Learning-based Simultaneous Detection and Characterization of Time Delay Attack in Cyber-Physical Systems

Prakhar Ganesh, Xin Lou, Yao Chen, Rui Tan, *Senior Member, IEEE,* David K.Y. Yau, *Senior Member, IEEE,* Deming Chen, *Fellow, IEEE,* Marianne Winslett

***Abstract*—Control and communication technologies are key building blocks of cyber-physical systems (CPSes) that can improve the efficiency of the physical processes. However, they also make a CPS vulnerable to cyberattacks that can cause disruptions or even severe damage. This paper focuses on one particular type of CPS cyberattack, namely the time delay attack (TDA), which exploits vulnerabilities in the communication channels to cause potentially serious harm to the system. Much work proposed for TDA detection is tested offline only and under strong assumptions. In order to construct a practical solution to deal with real-world scenarios, we propose a deep learning based method to detect and characterize TDA. Specifically, we design a hierarchical long short-term memory model to process raw data streams from relevant CPS sensors online and continually monitor embedded signals in the data to detect and characterize the attack. Moreover, various strategies of interpreting the outputs of the model are proposed, which allow the user to tune the performance based on different objectives. We evaluate our model on two representative types of CPS, namely power plant control system (PPCS) and automatic generation control (AGC)[1]. For TDA detection, our solution achieves an accuracy of 92% in PPCS, compared with 81% by random forests (RFs) and 72% by k-nearest neighbours (kNNs). For AGC, our solution achieves 98% accuracy, compared with 74% by RFs and 71% by kNNs. It also reduces the mean absolute error in the delay value characterization from about six to two seconds in the PPCS, and from about three seconds to half a second in the AGC, with about 3x to 4x shorter reaction latency in both systems.**

***Index Terms*—Smart grid; cyber-physical system; time delay attack; attack detection; attack characterization; deep learning**

I. Introduction

A cyber-physical system (CPS) exploits information and communication technologies (ICT) to manage the dynamics of complex physical systems and improve operation efficiency and agility [1]. However, the very introduction of ICT makes them vulnerable to various cyberattacks [2]. A common defense strategy relies on air gaps or firewalls to keep intrusions at bay. However, the need for more resilient solutions than perimeter defense is clearly evidenced by various recent intrusions such as stepping stone attacks [3] and insider attacks [4]. In general, strategic attackers can breach the defense parameter to disrupt, sabotage, or harm mission-critical systems such as industrial control [5]–[7].

Well studied examples of effective CPS cyberattacks include false data injection (FDI) and denial of service (DoS) [8], [9]. FDI attacks corrupt key data in transmission. Generally, they require the attacker to breach sophisticated cryptographic protection of relevant data packets [8]. On the other hand, DoS attacks may flood relevant transmission channels with bogus traffic to prevent legitimate data packets from going through. They can be achieved without breaking any cryptographic protection to effect nevertheless powerful impacts such as system shutdown [9]. It can be relatively easy to detect and mitigate – e.g., using a rate-limiting firewall – a DoS attack that operates by brute force [10].

Compared with the above attacks, the time delay attack (TDA) [11] is arguably more challenging to deal with. The TDA simply delays (maliciously) data packets in transmission. Unlike the FDI attack, it does not require any parsing or modification of the packet content. Also unlike a flooding DoS attack, a carefully launched TDA may not obviously affect the pattern of traffic in transmission, which makes it difficult to detect. However, TDAs can cause great harm in a CPS; e.g., closed-loop control in mission-critical systems (e.g., power grids) depends critically on timely feedback to adapt its operations accurately in real time. TDAs are capable of affecting both the safety and the stability of the system and can cause considerable damage if left undefended [12], [13]. The effects of TDAs have been studied extensively in the literature [12]–[16]. In principle, high-precision clock synchronization can help secure against the TDA. However, secure clock synchronization is highly non-trivial; it is itself prone to various known cyberattacks even if performed over encrypted channels [17]. Thus, it is imperative to devise a defense-in-depth solution to address the TDA resiliently, in an orthogonal manner to any secure clock synchronization that might be available.

