IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received June 11, 2020, accepted June 27, 2020, date of publication July 7, 2020, date of current version July 17, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3007609

Towards Accurate and Efficient Classification of Power System Contingencies and Cyber-Attacks Using Recurrent Neural Networks

WEI-CHIH HONG, (Member, IEEE), DING-RAY HUANG, CHIH-LUNG CHEN, AND JUNG-SAN LEE
Department of Information Engineering and Computer Science, Feng Chia University, Taichung 40724, Taiwan

Corresponding author: Wei-Chih Hong (wchong@fcu.edu.tw)

This work was supported in part by the Taiwan Information Security Center at NCHU (TWISC@NCHU), and in part by the Ministry of Science and Technology, Taiwan, under Grant MOST 108-2218-E-005-016 and Grant MOST 109-2218-E-005-005.

ABSTRACT Correct and timely responses to abnormal conditions in the power systems are crucial to their sound operation. In order for the operator or the automated response system to take prompt measures during system contingencies, it is critical to facilitate an accurate mechanism for the classification of the events and disturbances in the power grid. The massive amount of time-synchronized data recorded by the phasor measurement units can be combined with logs from other components in the power grid to create datasets for event and intrusion detection. This paper presents the results of applying deep learning techniques on open datasets recorded from a power system testbed to classify contingencies and cyber-attacks. Three different designs of recurrent neural networks (RNN) are investigated and tested for discriminating binary and multiclass events. Experiment results show 100% and 99.99% accuracy when applying the proposed classifiers on large scale binary and multiclass datasets respectively. It is also shown that one can improve the efficiency of the scheme by selectively eliminating 75% of the features in the dataset while maintaining as high as 99.96% accuracy in classifying multiclass events. Additionally, the feasibility of the design is validated by the low classification latency recorded on the low-end embedded system Jetson Nano. These promising results demonstrate the potential of employing RNN techniques in developing event and intrusion detection systems for power grids.

INDEX TERMS Intrusion detection, recurrent neural networks, smart grids, phasor measurement units.

I. INTRODUCTION

Nowadays, as the increasing adoption of information and communication technology (ICT), power systems are evolving into a large scale cyber-physical system. The smart meters, phasor measurement units (PMUs), distributed power flow control devices, together with other intelligent components are connected by ICT networks to transform the traditional power grids into more efficient and more flexible smart grids. At the same time, the connected devices and the underlying data network also unleash the threat of potential cyber-attacks against the power systems. For instance, the 2015 Ukraine blackout was reported to be caused by cyber-attacks that compromised components in the ICT network and switched off the affected substations [1]. Additionally, new types of attacks specifically tailored to undermine various components and mechanisms in the smart grid architecture, thus causing significant impact and economical losses [2]–[4]. It is crucial to take this category of attacks into account when considering the reliability and security of the power systems.

The PMUs are global positioning system (GPS)-enabled technologies that can produce time-synchronized, wide-area measurements in the transmission system of the power grid. Measured data of the PMUs, also known as the synchrophasors, facilitate advanced monitor, control, and management functionalities for the power system. Conventionally, mathematical models are derived for the state estimation (SE) module to estimate the operating conditions of the power system using the synchrophasors and other measurements of

The associate editor coordinating the review of this manuscript and approving it for publication was Canbing Li.

system topology and dynamics. Based on the output of the SE module, the contingency analysis (CA) module identifies overloads in the power system and provides analysis results for the operators or the automated self-healing function to take prompt countermeasures. The combination of the potential cyber-attacks and the inherent contingencies originated from natural failure events makes SE and CA more challenging tasks. For example, the adversaries could inject falsified data to deceive the anomaly detection functions, and wrongly classified contingencies could lead to improper response that renders negative impacts on the power system [4]. As a result, an accurate classification system for power system contingencies and cyber-attacks is much needed.

Recurrent neural network (RNN) is a category of deep learning techniques, which are capable of extracting temporal features from sequences of data and building models for classification problems. For example, RNN and its variants have been successfully applied to the fields of computer vision [5], natural language processing [6], and speech recognition [7], among others. Many recent works suggested the application of RNN-based models on analyzing network traffic can attain high accuracy classification of cyber-attacks [8]–[10]. Although applying RNN on the time sequences of synchrophasor data to construct classifiers seems straightforward, detecting intrusions and contingencies in smart grid all at once complicates the problem and might deteriorate the classification accuracy. Moreover, training and classifying with deep neural network models tend to consume large amount of memory and computing power. It is then questionable whether the RNN-based classifiers suit the requirements of an accurate and responsive contingency and cyber-attack detection system for smart grid.

In this paper, we set out to investigate the performance and efficiency of RNN-based classifiers in the framework of an event and intrusion detection system (EIDS) for the power systems (see Fig. 1). In the EIDS framework, there is a data collector responsible for receiving real-time logs and measurements from various devices and sensors in the power system, e.g. the PMUs, the relays, the control panels, etc. Additionally, a network intrusion detection system (NIDS) is employed to detect malicious attacks or penetration activities in the network that connects the devices. The alerts and/or logged data of the NIDS are also fed into the data collector for further processing. The collected data will be filtered, labeled and fed into the model trainer to generate the underlying model for the classification process. On the other hand, real-time data will be sent to the event classifier for instant identification of current events. Alerts of contingencies and cyber-attacks will be sent to the control center of the grid immediately so that timely response can be made to minimize potential damage.

The EIDS can be seen as a functional part of the advanced distribution management system (ADMS) [11], which controls the distribution grid to maintain reliable and safe operation of the power system. Depending on the scale of the power system, the EIDS framework could be implemented in either

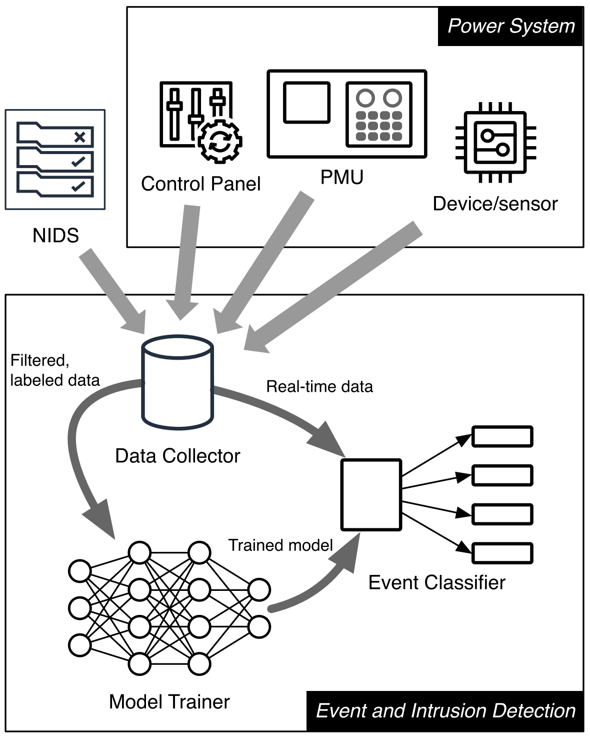


FIGURE 1. Framework for machine learning-based event and intrusion detection of the power systems.

a centralized or a distributed manner. In practical use cases, a large-scale power system may consist of hundreds of buses and PMUs. It is impractical to train a single model with tens of thousands of features for the whole system due to high computation demands of the RNN model training algorithm. Moreover, transmitting large amount of measurements across a vast area to a control center for real-time detection requires huge investment in building an efficient network infrastructure. A feasible solution is to partition the whole power system into smaller overlapping zones or subsystems, so that a modest amount of data can be transmitted and processed efficiently in the regional EIDS of each zone. The alerts and event notifications from various zones will be sent to the control center of the ADMS, where the inter-zone correlation of the cyber-attacks can be taken into account. On the contrary, if a relatively small power system with a limited number of PMUs is concerned (e.g. microgrids), a centralized EIDS can be implemented and the inter-space and inter-temporal characteristics of the cyber-attacks are considered as a whole.

The primary contribution of this paper is to evaluate the performances of three types of RNN-based classifiers, including SimpleRNN, Long Short-Term Memory (LSTM), and Gated Recurrent Unit (GRU), in the EIDS framework for the power systems. Our experiment results clearly show that the LSTM and the GRU classifiers outperform previous methods in [12]–[14] in terms of all the metrics evaluated for all datasets. In practice, these two classifiers can easily attain over 99.99% accuracy even for the large scale multiclass dataset as long as the hyperparameters are set properly.

Secondly, we have examined the possibility to greatly reduce the size of the feature set and maintain as high as 99.96% accuracy using the LSTM classifier. This possibility enables faster rebuild of the deep learning models after a topology reconfiguration of the distribution network, which is considered crucial to the operation of the smart grid [15].

