Data-driven Cyberattack Detection for Photovoltaic (PV) Systems through Analyzing Micro-PMU Data

Qi Li, Fangyu Li, Jinan Zhang, Jin Ye, Wenzhan Song
School of Electrical and Computer Engineering
University of Georgia
Athens, Georgia, 30602
Email: {ql61608, fangyu.li, jinan.zhang, jin.ye, wsong}@uga.edu

Alan Mantooth
Department of Electrical Engineering
University of Arkansas
Fayetteville, Arkansas, 72701
Email: mantooth@uark.edu

***Abstract*—With increasing exposure to software-based sensing and control, Photovoltaic (PV) systems are facing higher risks of cyber attacks. To ensure the system stability and minimize potential economic losses, it is imperative to monitor operating states and detect attacks at the early stage. To meet this demand, Micro-Phasor Measurement Units (μPMU) are increasingly popular in monitoring distribution networks. However, due to the relatively low sampling rate, μPMU has not yet been used to detect and classify cyber-attacks in power electronics enabled smart grid. To our knowledge, this is one of the first attempts to use μPMU to detect cyber attacks that degrade the performance of power electronics systems. We propose to apply data-driven methods on micro-PMU data to implement attack detection. We have evaluated data-driven methods, including decision tree (DT), K-nearest neighbor (KNN), support vector machine (SVM), artificial neural network (ANN), long short-term memory (LSTM) and convolutional neural network (CNN). The proposed CNN model achieves the required performances with the highest 99.23% accuracy and 0.9963 F_1 score.**

***Index Terms*—PV systems, Cyber physical security, Data-driven models**

I. Introduction

As a promising alternative power resource, Photovoltaic (PV) systems are increasingly deployed in recent years to meet the growing demand for electricity [1]. With the advent of the Internet of Things (IoT) era, PV systems have been exposed to cyber/physical attacks That are more likely to occur than before [2]. To ensure the PV system's normal operation and maximum output power, effective and quick detection of potential attacks is critical. Some literature discussed the impact of cyber attack drive, electronics, and electric vehicles [3]–[6]. However, most current works focus on fault detection and diagnosis [7]. The only relevant work is cyber-physical attack detection and diagnosis in PV farms via analyzing electrical waveform data [8]. As electrical waveform data may not be available in the actual power grid, phasor measurement units (PMUs) are most commonly used in monitoring today's power grid characteristics (magnitude, frequency, and phase angle of an electrical quantity) [9], [10].

The advent of PMU in the power system has revolutionized the way the electric power grid is monitored and controlled [9]. Thanks to the synchronization achieved by GPS, PMU can provide positive sequence voltage and current measurements synchronized within a microsecond, which allows the monitoring of dynamic phenomena. For the past few years, with higher accuracy and higher sampling rate (120Hz), micro-PMUs (μPMU) have been introduced as new sensor technologies with the goal of enhancing real-time monitoring in power distribution systems [11]. However, compared with the electrical waveform data with a sampling rate higher than 1kHz, the feasibility of using μPMU for cyber-attack detection of power electronics systems needs to be studied. This is because the potential cyber-attacks could bring some high-frequency components that might be neglected by μPMU.

Here, we propose modern data-driven methods to analyze μPMU data for cyber-attack detection in the PV systems. Compared with traditional methods, data-driven methods [12], [13] have the advantage of better feature extraction. They do not need to consider explicit mathematical models of various attacks, as it is tough to describe the different types of attacks in a specific model. The data-driven methods that we choose are as follows: Decision Tree (DT), K-nearest neighbor (KNN), Support Vector Machine (SVM), Artificial Neural Network (ANN), Long short-term memory (LSTM) and Convolutional Neural Network (CNN) [14], [15], which cover classic machine learning and deep learning models. We try to find out which fits better with μPMU data.

In this paper, we first built a solar farm model using MATLAB Simulink, simulating the attack scenarios on the solar farm and generating the μPMU sample data. Secondly, we implement attack detection experiments in data-driven methods mentioned above. Lastly, we analyzed the experimental results and verified the feasibility of μPMU in attack detection for the PV system of a solar farm.