In this paper, we solve the TDA detection and characterization problems without assuming secure clock synchronization. In the case of positive detection, the characterization further estimates the malicious delay. Existing solutions for detecting and characterizing the TDA have the following major drawbacks. (i) As a real-world CPS can be highly complex, it in general precludes accurate system modeling and requires approximation or modeling under controlled environment settings [18], [19]; this issue causes difficulties for solutions that require such modeling to expose the effects of any TDAs [20]–[26]. (ii) Long streams of continuously generated time-series

P. Ganesh, X. Lou and Y. Chen are with Advanced Digital Sciences Center, Illinois at Singapore (e-mail: prakhar.g@adsc-create.edu.sg, lou.xin@adsc-create.edu.sg, yao.chen@adsc-create.edu.sg). R. Tan is with Nanyang Technological University, Singapore (email : tanrui@ntu.edu.sg). D.K.Y. Yau is with Singapore University of Technology and Design (email : david_yau@sutd.edu.sg). D. Chen and M. Winslett are with University of Illinois at Urbana-Champaign, USA (email : dchen@illinois.edu, winslett@illinois.edu)

[1]Code and dataset can be found at :https://github.com/prakharg24/tda

data in typical systems cannot be handled by commonly used sequential data analysis methods like simple RNNs [27], which make it hard for existing methods to analyze required features of the attack for good performance [28], [29]. (iii) The need for timely online detection and characterization to mitigate an attack's impact in real-time precludes offline (postmortem) analysis of collected data sequences that last until the attack is long gone [12], [13], [20]–[23], [28], [30]–[32].

To overcome the aforementioned challenges, in this paper we propose a deep learning (DL) based approach, which directly uses data traces from relevant sensors to learn requisite key features of the system to expose any ongoing TDAs. Being data driven, it removes the need to model *a priori* the (likely complex) CPS accurately and comprehensively. It has the following key desirable features:

Robustness: The proposed feature extraction and delay estimation are robust, in that they allow accurate predictions from sensor data streams even in the presence of realistic measurement noises.

Continuous online monitoring: The DL refines information available incrementally in progressive data streams to catch a TDA as early as possible and thereby enable timely responses and mitigation, as opposed to the postmortem analysis of signals commonly found in existing work.

Integrated detection-characterization learning model: The proposed hierarchical long short-term memory model admits simultaneous detection and characterization synergistically, i.e., sharing the same backbone learning features between the two tasks, which has higher computational efficiency than independent solutions for the two respective problems.

User-centric tunable interpretation: The proposed method admits a spectrum of interpretations for the model's outputs, among which the user can make specific choices to prioritize different performance aspects such as *reaction latency* (i.e., delay from the attack's start to the detection and characterization results) and overall accuracy. This tunability of our solution is independent of the DL model training; it thus does not require any form of retraining for the same.

We apply the proposed solution to TDAs against a power plant control system (PPCS) and a power grid automatic generation control system (AGC) to evaluate its performance. These systems are representative examples of real-world mission-critical CPSes under closed-loop control. Diverse evaluation results show that our solution can efficiently and accurately detect an ongoing TDA and characterize the delay value using online data streams from relevant sensors. Comparison results with conventional learning-based solutions including k-nearest neighbor (kNN), random forest (RF), and vanilla LSTM substantiate the advantages of our solution within the state of the art.

The rest of the paper is organized as follows. Section II reviews related work. Section III introduces our system and attack models. Section IV presents the proposed learning-based solution to the TDA detection and characterization problems. Section V presents extensive evaluation results to illustrate our solution and compare its performance with other machine learning based solutions. Section VI concludes.