The third contribution of this paper is to assess the feasibility of the classifiers by investigating their efficient applications in the distributed EIDS framework. Previously, it has been argued that the memory consumption and time complexity of the deep learning techniques may render their incompetence in real-time detection [13], [14]. Nevertheless, our results on the classification time of both LSTM and GRU on NVIDIA Jetson Nano, a low-price embedded system for accelerated neural network computation, have proven the practicality of such combinations to provide timely event and intrusion detection.

The remainder of the paper is organized as follows. Section II provides a survey of previous related works in the literature. Section III gives a brief description of the three RNN models as well as the classifier design. Next, the methodologies we adopt for evaluating the classifiers are depicted in Section IV, including the dataset, the steps of data preprocessing, and the evaluation metrics. In Section V, we present the experiment results and discussions. Finally, Section VI concludes the paper.

II. RELATED WORKS

The intrusion detection system (IDS) has been extensively discussed in the literature for various target systems in the smart grid, including the advanced metering infrastructure (AMI), the Supervisory Control and Data Acquisition (SCADA) system, the substation, and the synchrophasor system. In [16], it is pointed out that the smart grid IDS must perform timely detection of a wide range of intrusions with high accuracy and precision. Furthermore, the IDS should be scalable and provide attentive performance of computing resources.

Traditionally, signature-based techniques are used for detecting malicious traffic or malware by matching distinctive patterns or characteristics of known attacks. This type of detection techniques provides low false positive rate and is suitable for applications in the communication networks in smart grid. For example, signature-based Snort rules were proposed for detecting cyber-attacks in the Modbus [17] and DNP3 [18] networks of the SCADA system respectively. The major drawback of using signatures is that it can not detect unknown attacks and current knowledge base about cyber-attacks in smart grid is limited.

On the other hand, anomaly-based IDS establishes profiles of the normal behaviors of the system and identifies abnormal behaviors as intrusions. The profiles can be constructed using heuristics, statistical analysis, or machine learning algorithms. For example, He and Blum proposed to combine locally optimum hypothesis tests and the generalized likelihood test to improve the detection probability of intrusions and failures in the smart grid [19]. In [20], Khan *et al.* designed signature-based as well as heuristic and stateful rules for a synchrophasor specific IDS to detect both known and unknown cyber-attacks in the IEEE C37.118 communication framework. For the AMI, Faisal *et al.* tested 7 data stream mining algorithms and provided suggestions for their applications in the proposed distributed IDS framework [21]. Generally, anomaly-based IDSs are more capable of detecting unknown attacks, but might be less accurate in identifying known attacks than signature-based IDPSs.

Compared with conventional IDS techniques, it is easier to build a machine learning-based classifier that can discriminate power system faults as well as cyber-attacks at the same time. In [12], a set of traditional machine learning algorithms including OneR, NNGE, Random Forests, Naïve Bayes, SVM, and JRipper are tested on the same datasets used in our work. Among these algorithms, JRipper consistently achieves the best accuracy results in three different classification schemes, namely Binary, Three-class, and Multiclass. Additionally, the authors also incorporate the Adaboost method to further improve the performances of JRipper. However, the JRipper+Adaboost algorithm only attain sub-90% accuracy in discriminating Multiclass data because it does not consider the sequential nature of the events.

Pan *et al.* [13] proposed the application of the common path mining technique on the same power system cyber-attack dataset as in [12]. By taking into account the temporal features of the PMU data, the common path classifier can distinguish finer grained classes of the power system events at 93% accuracy. Subsequently, Adhikari *et al.* [14] combined State Extraction Method (STEM) preprocessing and Non-Nested Generalized Exemplars (NNGE) classification to boost the classifier's ability to discriminate 41 classes at 93% accuracy on an augmented dataset. The average classification time of the NNGE+STEM algorithm was reported at 0.2 millisecond. However, because of the criticality of the EIDS for power systems, the 93% accuracy can not be considered sufficient and near perfect performance should be pursued. In view of this, we propose to employ RNN-based classifiers in the EIDS to achieve higher accuracy while maintaining adequate efficiency to make timely decision on the events.

Recently, a wide range of experiment results of applying RNN and its variants for the detection of cyber-attacks have been reported in the literature. Yin *et al.* [8] tested the simple RNN model with single recurrent hidden layer on the NSL-KDD dataset [22] for both binary and multiclass classifications. Experiment results of varying the number of hidden nodes and learning rates are presented in this work. The results show that fully connected models outperform the reduced-sized RNN models both in training and testing accuracy.

Besides simple RNN, Long Short-Term Memory (LSTM) models have also been extensively investigated and shown to attain better performances on detecting cyber-attacks. Kim *et al.* [9] applied the LSTM model on the 10% KDD Cup 99 dataset and achieved 96.93% accuracy in classifying 5 categories of cyber-attack scenarios. In [23], Naseer *et al.* implemented and compared three deep learning models, namely Autoencoder, LSTM, and convolutional neural network (CNN), together with a set of conventional machine learning-based models for intrusion detection on the NSL-KDD dataset. Diro and Chilamkurti [24] proposed a LSTM model for distributed cyber-attack detection in fog-to-things communication and experimented on two different datasets. It is shown that distributing training data over the fog nodes can improve the detection accuracy and scalability.

Xu *et al.* constructed a deep neural network based IDS using Gated Recurrent Units (GRU) and multilayer perceptron (MLP) [10]. The experiments on KDD 99 and NSL-KDD datasets show that GRU performs better than LSTM in terms of accuracy, detection rate, and false positive rate in the proposed architecture.

Despite the high classification accuracy of the well-trained deep learning models, there is concern about their efficiency in processing large volume synchrophasor data generated by the PMUs in relatively high frequencies [13], [14]. To our knowledge, there was no previous work addressing the efficiency issue of applying deep learning techniques on synchrophasor data for real-time EIDS. In this paper, the feasibility of the RNN-based classifiers will be demonstrated using the timing results on a low-end embedded system. Moreover, the performance results on reduced feature sets indicate that good tradeoff between the accuracy and the efficiency can be reached for the proposed classifiers.

III. RNN-BASED CLASSIFIERS

In this section, we first illustrate three variants of the RNN models, and then depict the proposed design of the RNN-based classifiers.

A. SIMPLE RNN, LSTM, AND GRU

Unlike traditional feed-forward neural networks, RNN introduces a loop that feeds back previous state of the hidden layer to the current output. Fig. 2a is the circuit diagram of a generic RNN model, where $\boldsymbol{x}, \hat{\boldsymbol{y}}, \boldsymbol{h}$ are the input, output, and state vectors of the hidden layer respectively. The square box with a capital D represents a delay of a single time step. This recurrent loop effectively stores the "memories" from previous inputs and combines them into the computation of the current state and output of the network. Such design provides RNN with the ability to summarize the information from the past input sequence and contribute to the prediction output.

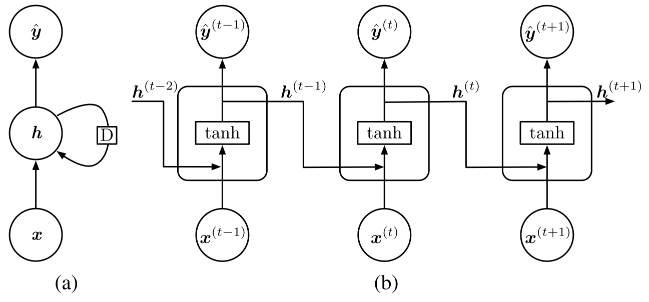


FIGURE 2. Recurrent neural network: (a) circuit diagram, (b) unrolled computational graph.

Fig. 2b depicts the unfolded computational graph of the SimpleRNN model. At each time step, an input vector $\boldsymbol{x}^{(i)}$ is concatenated with the hidden state of the previous time step $\boldsymbol{h}^{(i-1)}$ and fed into the hidden layer to compute the current state $\boldsymbol{h}^{(i)}$, which will be further processed by the output layer to produce the prediction output $\hat{\boldsymbol{y}}^{(i)}$. A typical set of update equations of the procedure described above would be the following.

$$\begin{cases} \boldsymbol{h}^{(t)} = \tanh(\boldsymbol{W}_{xh}\boldsymbol{x}^{(t)} + \boldsymbol{W}_{hh}\boldsymbol{h}^{(t-1)} + \boldsymbol{b}_h) \\ \hat{\boldsymbol{y}}^{(t)} = \mathrm{softmax}(\boldsymbol{W}_{hy}\boldsymbol{h}^{(t)} + \boldsymbol{b}_y) \end{cases} \tag{1}$$

where the parameters are the weight matrices $\boldsymbol{W}_{xh}$, $\boldsymbol{W}_{hh}$, and $\boldsymbol{W}_{hy}$ along with the bias vectors $\boldsymbol{b}_h$ and $\boldsymbol{b}_y$. In this example, the activation functions are tanh and softmax for the hidden and output layers respectively.