II. PV farm model and μPMU Dataset

A. Cyber attack model for PV farm

The proposed data-driven evaluation methods are applied to a grid-connected PV farm [8]. Fig. 1 shows the schematic diagram and topology of our solar farm Simulink model, which consists of solar panels, first stage DC/DC converter, second stage DC/AC inverter, and the transformer. The main power grid is modeled as an ideal voltage source with a linear load. One rate voltage of $260V/25kV$, 400kVA, transformer connects the PV farm, which includes four DC/DC converters and one DC/AC inverter, to the power grid.

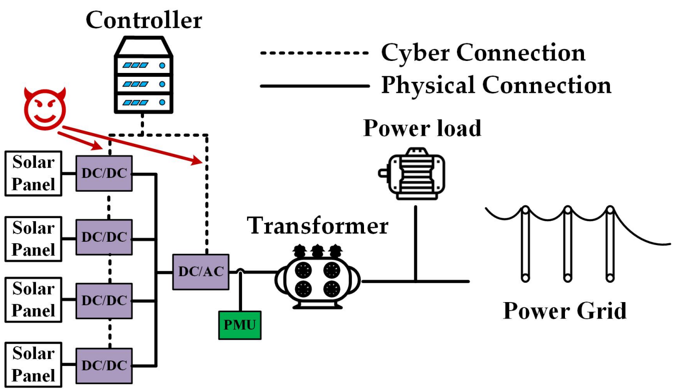

Fig. 1: Schematic diagram of the solar farm Simulink model. An attacker could use the cyber connection between the controller and DC/DC converter or DC/AC inverter to launch cyber attacks. The PMU is installed between DC/AC inverter and the transformer, which records the μPMU data.

In this paper, the cyber attacks we simulate are data integrity attacks, which are designed for compromising the DC/DC converter and DC/AC inverter sensors. To demonstrate cyber attacks on the sensor, the measurement and fake measurement are modeled as $y, \hat{y}$, respectively. Then, the cyber attack that changes sensor feedback can be expressed as,

$$\hat{y}(t) = \alpha y(t - t_{delay}), \quad t \in T_{attack}, \tag{1}$$

where, α is greater or smaller than 1, t_{delay} is the time dalay injection, $T_{attack} = [T_0, T_0 + T]$.

We considered the following three scenarios: 1) Normal with different output power levels. 2) False data injection attack on the DC/DC converter. This attack maliciously modifies the sensor/feedback data of the DC/DC converter. 3) Delay attack on DC/AC converter. This attack introduces delays in the feedback of the DC/AC inverter, which leads to degraded controller performance. For the DC/DC controller, cyber attacks on its sensor only change the voltage and current of the PV panel. The α_V and α_I represent fake measurement coefficient of voltage and current in the PV panel. $(\alpha_V, \alpha_I) \in [(0,0),(2,3),(2,0.3),(0.5,3),(0.5,0.3)]$. For the DC/AC controller, the cyber attacks would inject a time delay into the feedback signal of the sensor, $t_{delay} \in [0, 4ms, 6ms, 8ms, 10ms, 12ms, 14ms]$. To consider the uncertainty of cyber attacks, the attacks happened at different time are also simulated in our model, such as phase angles $0°, 30°, 60°, 90°, 120°, 150°, 180°$. In addition, to improve the robustness of the data-driven models by feeding in training data under different conditions, we also consider the radiation impact on power generation. The radiation on the PV panel varies in range of 900, 941, 967, 988, 1000 w/m^2. In this case, more than 3,900 training samples are simulated. The sampling frequency of raw waveform data is 1 kHz, and 0.7 seconds(s) data are captured for each scenario. After converting to μPMU data by PMU, each training sample has 84 sample points.

B. μPMU Dataset

Fig. 2 shows an example of a voltage μPMU data sample when there is no attack. The upper left picture shows the raw waveform data. Although the sampling rate of μPMU data is much less compared to the original waveform, the three features of the magnitude, frequency, and phase angle of the waveform are directly obtained from the hardware calculation of PMU device.