II. Related Work

CPS anomaly detection is an active area of research [33], [34]. Much previous work addresses the monitoring of control centers, without emphasizing related communication channels, which can be comparatively easier to exploit [11]. Machine learning-based methods have been recently applied for the anomaly detection with promising results. Advances in this approach have been reviewed [35]; typical examples employ conventional machine learning algorithms like support vector machine (SVM) [31] and DL models like LSTMs [30], [32] and generative adverserial networks (GANs) [28]. In [31], the authors compare two models, a one-class SVM and a one-layer LSTM, both working on anomaly detection in the context of a water treatment testbed. Under the same test setup, prior work [32] has used LSTM in conjunction with cumulative sum to detect anomalies. A GAN method has also been proposed [28], in which the GAN discriminator is trained using actual data samples and generator resconstructed data is used to detect possible anomalies. This aforementioned prior work does not consider TDA in the anomaly detection.

For the detection and characterization of TDA specifically in CPSes, existing work can be divided into two basic approaches: model-driven and data-driven.

Model-driven approaches focus on system modeling using complex mathematical models [20]–[26], [36]. For example, a modified controller with a built-in time-delay estimator has been proposed [23] for a continuous linear time-invariant system [36]. Also, multi-model internal controls have been decoupled [24] in order to create an approximation of the subject system. Their work assumes that there are no interactions between various partial systems present in the CPS. In [25], systems in which the signals being transmitted satisfy monotonicity and derivability properties are considered. In [26], recurrent least square methods are used to perform delay attack detection. However, their work assumes that the delay attack is introduced gradually into the system, and it can only detect TDAs of specific delay values, namely the pre-defined ones in the model settings. These model-driven methods require to create models for the target CPSes, a challenging if not impossible task for often highly complex and heterogeneous systems in practice.

Data-driven approaches, on the other hand, do not rely on any modeling assumptions and are often designed to be robust to noisy inputs. They thus show promise for being generalizable to diverse real-world situations. A limited number of studies exist that use a data-driven approach for TDA detection and characterization. In [30], a method of anomaly detection is proposed that analyzes the content of relevant data packets. They first create a feature representation, or "signature," for the baseline behavior of normal network packets. They then provide this signature to a Bloom filter for flagging anomalous network packets. Because it depends on a thorough history of the network behavior of every packet, it is a postmortem analysis that is only applicable to systems that have been traced extensively under comprehensive operational scenarios. In [37], a neural network model of system state variables is used to estimate TDA, in a hybrid attempt that combines

both the model-driven and data-driven approaches. The model-driven component of the solution [37] still needs a good enough approximation of the system to define the system state.

Another data-driven method has used a simple LSTM network [29] to characterize the TDA. However, they make several assumptions that may not be generally true of real-world scenarios. For example, the presence of an TDA is assumed known a priori, so that the detection problem is left unaddressed. In this paper, we build on the prior work [29] to achieve a DL-based solution with improved practicality and completeness.

III. TDA in Cyber-Physical Systems

In this paper, we address TDAs against PPCS and AGC in power grids. These systems are representative of mission-critical closed-loop CPS control that is widely deployed in the real world. Before detailing our solution, in the following, we first define the system and attack models.

A. System Model

We model a discrete-time closed-loop CPS control system. It consists of sensors, actuators, and controllers (e.g., PLCs). The sensors measure the system state, which is used as input for the controllers to determine their control decisions. The controllers transmit the corresponding commands to the actuators, which execute the commands and change the system state accordingly, thus completing the closed loop. The system state is subject to various disturbances, such as measurement noises, control setpoint changes, etc.

A typical PPCS structure is shown in Fig. 1, which is from ThermoPower [38], an open-source library based on OpenModelica [39]. As shown in Fig. 1, the power plant has three inputs: the power control (PC) signal, the gas flow rate (GR) control signal, and the void fraction (VF) control signal. Both PC and VF are maintained by a proportional-integral-derivative (PID) control algorithm, whereas GR is given directly as a user input setpoint. A typical PID controller compares the difference between the user input setpoint and the measured value to find an accurate and responsive correction to the control. The PPCS monitors its power generation, but the measurements are subject to additive random noises existing in the PPCS.

