous region and its applicability analysis," *2017 29th Chinese Control And Decision Conference (CCDC)*, Chongqing, China, 2017.

[6] Z. Zhou and V. Dinavahi, "Parallel Massive-Thread Electromagnetic Transient Simulation on GPU," *IEEE Trans. Power Delivery.*, vol. 29, no. 3, pp. 1045-1053, June 2014.

[7] J. H. Chow, *Power System Coherency and Model Reduction*, Springer New York, 2013.

[8] M. Cui, F. Li, H. Cui, S. Bu, and D. Shi, "Data-driven joint voltage stability assessment considering load uncertainty: A variational bayes inference integrated with multi-CNNs," *IEEE Trans. Power Syst.*, vol. 37, no. 3, pp. 1904–1915, May 2022.

[9] Z. Yi, Y. Xu, X. Wang, W. Gu, H. Sun, Q. Wu, and C. Wu, "An improved two-stage deep reinforcement learning approach for regulation service disaggregation in a virtual power plant," *IEEE Trans. Smart Grid*, vol. 13, no. 4, pp. 2844–2858, 2022.

[10] H. Long, R. Geng, W. Sheng, H. Hui, R. Li, and W. Gu, "Small sample solar power interval prediction based on instance-based transfer learning," *IEEE Trans. Ind. Appl.*, vol. 59, no. 5, pp. 5283–5292, 2023.

[11] Q. Liu, T. Liang, and V. Dinavahi, "Real-time hierarchical neural network based fault detection and isolation for high-speed railway system under hybrid AC/DC grid," *IEEE Trans. Power Del.*, vol. 35, no. 6, pp. 2853- 2864, Dec. 2020.

[12] N. Lin, S. Cao, and V. Dinavahi, "Massively parallel modeling of battery energy storage systems for AC/DC grid high-performance transient simulation," *IEEE Trans. Power Syst.*, vol. 38, no. 3, pp. 2736–2747, May 2023.

[13] N. Lin, S. Cao, and V. Dinavahi, "Comprehensive modeling of large photovoltaic systems for heterogeneous parallel transient simulation of integrated AC/DC grid," *IEEE Trans. Energy Conversion.*, vol. 35, no. 2, pp. 917–927, June 2020.

[14] A. Constantin, A. Ellerbrock, F. Fernandez, and J. Rueb.beta, "co-simulation of power electronic dominated networks," *IEEE Power Energy Mag.*, vol. 18, no. 2, pp. 84–89, Mar./Apr. 2020.

[15] K. Sano, S. Horiuchi, and T. Noda, "Comparison and selection of grid-tied inverter models for accurate and efficient EMT simulations," *IEEE Trans. Power Electron.*, vol. 37, no. 3, pp. 3462–3472, Mar. 2022.

[16] Q. Huang and V. Vittal, "Application of electromagnetic transient-transient stability hybrid simulation to FIDVR study," IEEE Transactions on Power Systems, vol. 31, no. 4, pp. 2634–2646, Jul. 2016.

[17] Q. Huang and V. Vittal, "Advanced EMT and phasor-domain hybrid simulation with simulation mode switching capability for transmission and distribution systems," IEEE Transactions on Power Systems, vol. 33, no. 6, pp. 6298–6308, Nov. 2018.

[18] S. Zhang, T. Liang, T. Cheng, and V. Dinavahi, "Machine learning based modeling for real-time inferencer-in-the-loop hardware emulation of high-speed rail microgrid," *IEEE Journal of Emerg. Sel. Topics Power Electron.*, vol. 3, no. 4, pp. 920–932, 2022.

[19] S. Zhang, T. Liang, and V. Dinavahi, "Hybrid ML-EMT-based digital twin for device-level HIL real-time emulation of ship-board microgrid on FPGA," *IEEE J. Emerg. Sel. Top. Ind. Electron.*, vol. 4, no. 4, pp. 1265–1277, Oct. 2023.

[20] S. Cao, V. Dinavahi, and N. Lin, "Machine learning based transient stability emulation and dynamic system equivalencing of large-scale AC-DC grids for faster-than-real-time digital twin," *IEEE Access*, vol. 10, pp. 112 975–112 988, 2022.

[21] H. Mohajerani, U. Deshpande, and N. C. Kar, "RNN-based High Fidelity Permanent Magnet Synchronous Motor emulator considering driving inverter switching faults," IEEE Journal of Emerging and Selected Topics in Industrial Electronics, pp. 1–14, 2024.

[22] MathWorks, "Wind Farm - Doubly-Fed Induction Generator (DFIG) Detailed Model," MATLAB & Simulink. [Online]. Available: https://www.mathworks.com/help/sps/ug/wind-farm-dfig-detailed-model.html. (Accessed: Dec. 20, 2023).

[23] F. Shiravani et al., "Predictive PI control for Maximum Power Point Tracking and DC-link voltage regulation of PMVG-based wind turbine systems," IEEE Journal of Emerging and Selected Topics in Industrial Electronics, pp. 1–11, 2024.

[24] K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder-decoder for statistical machine translation," *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014.

[25] S. HAYKIN, Neural Networks and Learning Machines. Upper Saddle River; Boston; Columbus etc.: Pearson Education, 2009.

[26] K. Sanjar, A. Rehman, A. Paul and K. JeongHong, "Weight Dropout for Preventing Neural Networks from Overfitting," 2020 8th International Conference on Orange Technology (ICOT), Daegu, Korea (South), 2020, pp. 1-4.

[27] S. Hochreiter, "The vanishing gradient problem during learning recurrent neural nets and problem solutions," *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 06, no. 02, pp. 107-116, Apr. 1998.

[28] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," arXiv preprint arXiv:1412.6980, 2014. [Online]. Available: https://arxiv.org/abs/1412.6980

[29] PyTorch, "Quantization," in *PyTorch 2.1 Documentation*. [Online]. Available: https://pytorch.org/docs/stable/quantization.html. (Accessed: Dec. 15, 2023).

[30] Xilinx, "Vitis High-Level Synthesis User Guide," version 2023.2, Dec. 2023. [Online]. Available: https://docs.amd.com/r/en-US/ug1399-vitis-hls. (Accessed: Apr. 28, 2024).