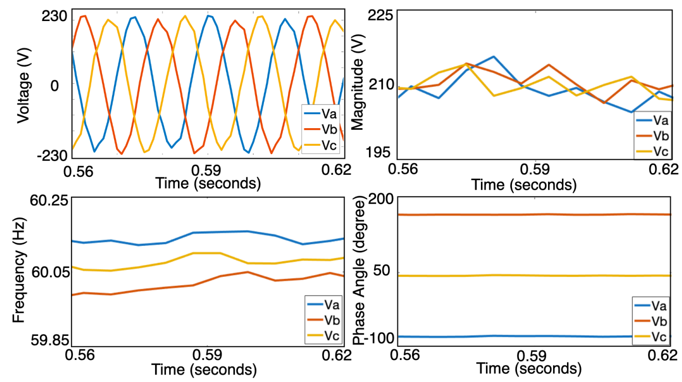

Fig. 2: An example of a μPMU voltage data sample (normal). The upper left depicts the raw waveform data, and the rest represent its corresponding μPMU data.

As discussed above, the μPMU data are collected from the PMU device annotated in Fig. 1. At each sampling time, an 18-dimensional μPMU data vector is obtained. We denote three-phase (a, b, c) voltage (V) μPMU data (θ, F, M represent phase angle, frequency and magnitude, respectively) as: θ_{V_a}, F_{V_a}, M_{V_a}, θ_{V_b}, F_{V_b}, M_{V_b}, θ_{V_c}, F_{V_c}, M_{V_c} and three-phase current (I) μPMU data as: θ_{I_a}, F_{I_a}, M_{I_a}, θ_{I_b}, F_{I_b}, M_{I_b}, θ_{I_c}, F_{I_c}, M_{I_c}.

All the data at each time point constitute a μPMU data sample, which can be represented as a 1-D time sequence:

$$X = (X_1^{\mu\text{PMU}}, X_2^{\mu\text{PMU}}, ..., X_t^{\mu\text{PMU}}), \tag{2}$$

where $X_t^{\mu\text{PMU}}$ represents the μPMU data sampled at time t. The equation

$$X_t^{\mu\text{PMU}} = (\theta_{V_a}^t, F_{V_a}^t, M_{V_a}^t ..., \theta_{I_c}^t, F_{I_c}^t, M_{I_c}^t), \tag{3}$$

shows the features that μPMU sample data have (18 features in total).

III. Data-driven methods

In order to extensively verify the feasibility of using μPMU data for data-driven based attack detection, we have selected and implemented 6 different kinds of popular supervised data-driven methods, including both machine learning and deep learning multiclass classification algorithms [14]–[16]. They are Decision Tree (DT), K-Nearest Neighbor (KNN), Support Vector Machine (SVM), Neural Network (NN), Long Short-Term Memory (LSTM) and Convolutional Neural Networks (CNNs). According to the inherent complexity of the model, Each method has different processing capabilities to handle the problem. Through comparison and evaluation, we can find the most suitable solution for us.

A. Machine Learning

1) Decision Tree: The DT is a tree-like structure in which each internal node represents a decision or selection on a feature. Each leaf node represents a class label (decision made after considering all features), and branches represent conjunctions of features that lead to the class labels. The paths from the root to the leaf represent classification rules. It has advantages like no assumptions about the shape of the data and fasts for inference. But it also easily get overfit.

2) K-Nearest Neighbor: As can be seen from its name, the KNN algorithm assumes that similar things exist in close proximity. It is based on the principle that the value of the label of an unclassified instance can be determined by observing the class of its nearest neighbors. Known for its intuitive and simple, it is often used as a benchmark for more complex classifiers. KNN is a non-parametric algorithm, which means there are also no assumptions to be met, so it is straightforward to implement for multiclass classification problems. However, as the dataset grows, the efficiency of the algorithm declines very fast. It does not perform well on imbalanced data, which means that different classes of data in the dataset are relatively uneven. The key to good performance is to choose an optimal number of neighbors to be considered while classifying the new data entry, which is tricky.