In power grids, AGC maintains the system frequency at a nominal value (e.g., 60Hz) by adjusting setpoints of generators. It also maintains the net power interchanges among neighboring areas at scheduled values [40], where the in-grid area is usually operated by a utility and different areas are connected by tie-lines. A typical AGC structure is shown in Fig. 2, which we have modeled using the Powerworld simulator [41] and contains generators participating in the AGC. The AGC controller receives, over a communication network, tie-line flow measurements of each area's power export from their respective setpoints, as well as the power system's frequency, and computes the area control error (ACE). For the ith area, $ACE_i = \alpha_i \cdot \Delta P_{E_i} + \beta_i \cdot \Delta f_i$, where α_i and β_i are constants, ΔP_{E_i} and Δf_i are the ith area's power export from their respective setpoints and deviations of the grid frequency

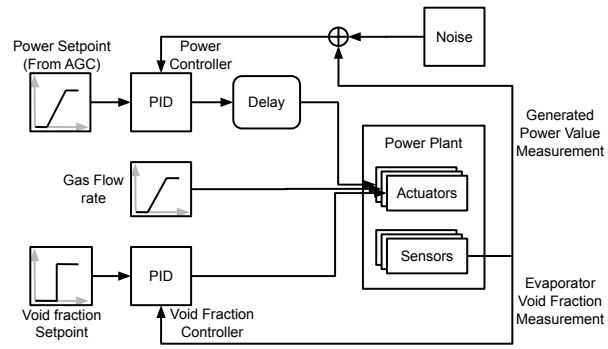


Fig. 1: Overview of PPCS system model with TDA.

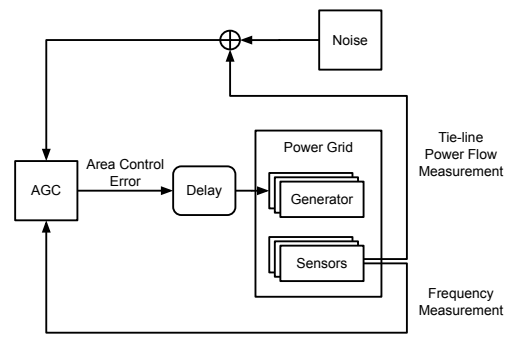


Fig. 2: Overview of AGC system model with TDA. The delay module attacks the ACE in one of the generators in the grid.

from the standard frequency, respectively. The control center sends the ACE back to the power grid generators to adjust the setpoints of the generators' primary control loop. In the system, the tie-line flow measurements are subject to additive random noises existing in the power grid.

The above process is repeated every AGC cycle, which is often two to four seconds. For both the PPCS and AGC, when there is no malicious delay in the system, the control command Y^k received at the power plant at time slot t can be defined as:

$$Y^k[t] = C^k[t], \tag{1}$$

where C^k respectively specifies one of the three control signal types, for $k \in (PC, GR, VF)$, in the PPCS. For the AGC, the ACE is sent by the controller to generators in the relevant control areas, i.e., $k \in (ACE_1, ACE_2, ..ACE_m)$ where m is the number of areas in the system.

B. Attack Model

Similar to prior work [12], [13], [29], the adversary compromises the communication path (e.g., through a malware-infected router) between the controller and the actuator to delay the transmission of control commands to the actuator, i.e., $Y^k[t]$ (refer to the delay module in Figs. 1 and 2). An example of the same is given in [26], where a traffic shaping virtual machine (VM) is used to create dummy bridges that forces the signal to loop through such bridges and cause signal delays. More generally, even in the absence of malicious attacks, anomalous delays might enter the system due to natural faults or failures, although we will focus on the (more

challenging) attack context in this paper. Note that malicious delays can cause instability in the system even when the number of compromised channels is limited. Assuming that a TDA happens at time slot $t=\bar{t}$ and the attacker delays the controller commands by τ seconds ($\tau=0$ is no attack), the input command signal $Y^k[t]$ received at the power plant under attack becomes

$$Y^k[t]=\begin{cases} C^k[t] & t<\bar{t} \\ C^k[t-\tau] & t\geq\bar{t}. \end{cases} \tag{2}$$

Note that, as shown in (2), the adversary does not need to modify the content of the control command in the TDA. Since the TDA occurs in a closed-loop feedback system, it can destabilize or otherwise corrupt the running state of the system and cause damage nevertheless.