When training RNN models, the back-propagation through time (BPTT) [25] algorithm is applied to the unrolled computational graph to compute the gradients, which are then used to optimize the parameters of the model by various gradient descent algorithms. In many cases, training simple RNN models becomes difficult because of the vanishing gradient problem. In order to address this problem, Hochrieter and Schiemidbuher proposed the long short-term memory (LSTM) model [26], which adopts a redesigned cell for the hidden layer to capture long-term dependencies in the input sequences.

Fig. 3a illustrates the structure of a LSTM cell, which incorporates the concept of gating units to control the flow of information and learn long-term dependencies adaptively. These gates, denoted by $\boldsymbol{i}^{(t)}$, $\boldsymbol{f}^{(t)}$, $\boldsymbol{o}^{(t)}$, are actually weight vectors that are multiplied with the gated vectors in a element-wise fashion, i.e., the Hadamard product. In addition to implicitly storing the memories in the hidden layer vector $\boldsymbol{h}^{(t)}$, the LSTM network includes a vector $\boldsymbol{c}^{(t)}$ as the internal state/memory of the cell. The update of the internal state consists of the component from the external input as well as the component from the previous state. The following recursive equations shows how the gates and states are computed.

$$\begin{cases} \boldsymbol{i}^{(t)} = \sigma(\boldsymbol{W}_{xi}\boldsymbol{x}^{(t)} + \boldsymbol{W}_{hi}\boldsymbol{h}^{(t-1)} + \boldsymbol{W}_{ci}\boldsymbol{c}^{(t-1)} + \boldsymbol{b}_i) \\ \boldsymbol{f}^{(t)} = \sigma(\boldsymbol{W}_{xf}\boldsymbol{x}^{(t)} + \boldsymbol{W}_{hf}\boldsymbol{h}^{(t-1)} + \boldsymbol{W}_{cf}\boldsymbol{c}^{(t-1)} + \boldsymbol{b}_f) \\ \boldsymbol{c}^{(t)} = \boldsymbol{f}^{(t)} \odot \boldsymbol{c}^{(t-1)} \\ \qquad + \boldsymbol{i}^{(t)} \odot \tanh(\boldsymbol{W}_{xc}\boldsymbol{x}^{(t)} + \boldsymbol{W}_{hc}\boldsymbol{h}^{(t-1)} + \boldsymbol{b}_c) \\ \boldsymbol{o}^{(t)} = \sigma(\boldsymbol{W}_{xo}\boldsymbol{x}^{(t)} + \boldsymbol{W}_{ho}\boldsymbol{h}^{(t-1)} + \boldsymbol{W}_{co}\boldsymbol{c}^{(t-1)} + \boldsymbol{b}_o) \\ \boldsymbol{h}^{(t)} = \boldsymbol{o}^{(t)} \odot \tanh(\boldsymbol{c}^{(t)}) \end{cases} \tag{2}$$

In (2), the symbol $\odot$ denotes the Hardamard product operation, σ denotes the sigmoid function, $\boldsymbol{W}$ with the subscripts $_{\alpha\beta}$ denotes the weight matrix for transforming $\boldsymbol{\alpha}$ vector to a

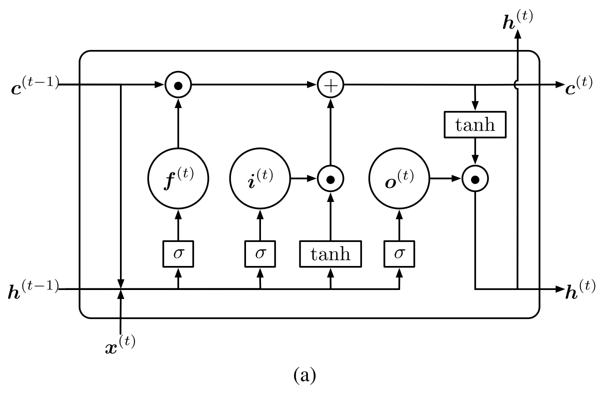
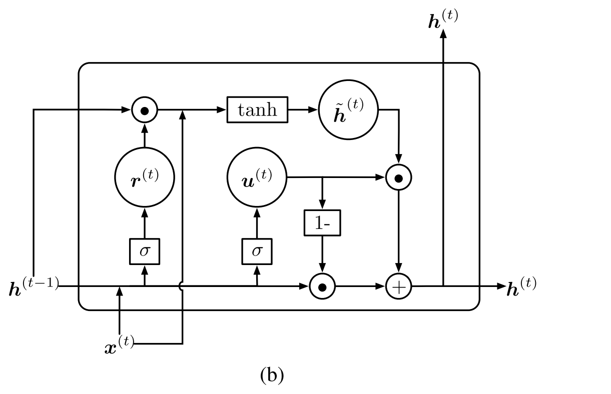


FIGURE 3. (a) LSTM cell, (b) GRU cell.



component of the $\boldsymbol{\beta}$ vector, and $\boldsymbol{b}$ with the subscript $_\gamma$ denotes the bias component of the vector $\boldsymbol{\gamma}$.

While the LSTM model has been shown to perform better than simple RNN in terms of the easiness of model training as well as the accuracy in classification results, the considerably increased number of parameters in LSTM usually incurs prolonged training process. The gated recurrent unit (GRU), proposed by Cho *et al.* [27], was designed with a two-gate structure to imitate the functionalities of the three gates in LSTM. Additionally, GRU also eliminates the explicit internal state in the LSTM cell. The recursive equations of GRU are the following:

$$\begin{cases} \boldsymbol{u}^{(t)} = \sigma(\boldsymbol{W}_{xu}\boldsymbol{x}^{(t)} + \boldsymbol{W}_{hu}\boldsymbol{h}^{(t-1)} + \boldsymbol{b}_u) \\ \boldsymbol{r}^{(t)} = \sigma(\boldsymbol{W}_{xr}\boldsymbol{x}^{(t)} + \boldsymbol{W}_{hr}\boldsymbol{h}^{(t-1)} + \boldsymbol{b}_r) \\ \tilde{\boldsymbol{h}}^{(t)} = \tanh(\boldsymbol{W}_{xh}\boldsymbol{x}^{(t)} + \boldsymbol{W}_{hh}(\boldsymbol{h}^{(t-1)} \odot \boldsymbol{r}^{(t)}) + \boldsymbol{b}_h) \\ \boldsymbol{h}^{(t)} = \boldsymbol{u}^{(t)} \odot \boldsymbol{h}^{(t-1)} + (\boldsymbol{1} - \boldsymbol{u}^{(t)}) \odot \tilde{\boldsymbol{h}}^{(t)}, \end{cases} \tag{3}$$

where the notations for the weight matrices and the biases are similar to those in (2), and the boldfaced $\boldsymbol{1}$ denotes a vector with all elements equal to 1. In (3), the vector $\boldsymbol{u}^{(t)}$ decides how much the cell updates its hidden state with the external input, and $\boldsymbol{r}^{(t)}$ the controls how much the previous state will be preserved. It has been shown that GRU provides comparable (and in some cases better) performances to LSTM [28].

B. CLASSIFIER ARCHITECTURE

The proposed classifier for power system event and intrusion detection adopts a basic architecture for RNN, which consists of an input layer, a number of hidden recurrent layers, and an output layer. The input layer accepts the preprocessed features as the input vector. As a result, the number of features in each time sample equals the dimension of the input layer. The number of hidden layers and the number of recurrent units in each hidden layer are two of the design parameters of this architecture. Generally, larger number of hidden layers and larger number of recurrent units would improve the accuracy of the classifier but require more computing power at the same time. As a result, it is important to find a set of acceptable parameters that provides sufficiently high accuracy and consumes reasonable time for model training and classification on the targeted hardware platform. Finally, the number of units in the output layer should be set to the number of classes in the datasets.

In addition, the length of the input sequences, which we call the *step size*, is another important architectural parameter for the RNN-based classifiers. The step size affects the accuracy of the classifier, as well as the execution time for training and testing the RNN models.