3) Support Vector Machine: The basic idea is to create a boundary or a hyperplane which separates the data into classes. Intuitively, the further from the hyperplane data points lie, the more confident we are that they have been correctly classified. In general, it's not easy to have a directly separable set of training data. That is where the kernel trick comes in, whose idea is mapping the non-linear separable dataset into a higher dimensional space where we could find a hyperplane to separate the data. In fact, we don't have to explicitly calculate the coordinates of the data points in the new space. We just need to compute the distance between data points in that space using a kernel function, which is typically crafted by hand rather than learned from data. SVM exhibited state-of-the-art performance on simple classification problems [16], and the theory behind makes it well understood and interpretable. However, SVM does not perform well for large datasets such as image classification, and the training time is much higher.

4) Neural Network: To avoid misunderstanding, NN in our paper refers specifically to a fully-connected neural network. As the basis of deep learning, NN is flexible and can be used for both regression and classification problems. Any data which can be made numeric could be used in NN. It is good to model with nonlinear data with a large number of inputs. NN performs well when the problem is simple; however, as the problem gets complicated, it is usually used as the output layer of a more complex network.

B. Deep Learning

1) Long short-term memory: LSTM is a variant of the recurrent neural network(RNN). It was developed to solve the vanishing gradient problem of RNN by adding a way to carry the past information across the time steps. In this case, information is saved for later, thus preventing older data from gradually vanishing during training. LSTM is versatile which can process not only single data points but also entire sequences of data, especially time series data. It shows powerful capability when handling scenarios like natural language processing (NLP), speech recognition. To increase the network capacity, the classic way is to stack multiple LSTM layers together, called Deep LSTM. In this paper, we implemented a two stack-layer LSTM network.

2) Convolutional Neural Network: Thanks to its hierarchical structure and powerful feature extraction capabilities, CNN has shown its promising performance to deal with image classification and object detection problems. Similarly, It also has the potential to handle time series data with its highly noise-resistant property and capacity to extract informative, deep features. By converting a window-length time series data sample into a matrix, we can feed it into the CNN like an image. Fig. 3 shows a simple architecture of CNN for classification, including the primary operations: 1) convolutional layer, whose main purpose is to extract features from the input data sample. 2) ReLU, whose main purpose is to introduce non-linearity in the network. 3) Pooling, also called subsampling or downsampling, whose main purpose is to reduce the dimensionality of each feature map but keep the most critical information. 4) Fully Connected Layer, whose main purpose is to use features from previous layers for classifying the input into various categories.

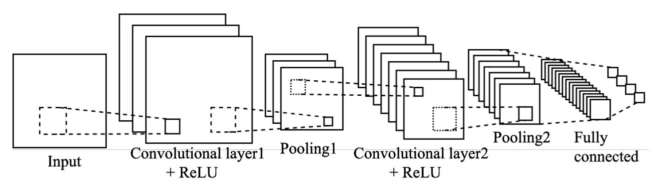

Fig. 3: An example CNN with two convolutional layers, two pooling layers, and a fully connected layer which decides the final classification of the image into one of several classes.

IV. Experiment and Evaluation

A. Experiment Setting

The proposed data-driven evaluation methods are applied to the solar farm benchmark model described in Fig. 1. We implemented them through Pytorch (1.3.1) and Sklearn (0.22.1) [17], [18] on a Ubuntu 16.04 server (CPU: i7-6850K, 3.60GHz; Memory: 64GB) armed with GPU (GeForce GTX 1080 Ti) for training. For the simulation, 3,939 training samples of 3 types of states (normal state, false data injection attack on the DC/DC converter, and delay attack on DC/AC converter, respectively) are generated by our solar farm model, whose simulation duration is 0.7 seconds, and the attack happens at 0.45 seconds. To evaluate the performance of those data-driven models, we use accuracy (ACC), precision (Prec), recall (Rec), and F_1-score (F_1) [19], [20] as metrics, which are frequently used in the evaluation of multiclass classification defined as follows:

$$Prec = \frac{TP}{TP + FP}, \tag{4}$$

$$Rec = \frac{TP}{TP + FN}, \tag{5}$$

$$F_1 = \frac{2 \times TP}{2 \times TP + FP + FN}, \tag{6}$$

$$Acc = \frac{TN + TP}{TN + TP + FP + FN}, \tag{7}$$

where true positive (TP) denotes the rate of actual attack is correctly predicted as attack, true negative (TN) denotes the rate of actual normal is correctly predicted as normal, false positive (FP) denotes the rate of actual attack is wrongly predicted as normal, false negative (FN) denotes the rate of actual normal is wrongly predicted as normal. All those metrics above are expected to be as high as possible, where F_1 takes both the Prec and Rec into consideration [21]. To eliminate the imbalance of the dataset, we calculated these metrics considering the weight of different states [22]. Besides, we also concern the Receiver operating characteristic (ROC) curve and Area Under Curve (AUC) (the area under ROC curve) [22], which are performance measurements for classification problem that usually appears in pairs. They tell how much the model could distinguish between classes. Higher the AUC, the better the model is.

B. Data Preprocessing

As the very first step of the data-driven approach, data preprocessing usually transforms raw data into a more understandable format. In our case, we first removed the unexpected distortion at the beginning of the data sample caused by Simulink simulation, which would lead to incorrect results. Then, according to the attack type, we labeled every data sample for training. Lastly, before training the models, due to the large different value range of different features, we applied data normalization using the Z-Score method [23] as a preprocessing on each feature of the μPMU data separately so that it had a standard deviation of 1 and a mean of 0. For example, the normalized θ_{V_a} can be obtained by:

$$Ph^{*t}_{V_a} = \frac{Ph^t_{V_a} - \overline{Ph_{V_a}}}{\sigma}, \tag{8}$$

where $\overline{\theta_{V_a}}$ and σ are the mean value and standard deviation of the feature θ_{V_a} during entire time range. Similarly, we can also get other normalized features. For every μPMU data sample, we constructed a normalized high-dimensional matrix $X_{t \times s}$, where t represents the length of the sample sequence and s represents the number of features. So in our case, $t = 85$ (120 Hz sampling rate in 0.7 seconds) and $s = 18$ (18 features in total), respectively. Eventually, each sample data to be input to the models denote by the matrix $X_{84 \times 18}$.

For the CNN model, we treat every input matrix $X_{84 \times 18}$ as a pixel matrix, just like CNN usually does for image classification. The width of the matrix represents the time steps, and the length of the matrix represents the normalized features at each time point. For the LSTM model, we treat every row (features at every time point) of $X_{84 \times 18}$ as the input of every time step, and the time sequence length of LSTM is the width of $X_{84 \times 18}$, namely 84. For other models, the input matrix is directly flattened into a one-dimensional vector with 84×18 elements.

C. Machine Learning Model Training

After a large number of trials and tuning, to achieve the tradeoff between convergence speed and performance, the hyperparameters of the six data-driven methods are decided as follows: 1) For CNN, we set the batch size 32, two convolutional layers (kernel size is 5, the activation function is ReLu, input channel and output channel are 1, 16, respectively, the stride is set at 1, and padding equals to 2), one max pooling layer. The output of convolutional layers directly connected one fully-connected layer that outputs the result. 2) For LSTM, the batch size is 32, the number of hidden states is 32, two stack layer, one fully-connected layer. To get more power to learn the latent information, we stack the extra lstm layer. 3) For ANN, the batch size is set at 32, and three fully-connected layers are deployed to make it comparable with the other two deep learning methods. 4) For SVM, we implemented using the SVC function from sklearn [17]. The sigmoid kernel function is adopted, and the regularization parameter equals to 1. 5) For DT, max depth is set to 2. 6) For KNN, the number of neighbors is set to 45. To achieve more reliable results, we adopt the 10-fold cross validation, which splits the available data into 10 partitions, instantiating 10 identical models, and training each one on 9 partitions while evaluating on the remaining partition. The validation metrics for the model used is the average of the 10 validation obtained.