IV. METHODOLOGIES

A. POWER SYSTEM ATTACK DATASETS

In this paper, we evaluate the performances of the three RNN models for power system contingencies classification with a set of open data, which can be found as the Dataset 1 on [29]. This dataset was collected from a 3-bus 2-generator power system testbed, which consists of 4 breakers (BR1–BR4) controlled by the intelligent electronic devices (IEDs), i.e., the relays R1–R4 respectively (see Fig. 4). The IEDs employ a distance protection scheme which trips the breakers automatically upon the detection of faults, whether true or faked. Manual commands can also be sent to the IEDs to trip the breakers when conducting maintenance on the power system. Each IED includes an embedded phasor measurement unit (PMU) to provide real-time measurements of the electrical signals, e.g., the magnitudes, phase angles, and frequencies of the voltages and currents.

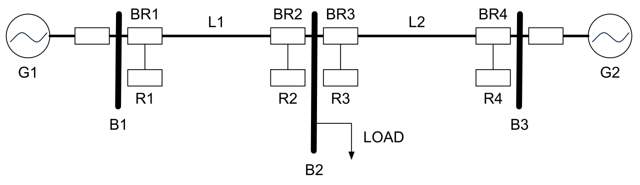


FIGURE 4. 3-bus 2-generator power system.

According to the authors, this testbed is capable of varying generation from two sources, as well as simulating various faults, control actions and cyber-attacks that result in loss of the transmission lines. Therefore, the datasets generated by this testbed is adequate for assessing the performances of machine learning based power system contingency classification schemes.

In the dataset, the first 128 columns are features and the last column is the 'marker' of the event. The synchrophasor technology of the PMU allows for time-synchronized measurements from distant IEDs. In the testbed, each PMU produces 29 types of measurements at each sampling time (see Table 1), and accordingly the measurements from 4 PMUs comprise 116 columns of the datasets. The remaining 12 columns of the datasets consists of logs from the IEDs and the control panel, as well as alerts from Snort. Since the testbed focuses on simulating power system related cyber-attacks, e.g., command injection attacks, the alerts generated by Snort only signify the detection of remote trip command activities in the network. Please refer to [29] for more details about the meanings of the features.

TABLE 1. Features extracted from PMU.

Feature	Description
PA1:VH – PA3:VH	Phase A-C Voltage Phase Angle
PM1:V – PM3:V	Phase A-C Voltage Phase Magnitude
PA4:IH – PA6:IH	Phase A-C Current Phase Angle
PM4:I – PM6:I	Phase A-C Current Phase Magnitude
PA7:VH – PA9:VH	Pos-Neg-Zero Voltage Phase Angle
PM7:V – PM9:V	Pos-Neg-Zero Voltage Phase Magnitude
PA10:IH – PA12:IH	Pos-Neg-Zero Current Phase Angle
PM10:I – PM12:I	Pos-Neg-Zero Current Phase Magnitude
F	Frequency for relays
DF	Frequency Delta (dF/dt) for relays
PA:Z	Apparent Impedance seen by relays
PA:ZH	Apparent Impedance Angle seen by relays
S	Status Flag for relays

The open dataset consists of 15 sub-datasets, each containing selected time samples that cover 37 power system event scenarios on the same network. The 37 scenarios can be further categorized into three groups, namely the Normal events, the Natural events, and the Attack events. The Normal event means the system operates normally and the Natural events include regular line maintenance and non-attack power line faults. The Attack events are divided into three types, data injection, remote tripping command injection, and relay setting change.

The original dataset provides three different labeling schemes for each of the 15 sub-datasets, specifically 2-class, 3-class, and multiclass labeling schemes. In the 2-class, i.e. binary, sub-datasets, the Natural and Normal events are labeled with the text "Normal" and the Attack events are labeled with the text "Attack". In the 3-class sub-datasets, the events are labeled with their group names. The multiclass sub-datasets are in ARFF format and the events are assigned with the numbered labels shown in Table 2.

In our experiments, we tested the performances of the classifiers on the binary and the multiclass schemes of the dataset. Specifically, classifying binary datasets only distinguishes whether the system is in Normal state or under cyber-attack.

TABLE 2. Event scenarios of the power system attack dataset.

Group	Types of scenarios	Multiclass labels	2-class labels
Normal	Normal operation	41	Normal
Natural	Single line-to-ground faults	1–6	
	Line maintenance	13, 14	
Attack	Data injection	7–12	Attack
	Remote tripping command injection	15–20	
	Relay setting change	21–30, 35–40	

On the other hand, classifying multiclass datasets further identifies the exact type of attack or the type of power line faults.

B. DATA PREPROCESSING

The first step of data preprocessing is to replace the measurements of the PMUs with their z-scores in both the binary and multiclass datasets. During this computation, we found that the impedance measurements of the PMUs contains `Inf` values, which would result in a whole column of `NaN` z-scores. Accordingly, we decided to remove the four impedance measurements since the impedance values can be derived from the voltage and current measurements. The 12 log columns only contain values of 0 and 1, so they were kept untouched during the preprocessing.

The second step is to replace the labels with one-hot encoded vectors. As a result, the labels of the binary and multiclass datasets become vectors with 2 and 37 elements respectively.

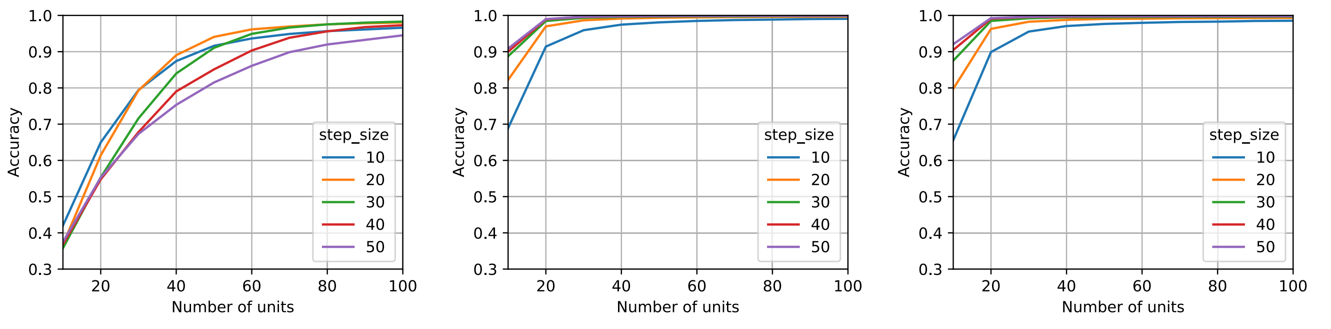
Since the RNN models require sequences of inputs to investigate the temporal dependencies of the time samples, the final step of the preprocessing procedure is to construct sequences from the original datasets. In order to construct sequences of length s, we join s consecutive time samples into a sequence and attach the sequence with the label of the time last sample. As a result, from a dataset with T samples, we construct a set of $(T-s+1)$ labeled sequences. The length of the sequences is an important parameter that affects the learning efficiency and running time of the RNN models.

C. EVALUATION METRICS

We adopt four commonly used metrics, namely accuracy, precision, recall, and F1-score, for evaluating the performances of the classification algorithms in this work. Generally, the result of a classification can be categorized into four possible conditions: true positive, true negative, false positive, false negative. If we denote the numbers of these four conditions by TP, TN, FP, FN, respectively, the four metrics can be calculated by the following equations.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{4}$$

$$Precision = \frac{TP}{TP + FP} \tag{5}$$



(a) SimpleRNN (b) LSTM (c) GRU

FIGURE 5. Multiclass accuracy (averaged over the 15 sub-datasets) of the three RNN-based classifiers with varying hyperparameters.

$$Recall = \frac{TP}{TP + FN} \tag{6}$$

$$F1 - score = \frac{2(Precision \times Recall)}{Precision + Recall} \tag{7}$$

Accuracy is the ratio of the correct decisions to the total number of test records, and is usually used as an indication of the overall performances of the system. Precision and recall are the ratios of the correct decisions to the number of the positive decisions and to the number of the actual positives respectively. For intrusion detection, precision represents the percentage of a real attack every time the IDS declares an attack. On the other hand, recall represents the ability of an IDS to detect real attacks. Examining precision or recall alone could be misleading when evaluating a classification system. For example, in the extreme case, if an IDS declares ''attack'' for each test record, the recall score will be equal to 1 since there is no false negative. However, this could result in a large number of false positives and thus lowering the precision of the IDS.

The F1-score, which is the harmonic mean of precision and recall, is also a score between 0 and 1. Therefore, in order to attain high F1-score, a classification system must perform well both in precision and recall.

V. EXPERIMENTS AND DISCUSSIONS

In this section, we present experiment results of training and testing three RNN-based classifiers, namely SimpleRNN, LSTM, and GRU, on two different magnitudes of datasets respectively. The first set of experiments is to evaluate the performance of the three RNN-based classifiers on the relatively small, original sub-datasets, each containing 4966–5569 time samples. In the second set of experiments, we test the effectiveness of the classifiers on large datasets by merging the 15 sub-datasets into a joint dataset with more than 77,000 time samples.