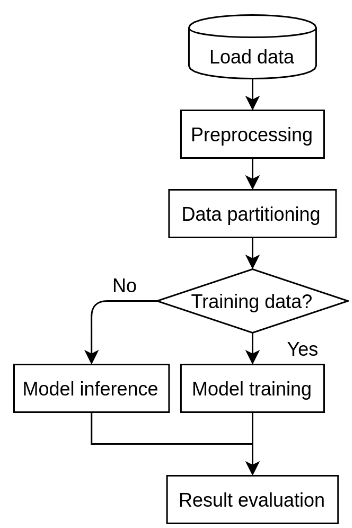

Fig. 4: The overall block diagram showing the workflow of the proposed data-driven evaluation methods.

The overall block diagram is shown in Fig. 4, which describes the workflow of the proposed data-driven evaluation method: 1) Firstly, the μPMU data are loaded from the database. 2)Then data preprocessing are applied, consisting of labeling data, cleaning the data by removing the useless data points, and normalize the data. 3) Next, data partitioning is used to split the data into training data and test data. It is noted that we used 10 fold cross validation, so we have 10 different randomly divided data partitions. 4) Once the model is trained using training data, we do the model inference with testing data, calculating the metrics. Repeat step (3) and (4) 10 times, the final evaluation results are the average values of metrics in each model inference.

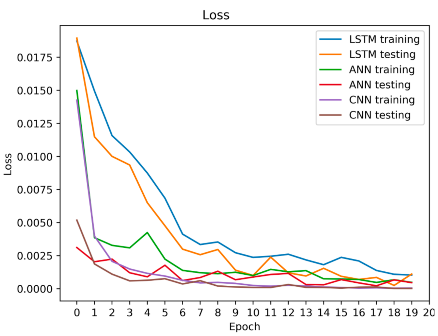

Fig. 5: Loss curve which depicts the training and testing loss curves of two deep learning models and ANN.

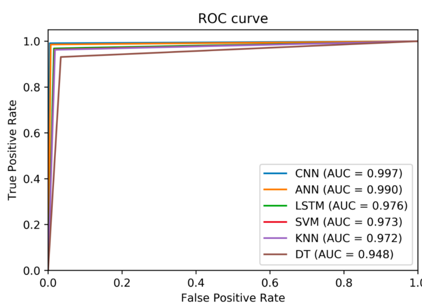

Fig. 6: ROC curve which depicts the ROC curves of all data-driven models and their corresponding AUC.

D. Evaluation

The experiment results are presented below after we conducted a 10-fold cross-validation [24]. Figs. 5 and 6 show the loss curve of two deep learning based methods with ANN and their ROC curve. Fig. 7 shows the confusion matrix of all six models. In the loss curve, it doesn't take many epochs for the three models to converge, and they all show promising results after around 12th epoch. Meanwhile, this fast convergence rate indicates that the dataset is not so complicated ,and models could learn to classify them quickly. In addition, it also proves that the μPMU data contains the key features to distinguish different attack scenarios, though its sampling rate is way lower than the raw waveform. From the ROC curve, we can tell all models show ideal performance. But the deep learning methods are still better than the others. As the scenarios become more complicated, the gap between them will become larger. It should be noted that LSTM is known for its superior memory capacity to deal with time series data. But interestingly, ANN is better than LSTM in both stability and performance. This may be because the time series is relatively too simple for LSTM to show its advantage [12]. Even though LSTM still achieved a very good performance (96.62% accuracy).

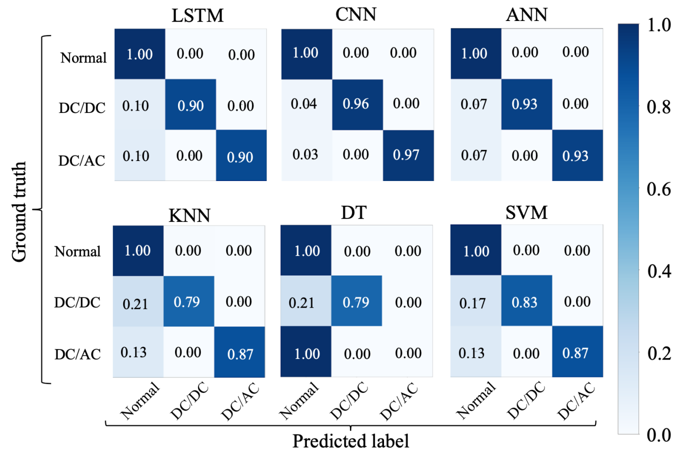

Fig. 7: Confusion matrix of six models, where the x-axis represents predicted label, and the y-axis represents ground truth.