The RNN models are implemented in Python using the Keras API [30] running on top of TensorFlow [31]. The hardware environment for model training is a server machine with one Intel Xeon E5-2630 v4 CPU, 64GB RAM, and a NVIDIA RTX 2070 card. The operating system is Ubuntu 16.04.6 LTS and the installed versions of Python, Keras, TensorFlow, and CUDA are 3.6.9, 2.2.5, 1.14.0, and 10.1 respectively.

A. RESULTS ON ORIGINAL SUB-DATASETS

The first phase of the experiments is to search for optimal sets of hyperparameters for the RNN models. Based on the results of a set of preliminary experiments, we set the number of epochs to 200, and the batch size to 128. We choose the Adam optimizer [32] with learning rate set to 0.001 for training the RNN models. This setting of training hyperparameters produces stable results from the trained models and allows us to focus on the architectural hyperparameters.

For small datasets, we only employ one hidden layer for the RNN models since they already produce near optimum performance results. We use grid search to decide the combination of the number of units in each hidden layer and the step size of the time sequences that achieves the best accuracy for each of the RNN models respectively. The grid search takes steps of 10 when varying the two architectural hyperparameters. Fig. 5 illustrates partial results of the grid search for the best multiclass accuracy of the sub-datasets, and shows how the two architectural hyperparameters affect the performances of the three classifiers. For SimpleRNN, we have found that setting the step size to 30 would attain best accuracy provided the recurrent layer consists of sufficiently large number of hidden units. Increasing the step size over 30 could not improve the performance since the SimpleRNN model is incapable of learning from long sequences. On the contrary, increasing the step size for LSTM and GRU improves their performances until the accuracy reaches over 99.9% and saturates.

Table 3 shows the grid search results for both the 2-class and multiclass schemes. The hyperparameters in Table 3 are applied in the 10-fold cross-validation to evaluate the performance metrics for each sub-dataset, and the average values of the 15 sub-datasets for 2-class and multiclass schemes are presented in Table 4 and 5 respectively. In order for comparison, we also tested a set of machine learning based classifiers, namely Naive Bayes, Decision Tree, KNN, Logistic Regression, Random Forest, and SVM, on the same

TABLE 3. Architectural hyperparameters for best accuracy on the original sub-datasets.

	2-class			Multiclass		
	SimpleRNN	LSTM	GRU	SimpleRNN	LSTM	GRU
Step size	20	40	50	30	160	200
Number of units	150	90	80	300	150	150
Number of layers	1	1	1	1	1	1

15 sub-datasets. We ran the `scikit-learn` implementation of the above classifiers, and the averaged results of the same 10-fold cross-validation are included in Table 4 and 5 as well. The results of the machine learning based classifiers are comparable to the performances in [12].

The three RNN-based classifiers significantly outperform the other machine learning classifiers by achieving over 99% accuracy in both binary and multi-class scenarios. According to the standard deviations (SD) of the accuracy listed in Table 4 and 5, it is clear that the RNN-based classifiers constantly perform well on the 15 sub-datasets.

TABLE 4. Performance comparisons of the RNN-based classifiers with other machine learning algorithms on small 2-class datasets (The 100% figures with asterisk in this table are the outcomes of rounded numbers. They do not represent 100 percent correct classification results.)

	Accuracy(SD)	Precision	Recall	F1-score
Naive Bayes	31.26% (3.81%)	0.5444	0.5062	0.2635
Decision Tree	61.30% (4.09%)	0.5306	0.5299	0.5257
KNN	61.72% (4.18%)	0.5261	0.5251	0.5224
Logistic Regression	66.76% (5.57%)	0.5426	0.5128	0.4882
Random Forest	64.30% (4.99%)	0.5283	0.5200	0.5139
SVM	69.42% (5.23%)	0.5257	0.5116	0.4720
SimpleRNN	99.88% (0.05%)	0.9987	0.9984	0.9985
LSTM	**100.00% (0.01%)***	**1.0000***	**1.0000***	**1.0000***
GRU	99.99% (0.01%)	0.9998	0.9999	0.9999

TABLE 5. Performance comparisons of the RNN-based classifiers with other machine learning algorithms on small multiclass datasets.

	Accuracy(SD)	Precision	Recall	F1-score
Naive Bayes	8.23% (1.53%)	0.0811	0.1250	0.0658
Decision Tree	53.45% (1.97%)	0.5828	0.5776	0.5462
KNN	36.28% (1.70%)	0.3992	0.3967	0.3647
Logistic Regression	25.48% (2.91%)	0.2244	0.2297	0.2036
Random Forest	61.78% (2.23%)	0.6885	0.6761	0.6466
SVM	15.56% (2.91%)	0.1221	0.1238	0.1037
SimpleRNN	99.17% (0.12%)	0.9917	0.9885	0.9894
LSTM	**99.98% (0.02%)**	**0.9997**	**0.9998**	**0.9997**
GRU	99.92% (0.04%)	0.9991	0.9988	0.9989

For the small sub-datasets, the LSTM always produces the best performance in all four metrics. It is noteworthy that the scores of 100% with asterisks in Table 4 are rounded numbers. They do not represent perfect classification results in all experiments of the LSTM classifier. In fact, there is one classification error that occurs in the validation of the 14-th sub-dataset using LSTM for 2-class classification.

B. DISCREPANCY BETWEEN THE SUB-DATASETS

Before merging the 15 sub-datasets, we examined the discrepancy of the sub-datasets via the following experiment. In each round, one sub-dataset is selected as the test set and the other 14 sub-datasets are merged into a set of training data. The LSTM classifier with the hyperparameters listed in Table 3 is applied on the multiclass datasets.

The results in Fig. 6 shows that the models trained with 14 merged sub-datasets can only predict events in the distinct sub-dataset with only 60% to 80% accuracy. This means that although the sub-datasets all contain randomly selected time samples of the same 37 power system event scenarios simulated on the same testbed, the number of the selected events in each sub-dataset is too low to be sufficiently representative for each single class of events. It also explains why merging the datasets comprises a more difficult problem for the classifier to learn, especially for the multiclass case.

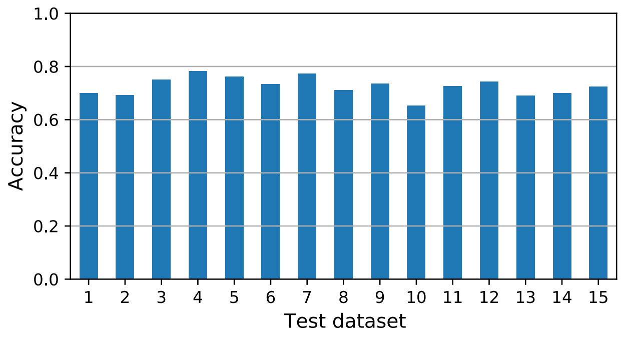


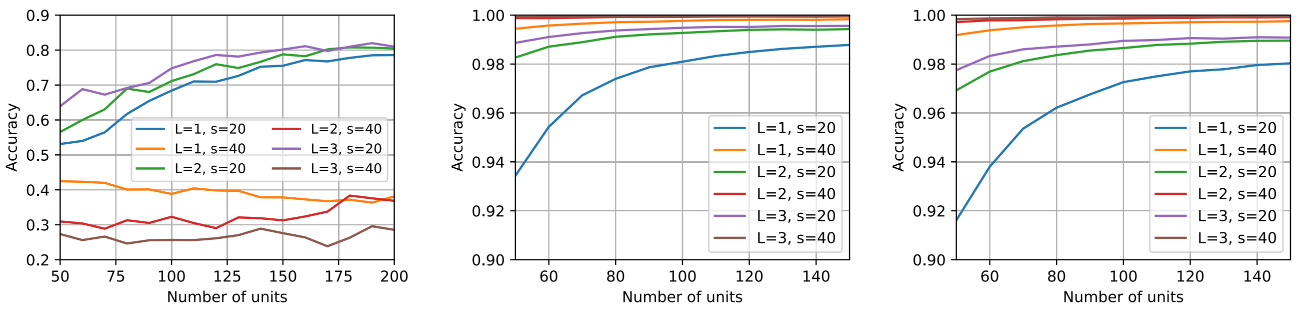
FIGURE 6. Discrepancy between the sub-datasets.

C. RESULTS ON MERGED DATASET

For the merged dataset, the same set of training hyperparameters as in Section V-A is used to conduct the grid search. The difference is that the number of hidden layers is considered in the multiclass case to boost the classification performance.

In Fig. 7, partial grid search results of the step sizes 20 and 40 are shown to demonstrate the trend of the classification accuracy for the merged multiclass dataset. The SimpleRNN model only achieves around 80% multiclass accuracy when the step size is set to 20, and gets poor results with larger step sizes. The reason is that the discrepancies among the events belonging to the same scenario in different sub-datasets require longer sequences to consolidate and learn the implicit information among them. Since the SimpleRNN cell lacks *long-term memory*, its failure to capture the long-term dependency in the time sequences results in the low accuracy in classifying the scenarios in the merged dataset. The experiment result in Fig. 7a also shows that increasing the number of hidden layers only deteriorates the accuracy of the SimpleRNN model when the step size equals 40.

On the contrary, one can boost the performance of LSTM and GRU classifiers and obtain over 99.99% accuracy by increasing the step size, the number of hidden layers, and the



(a) SimpleRNN (b) LSTM (c) GRU

FIGURE 7. Multiclass accuracy of the three RNN-based classifiers with varying hyperparameters on the merged dataset (L: number of hidden layers; s: step size).

number of units. It is noteworthy that the benefit of increasing the number of hidden layers quickly saturates when we increase the step size to 40. In fact, when the step size is set to 150 and the number of units is larger than 100, adding another hidden layer to the 2-layer LSTM and GRU classifiers only improves less than 0.01% accuracy.

Table 6 gives the grid search result of the architectural hyperparameters for best classification accuracy, and the 10-fold cross-validation metrics corresponding to the hyperparameters are compared with other machine learning classifiers in Table 7 and 8. The performance metrics of Adaboost+JRip, Common Path Mining and NNGE+STEM from [14] are included for comparison as well.

TABLE 6. Architectural hyperparameters for best accuracy on the merged dataset.

	2-class			Multiclass		
	SimpleRNN	LSTM	GRU	SimpleRNN	LSTM	GRU
Step size	20	50	60	20	150	150
Number of units	130	140	120	330	150	150
Number of layers	1	1	1	3	3	3

TABLE 7. Performance comparisons of the RNN-based classifiers with other machine learning algorithms on the merged 2-class datasets (The 100% figures with asterisk in this table are the outcomes of rounded numbers. They do not represent 100 percent correct classification results.)

	Accuracy	Precision	Recall	F1-score
Naive Bayes	30.91%	0.5523	0.5090	0.2630
Decision Tree	65.97%	0.5850	0.5854	0.5851
KNN	63.76%	0.5525	0.5509	0.5513
Logistic Regression	71.00%	0.5419	0.5054	0.4391
Random Forest	69.54%	0.5992	0.5717	0.5740
SVM	71.41%	0.6289	0.5052	0.4322
SimpleRNN	98.41%	0.9828	0.9783	0.9805
LSTM	**100.00%**	**1.0000**	**1.0000**	**1.0000**
GRU	100.00%*	1.0000*	0.9999	1.0000*
Adaboost+JRip	95%	0.95	0.91	0.90
Common Path Mining	96%	0.99	0.95	0.97
NNGE+STEM	98%	0.97	0.97	0.97

TABLE 8. Performance comparisons of the RNN-based classifiers with other machine learning algorithms on the merged multiclass datasets.

	Accuracy	Precision	Recall	F1-score
Naive Bayes	6.67%	0.0924	0.0387	0.0161
Decision Tree	17.06%	0.1416	0.1411	0.1399
KNN	11.92%	0.1071	0.1003	0.1012
Logistic Regression	18.55%	0.1283	0.1174	0.0957
Random Forest	18.11%	0.1504	0.1467	0.1460
SVM	17.82%	0.1227	0.1107	0.0891
SimpleRNN	82.81%	0.8287	0.8198	0.8218
LSTM	**99.99%**	**0.9999**	**0.9999**	**0.9999**
GRU	**99.99%**	**0.9999**	**0.9999**	**0.9999**
Adaboost+JRip	90%	0.85	0.80	0.81
Common Path Mining[a]	93%	0.98	0.95	0.96
NNGE+ STEM	93%	0.93	0.93	0.93

[a]The metrics of the common path mining were evaluated for 7 classes instead of 37. This is because the authors grouped common classes to improve its performance in [13].

For the merged 2-class dataset, SimpleRNN produces around the same level of performance as NNGE+STEM, which was known to have the best performance on the dataset. Surprisingly, LSTM perfectly classifies the merged 2-class dataset, and GRU gives a nearly perfect classification result. Considering the large amount of events in the merged dataset, such result provides us with high confidence in adopting the LSTM and/or GRU models for distinguishing abnormal behaviors of a power system.

For the merged multiclass dataset, LSTM and GRU still maintain higher than 99.99% accuracy. It proves the robustness and learning capability of LSTM and GRU classifiers in dealing with large datasets. On the other hand, the highest accuracy of SimpleRNN drops to 82.81%, which is unacceptable for critical contingency classification of a power system.

D. REDUCING THE FEATURE SET

In terms of the gained information for event and intrusion classification, there exist redundancy in the returned data from PMU. For example, the three phase voltages and currents are closely related to the symmetrical components (i.e. the Pos-Neg-Zero voltages and currents) in the way that each of them can be derived from the other using complex linear transformation [33]. The impedance data are also redundant since we already have the voltage and current phasors. Accordingly, we have done extensive experiments to test the performance of the LSTM and GRU classifiers on the merged multiclass dataset with reduced set of features.

Fig. 8 compares the accuracy of the LSTM and GRU classifiers on the feature sets listed in Table 9. Set 0 is the original full feature set. Each of the other feature sets includes a different subset of the voltage and/or current phasors. The other PMU data, including the frequency, the frequency delta, the impedance, and the status for the relay, are left out because they showed little or no significance in improving the classification accuracy. Furthermore, since the Snort alert in the dataset only contains one-bit information about whether there exists remote trip command packet or not, this feature is also excluded due to its lack of significance in our experiments. In this set of experiments, both the step size and number of units are set to 150, and the number of hidden layers vary from 1 to 3.

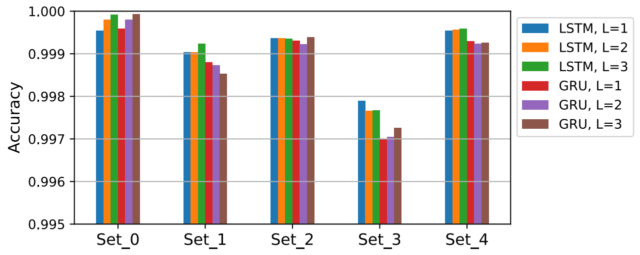


FIGURE 8. Performance comparison of LSTM and GRU classifiers on the merged multiclass dataset with reduced features (Step size = 150, Number of units = 150).

TABLE 9. Feature sets for performance comparison.

Feature	Set 0	Set 1	Set 2	Set 3	Set 4
Phase A-C voltage	√	√			
Phase A-C current	√	√			
Pos-Neg-Zero voltage	√		√	√	
Pos-Neg-Zero current	√		√		√
Other PMU data	√				
IED/Control panel logs	√	√	√	√	√
Snort alert	√				

First, the differences between Set 1 and Set 2 in Fig. 8 shows a small advantage of using the symmetrical components representation over the phase voltage and current in event classification. Second, from the performance results of Set 3 and Set 4, we find that the sequence currents of the symmetrical components carry more information than the sequence voltages. Yet another interesting finding is that increasing the number of hidden layers does not bring much accuracy improvement for the reduced feature datasets.

In summary, although the reduced feature sets all produce inferior performance in classifying the events, the experiment result proves that one can attain accuracy comparable to that of the full feature set if the features are chosen carefully. Take Set 4 as an example, as high as 99.96% classification accuracy can be attained using 32 features, which is only one fourth of the original 128 features. This result greatly reduces the amount of data that need to be transmitted and stored for EIDS, and reduces the computing time for model training and classification as well (see Section V-E). Moreover, it significantly slims down the memory requirement for model training, and this is crucial for accelerating the computation with GPUs due to their limited memory space.