From Table I, in comparison, CNN outperforms all other models in all metrics (99.23% accuracy). We contribute this to its powerful capability of extracting features and latent information; moreover, it's indeed possible to visually distinguish a part of μPMU data. This means that CNN not only has outstanding capability in dealing with the image but also has the potential to process time series data. As for other models, it makes sense that SVM is better than KNN and DT when there are not many data samples. It is worth mentioning that during training, SVM takes much more time than KNN and DT. In short, the proposed data-driven methods all perform well in the attack detection for our solar farm power system, which shows the potential for μPMU data to be used in the security area of the power system.

Table I: Quantitative evaluation metrics of different data-driven methods

Model\Metrics	Acc	Prec	Rec	F_1	AUC
DT	0.9390	0.9816	0.9313	0.9542	0.9485
KNN	0.9671	0.9694	0.9626	0.9642	0.9719
SVM	0.9631	0.9721	0.9641	0.9660	0.9731
LSTM	0.9662	0.9719	0.9685	0.9692	0.9764
ANN	0.9894	0.9879	0.9871	0.9857	0.9904
CNN	**0.9923**	**0.9966**	**0.9964**	**0.9963**	**0.9973**

V. Future Works

For our future research, we will use μPMU data for a variety of attack detection in power electronics enabled smart grids. We will also consider testing our method by implementing it in the real system. The heavy training tasks can be handed over to the data center or central server. Once the training is done, the trained model will be put into some lightweight and low-power light IoT devices such as Raspberry Pi. Then they will be deployed in the power system. As long as the IoT devices could receive μPMU data, they could easily do the attack detection, because the computational cost of model inference is much lower than training. Once an attack is detected, the alarm will be sent to the user or operators.

VI. Conclusions

This paper proposes implementing modern data-driven methods to validate the possibility of using μPMU data to detect and diagnose the potential attacks simulated in our PV system model of the solar farm. In conclusion, the contributions of this paper are as follows: 1) A PV power system model of a solar farm is built, which could simulate various cyber or physical attacks. Based on this model, six different data-driven methods are applied to the μPMU data for detecting cyber-attacks on the DC/DC converter and DC/AC inverter. 2) Experiment results are analyzed, and they indicate the μPMU data can be used by data-driven methods to successfully detect the attacks on the PV system model with high accuracy. Our work shows the μPMU data have the potential to be used in modern data-driven methods to solve the security problems in nowadays power system.

VII. Acknowledgements

Our research is sponsored by NSF-ECCS-1946057 and the Department of Energy Solar Technologies Office.

References

[1] B. Subudhi and R. Pradhan, "A comparative study on maximum power point tracking techniques for photovoltaic power systems," *IEEE transactions on Sustainable Energy*, vol. 4, no. 1, pp. 89–98, 2012.

[2] J. C. Balda, A. Mantooth, R. Blum, and P. Tenti, "Cybersecurity and power electronics: Addressing the security vulnerabilities of the internet of things," *IEEE Power Electronics Magazine*, vol. 4, no. 4, pp. 37–43, 2017.

[3] B. Yang, L. Guo, F. Li, J. Ye, and W. Song, "Impact analysis of data integrity attacks on power electronics and electric drives," in *2019 IEEE Transportation Electrification Conference and Expo (ITEC)*. IEEE, 2019, pp. 1–6.

[4] B. Yang, F. Li, J. Ye, and W. Song, "Condition monitoring and fault diagnosis of generators in power networks," in *2019 IEEE Power & Energy Society General Meeting (PESGM)*. IEEE, 2019, pp. 1–5.