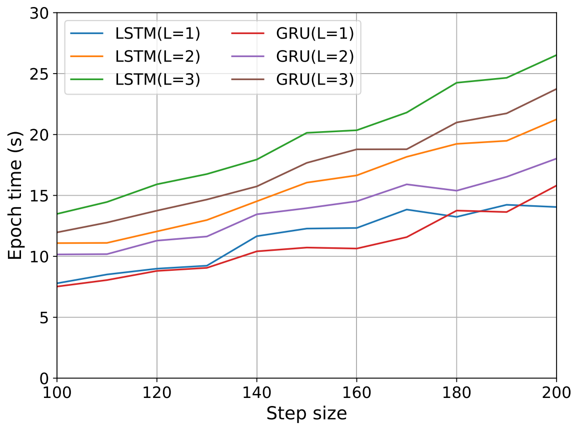
E. TRAINING AND CLASSIFICATION TIME

Next, we examine the epoch time of model training as well as the classification time to compare the efficiency of the classifiers in different setups. Fig. 9 illustrates the epoch time of the model training on our system for two datasets with full and reduced feature sets respectively. More specifically, Set 4 in Table 9 is compared with Set 0 since it achieves the highest accuracy among the reduced feature sets. We vary the step size and the number of hidden layers in this set of experiments, but the number of units in each hidden layer is set to 150 in all models. The statistic of each parameter setting in Fig. 9 is an average of 200 epochs. In average, the reduction in epoch time of LSTM model training due to the feature set reduction is 45.2%, 34.4%, 25.4% for L=1, L=2, L=3 respectively, and the reduction in epoch time of GRU is 48.1%, 34.4%, 25.4% respectively. The reason why the reduction ratio becomes lower as we increase the number of hidden layers is because the reduction in the number of features only reduces the nodes of the input layer.

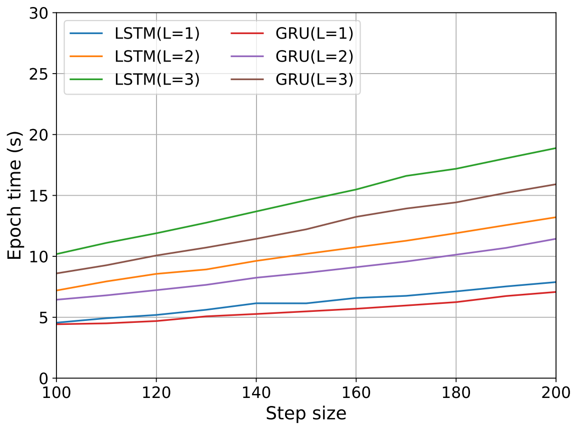
In practice, the importance of the model training time depends on the variation of the system. If the system changes very often and the model needs to be re-trained frequently, it would be crucial to keep the training time down by minimizing the model or raising the computing capability. On the other hand, if the system structure does not change, the model training time would be just an upfront cost of constructing the EIDS since there is no need to re-train the model.

Another fundamental factor that defines the efficiency of an EIDS is the classification time, which plays an important part in the event detection time. In order to evaluate the feasibility of employing economic hardware in the distributed EIDS framework, we test the classification time on the NVIDIA Jetson Nano board, an embedded platform designed for artificial intelligence (AI) applications. The hyperparameter settings are the same as the epoch time comparison in Fig. 9. Each trained model and its corresponding test dataset (each consists of around 7600 input sequences) were prepared by the GPU server and downloaded to the Jetson Nano to run 50 repetitions and compute the average classification time.

Fig. 10 compares the average classification time per input sequence of the merged multiclass Set 0 and Set 4. First of all, the classification time grows linearly with the step size when the other parameters are fixed, and the LSTM always runs slower since it has more parameters than GRU.

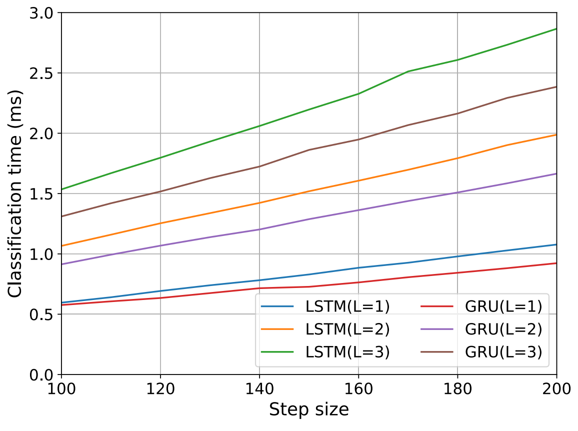


(a) Full feature set

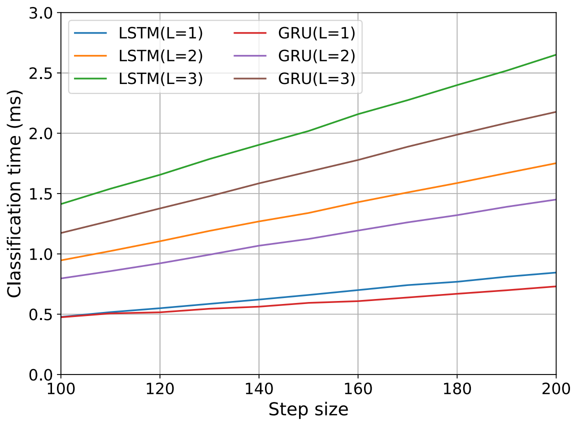


(b) Reduced feature (Set 4)

FIGURE 9. Epoch time of LSTM and GRU model training for the merged multiclass dataset with full/reduced feature sets (fixed number of units = 150).



(a) Full feature set



(b) Reduced feature (Set 4)

FIGURE 10. Classification time of LSTM and GRU on the merged multiclass dataset using Jetson Nano (fixed number of units = 150).

Second, the reduction in classification time is not as significant as the reduction in epoch time. In average, we observe 20.6%, 11.5%, 7.9% reduction in the classification time for L=1, L=2, L=3 LSTM models respectively, and 19.6%, 12.7%, 9.1% reduction for L=1, L=2, L=3 GRU models respectively. Again, the reduction ratio drops noticeably as the number of hidden layers increases. It is also noteworthy that increasing the number of hidden layers significantly increases the classification time on Jetson Nano due to the limited parallel computing power of the hardware. For the L=3 LSTM model, the average time of classifying the Set 4 data can be as high as 3.1 times the average time of the L=1 LSTM model.

According to the authors of [13] and [14], the PMUs in the testbed produces as many as 120 samples per second, which is the highest frequency of the PMU data. Therefore, if the EIDS can collect and process the measurements and logs within the 8.33 millisecond inter-measurement time, the exact data manipulated by the attacker can be pinpointed. Obviously, all results in Fig. 10 are well below the time frame. This provides ample margin for the application to other power systems larger than the 3-bus 2-generator scenario in Fig. 4 using this relatively low-end embedded system for AI computation. However, the tradeoff between the classification time and the accuracy when deciding the hyperparameters, especially the number of hidden layers, should be evaluated carefully.

VI. CONCLUSION

In this paper, we presented the evaluation of applying three RNN models to the classification of power system contingencies and cyber-attacks. Through extensive experiments on two different scales of datasets recorded from a cyber-physical testbed for power system, we have shown that well-trained LSTM and GRU classifiers are capable of identifying multiclass attacks and contingencies with nearly perfect accuracy, which are significant improvements over the previous methods in the literature. Although the testbed only simulates a relatively small system, the proposed design can be easily extended to larger power grids because of its simplicity and scalability. Moreover, we have demonstrated the capability of the proposed classifiers to produce nearly the same level of accuracy using the datasets with one fourth of the original number of features. The low detection latency of applying LSTM and GRU on the low-cost embedded platform also exemplifies the feasibility of the proposed method in the distributed EIDS framework, from which timely alerts can be sent to the control centers in the smart grid and prompt response measures can be taken to minimize the damage.

In future smart grid, the ADMS will evolve into an automated and reliable operating system of the distribution grid. Accurate and efficient classification of the contingencies and cyber-attacks will play an important part in the reliability of the ADMS and the smart grid. We believe combining the power of the deep learning techniques and the real-time measurements from the PMUs leads toward this objective.

It is worth mentioning that devising an accurate and efficient classifier only solves half of the problem. Another challenging task would be how to collect and/or generate sufficient site-specific labeled training data for all known attacks and contingencies in the operating power systems. This would lead to future research directions, such as building a scalable and valid testbed for emulating smart grid systems, or investigating unsupervised machine learning techniques for smart grid EIDS.

REFERENCES

[1] *Cyber-Attack Against Ukrainian Critical Infrastructure*. Accessed: Jul. 6, 2020. [Online]. Available: https://ics-cert.us-cert.gov/alerts/IR-ALERT-H-16-056-01

[2] H. He and J. Yan, "Cyber-physical attacks and defences in the smart grid: A survey," *IET Cyber-Phys. Syst., Theory Appl.*, vol. 1, no. 1, pp. 13–27, Dec. 2016.

[3] G. Liang, J. Zhao, F. Luo, S. R. Weller, and Z. Y. Dong, "A review of false data injection attacks against modern power systems," *IEEE Trans. Smart Grid*, vol. 8, no. 4, pp. 1630–1638, Jul. 2017.

[4] J.-W. Kang, I.-Y. Joo, and D.-H. Choi, "False data injection attacks on contingency analysis: Attack strategies and impact assessment," *IEEE Access*, vol. 6, pp. 8841–8851, 2018.

[5] J. Donahue, L. A. Hendricks, M. Rohrbach, S. Venugopalan, S. Guadarrama, K. Saenko, and T. Darrell, "Long-term recurrent convolutional networks for visual recognition and description," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 4, pp. 677–691, Apr. 2017.

[6] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Advances in Neural Information Processing Systems*, Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, Eds. Red Hook, NY, USA: Curran Associates, 2014, pp. 3104–3112. [Online]. Available: http://papers.nips.cc/paper/5346-sequence-to-sequence-learning-with-neural-networks.pdf

[7] A. Graves and N. Jaitly, "Towards end-to-end speech recognition with recurrent neural networks," in *Proc. 31st Int. Conf. Mach. Learn.*, in Proceedings of Machine Learning Research, vol. 32, no. 2, E. P. Xing and T. Jebara, Eds. Bejing, China: PMLR, Jun. 2014, pp. 1764–1772. [Online]. Available: http://proceedings.mlr.press/v32/graves14.html

[8] C. Yin, Y. Zhu, J. Fei, and X. He, "A deep learning approach for intrusion detection using recurrent neural networks," *IEEE Access*, vol. 5, pp. 21954–21961, 2017.

[9] J. Kim, J. Kim, H. L. T. Thu, and H. Kim, "Long short term memory recurrent neural network classifier for intrusion detection," in *Proc. Int. Conf. Platform Technol. Service (PlatCon)*, Feb. 2016, pp. 1–5.

[10] C. Xu, J. Shen, X. Du, and F. Zhang, "An intrusion detection system using a deep neural network with gated recurrent units," *IEEE Access*, vol. 6, pp. 48697–48707, 2018.

[11] M. Jamei, E. Stewart, S. Peisert, A. Scaglione, C. McParland, C. Roberts, and A. McEachern, "Micro synchrophasor-based intrusion detection in automated distribution systems: Toward critical infrastructure security," *IEEE Internet Comput.*, vol. 20, no. 5, pp. 18–27, Sep. 2016.

[12] R. C. B. Hink, J. M. Beaver, M. A. Buckner, T. Morris, U. Adhikari, and S. Pan, "Machine learning for power system disturbance and cyber-attack discrimination," in *Proc. 7th Int. Symp. Resilient Control Syst. (ISRCS)*, Aug. 2014, pp. 1–8.

[13] S. Pan, T. Morris, and U. Adhikari, "Classification of disturbances and cyber-attacks in power systems using heterogeneous time-synchronized data," *IEEE Trans. Ind. Informat.*, vol. 11, no. 3, pp. 650–662, Jun. 2015.

[14] U. Adhikari, T. H. Morris, and S. Pan, "Applying non-nested generalized exemplars classification for cyber-power event and intrusion detection," *IEEE Trans. Smart Grid*, vol. 9, no. 5, pp. 3928–3941, Sep. 2018.

[15] S. Golshannavaz, S. Afsharnia, and F. Aminifar, "Smart distribution grid: Optimal day-ahead scheduling with reconfigurable topology," *IEEE Trans. Smart Grid*, vol. 5, no. 5, pp. 2402–2411, Sep. 2014.

[16] P. I. Radoglou-Grammatikis and P. G. Sarigiannidis, "Securing the smart grid: A comprehensive compilation of intrusion detection and prevention systems," *IEEE Access*, vol. 7, pp. 46595–46620, 2019.

[17] T. H. Morris, B. A. Jones, R. B. Vaughn, and Y. S. Dandass, "Deterministic intrusion detection rules for MODBUS protocols," in *Proc. 46th Hawaii Int. Conf. Syst. Sci.*, Jan. 2013, pp. 1773–1781.

[18] H. Li, G. Liu, W. Jiang, and Y. Dai, "Designing snort rules to detect abnormal DNP3 network data," in *Proc. Int. Conf. Control, Automat. Inf. Sci. (ICCAIS)*, Oct. 2015, pp. 343–348.

[19] Q. He and R. S. Blum, "Smart grid monitoring for intrusion and fault detection with new locally optimum testing procedures," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, May 2011, pp. 3852–3855.

[20] R. Khan, A. Albalushi, K. McLaughlin, D. Laverty, and S. Sezer, "Model based intrusion detection system for synchrophasor applications in smart grid," in *Proc. IEEE Power Energy Soc. Gen. Meeting*, Jul. 2017, pp. 1–5.

[21] M. A. Faisal, Z. Aung, J. R. Williams, and A. Sanchez, "Data-stream-based intrusion detection system for advanced metering infrastructure in smart grid: A feasibility study," *IEEE Syst. J.*, vol. 9, no. 1, pp. 31–44, Mar. 2015.

[22] S. Revathi and A. Malathi, "A detailed analysis on NSL-KDD dataset using various machine learning techniques for intrusion detection," *Int. J. Eng. Res. Technol.*, vol. 2, no. 12, pp. 1848–1853, Dec. 2013.

[23] S. Naseer, Y. Saleem, S. Khalid, M. K. Bashir, J. Han, M. M. Iqbal, and K. Han, "Enhanced network anomaly detection based on deep neural networks," *IEEE Access*, vol. 6, pp. 48231–48246, 2018.

[24] A. Diro and N. Chilamkurti, "Leveraging LSTM networks for attack detection in fog-to-things communications," *IEEE Commun. Mag.*, vol. 56, no. 9, pp. 124–130, Sep. 2018.

[25] P. J. Werbos, "Backpropagation through time: What it does and how to do it," *Proc. IEEE*, vol. 78, no. 10, pp. 1550–1560, Oct. 1990.

[26] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997, doi: 10.1162/neco.1997.9.8.1735.

[27] K. Cho, B. van Merrienboer, D. Bahdanau, and Y. Bengio, "On the properties of neural machine translation: Encoder-decoder approaches," Sep. 2014, *arXiv:1409.1259*. [Online]. Available: http://arxiv.org/abs/1409.1259

[28] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," Dec. 2014, *arXiv:1412.3555*. [Online]. Available: http://arxiv.org/abs/1412.3555

[29] U. Adhikari, S. Pan, T. Morris, R. Borges, and J. Beaver. *Industrial Control System (ICS) Cyber Attack Datasets, Dataset 1: Power System Datasets*. Accessed: Jul. 6, 2020. [Online]. Available: https://sites.google.com/a/uah.edu/tommy-morris-uah/ics-data-sets

[30] F. Chollet. (2015). *Keras*. [Online]. Available: https://keras.io

[31] M. Abadi. (2015). *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. [Online]. Available: http://tensorflow.org/

[32] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," Dec. 2014, *arXiv:1412.6980*. [Online]. Available: http://arxiv.org/abs/1412.6980

[33] C. L. Fortescue, "Method of symmetrical co-ordinates applied to the solution of polyphase networks," *Trans. Amer. Inst. Electr. Eng.*, vol. 37, no. 2, pp. 1027–1140, Jul. 1918.

WEI-CHIH HONG (Member, IEEE) received the B.S. and M.S. degrees in electrical engineering and the Ph.D. degree in communications engineering from National Taiwan University, Taipei, in 1996, 1998, and 2010, respectively.

From 1999 to 2003, he has worked as an Assistant Researcher with the Telecommunication Laboratories, Chunghwa Telecom. From 2010 to 2014, he conducted postdoctoral researches with Technische Universität Darmstadt, Germany, and Academia Sinica, Taiwan. Since 2014, he has been an Assistant Professor with the Department of Information Engineering and Computer Science, Feng Chia University, Taichung, Taiwan. His current research interests include cryptographic engineering, side channel analysis, and the application of machine learning techniques in information security.

DING-RAY HUANG received the B.S. degree in information engineering and computer science from Feng Chia University, Taichung, Taiwan, in 2017, where he is currently pursuing the M.S. degree in computer science. His research interests include machine learning and the application of distributed systems.

CHIH-LUNG CHEN is currently pursuing the M.S. degree in information engineering and computer science with Feng Chia University, Taichung, Taiwan. His current research interests include network security and machine learning.

JUNG-SAN LEE received the Ph.D. degree in computer science and information engineering from National Chung Cheng University, Chiayi, Taiwan, in 2008. Since 2017, he has been a Professor with the Department of Information Engineering and Computer Science, Feng Chia University, Taichung, Taiwan. His current research interests include network management, electronic commerce, and blockchain.