[5] B. Yang, L. Guo, and J. Ye, "Real-time simulation of electric vehicle powertrain: hardware-in-the-loop (hil) testbed for cyber-physical security," in *2020 IEEE Transportation Electrification Conference and Expo (ITEC)*. IEEE, 2020.

[6] L. Guo, B. Yang, and J. Ye, "Enhanced cyber-physical security of steering stability control system for four-wheel independent drive electric vehicles," in *2020 IEEE Transportation Electrification Conference and Expo (ITEC)*. IEEE, 2020.

[7] Z. Xue, K. Xiahou, M. Li, T. Ji, and Q. Wu, "Diagnosis of multiple open-circuit switch faults based on long short-term memory network for dfig-based wind turbine systems," *IEEE Journal of Emerging and Selected Topics in Power Electronics*, 2019.

[8] F. Li, R. Xie, B. Yang, L. Guo, P. Ma, J. Shi, J. Ye, and W. Song, "Detection and identification of cyber and physical attacks on distribution power grids with pvs: An online high-dimensional data-driven approach," *IEEE Journal of Emerging and Selected Topics in Power Electronics*, pp. 1–10, 2019, online.

[9] D. Kumar, D. Ghosh, and D. K. Mohanta, "Simulation of phasor measurement unit (pmu) in matlab," in *2015 International Conference on Signal Processing and Communication Engineering Systems*. IEEE, 2015, pp. 15–18.

[10] A. G. Phadke and J. S. Thorp, *Synchronized phasor measurements and their applications*. Springer, 2008, vol. 1.

[11] H. Mohsenian-Rad, E. Stewart, and E. Cortez, "Distribution synchrophasors: Pairing big data with analytics to create actionable information," *IEEE Power and Energy Magazine*, vol. 16, no. 3, pp. 26–34, 2018.

[12] F. Li, A. Shinde, Y. Shi, J. Ye, X.-Y. Li, and W.-Z. Song, "System statistics learning-based iot security: Feasibility and suitability," *IEEE Internet of Things Journal*, vol. 6, no. 4, pp. 6396–6403, 2019.

[13] F. Li, Y. Shi, A. Shinde, J. Ye, and W.-Z. Song, "Enhanced cyber-physical security in internet of things through energy auditing," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 5224–5231, 2019.

[14] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016.

[15] S. B. Kotsiantis, I. Zaharakis, and P. Pintelas, "Supervised machine learning: A review of classification techniques," *Emerging artificial intelligence applications in computer engineering*, vol. 160, pp. 3–24, 2007.

[16] F. Chollet, *Deep Learning mit Python und Keras: Das Praxis-Handbuch vom Entwickler der Keras-Bibliothek*. MITP-Verlags GmbH & Co. KG, 2018.

[17] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[18] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," 2017.

[19] Y. Shi, F. Li, T. Liu, F. R. Beyette, and W. Song, "Dynamic time-frequency feature extraction for brain activity recognition," in *2018 40th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*. IEEE, 2018, pp. 3104–3107.

[20] M. Sokolova and G. Lapalme, "A systematic analysis of performance measures for classification tasks," *Information processing & management*, vol. 45, no. 4, pp. 427–437, 2009.

[21] J. Clemente, F. Li, M. Valero, and W. Song, "Smart seismic sensing for indoor fall detection, location, and notification," *IEEE journal of biomedical and health informatics*, vol. 24, no. 2, pp. 524–532, 2019.

[22] J. Davis and M. Goadrich, "The relationship between precision-recall and roc curves," in *Proceedings of the 23rd international conference on Machine learning*. ACM, 2006, pp. 233–240.

[23] A. Jain, K. Nandakumar, and A. Ross, "Score normalization in multimodal biometric systems," *Pattern recognition*, vol. 38, no. 12, pp. 2270–2285, 2005.

[24] R. Kohavi *et al.*, "A study of cross-validation and bootstrap for accuracy estimation and model selection," in *Ijcai*, vol. 14, no. 2. Montreal, Canada, 1995, pp. 1137–1145.