In this paper, we do not assume trustworthy clock synchronization between the controller and the power plant. It is because traditional clock synchronization protocols based on round trip time (RTT) measurements (e.g., NTP) are susceptible to attacks [17] and later novel solutions (e.g., [4]) may not be available in a particular deployment. Hence, we adopt a defense-in-depth paradigm to detect and characterize a TDA, independent of any orthogonal efforts to synchronize clocks securely in the CPS.

IV. Proposed Solution

We propose a novel solution to detect and characterize a TDA in a CPS, focusing particularly on its practical implications. We first give a mathematical formulation of the problem. We then propose a DL model to provide continual periodic assessments of the status of any ongoing TDA. Importantly, the DL uses a specialised training protocol targeted to our application domain.

A. Problem Formulation

In general, the TDA causes actuators to execute outdated commands (Eq. (2)), which affects the state of the system adversely and these effects are reflected in the run-time collected sensor data, which are provided as input to the proposed learning model. The run-time inputs to the learning are given as a continuous stream of online (vector) data, analyzed as they become available, rather than postmortem "whole" data traces analyzed offline for, say, after-the-fact forensics. Since we are working with a discrete-time system, the output of the proposed model is an integer estimate of the *TDA value*, which is the amount of the malicious delay in seconds. By processing inputs as soon as they are available incrementally in ongoing data streams, we allow real-time responses (e.g., mitigation) to any detected attacks. The functionality of the model is specified as

$$\tau_e=M(Z^x[t]),\ t=t_1,t_2,\ldots,t_e, \tag{3}$$

where τ_e is the estimated malicious delay introduced by the TDA, $M(\cdot)$ is the learning model, $Z^x[t]$ is the data of sensor x at time t, T denotes the length of a trace of the readings, and $t_e<T$ is the last time slot used by the model to estimate the delay τ_e. In the problem formulation (3), as a plugin solution, the input to our DL model is the sensor (vector) data stream $Z^x[t]$. The DL model $M(\cdot)$ keeps processing the received input trace to extract features that are useful for the delay detection and characterization.

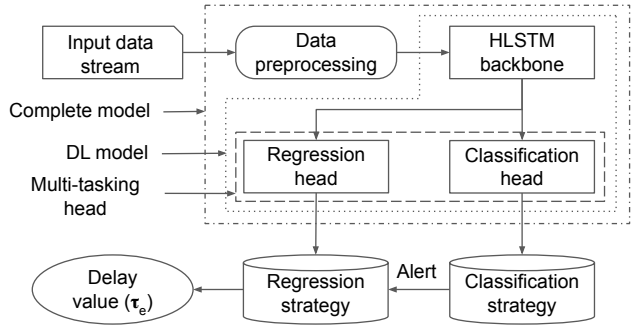


Fig. 3: Structural overview of the proposed solution.

In this work, we consider three types of sensor readings in the PPCS, namely the temperature TM, the pressure PR, and the generated electricity GE. We consider a different set of sensor readings in the AGC, namely the tie-line sensor values $Tieline_1$ to $Tieline_n$, where n is the number of tie-lines in the system. We denote the sensor readings as $Z^x[t], t=\{1,2,\ldots,T\},\ x\in\{TM,PR,GE\}$ for PPCS and $x\in\{Tieline_1,..,Tieline_n\}$ for AGC.

Assume that the system is under a TDA of value τ introduced at $t=\bar{t}$. Then the accuracy of the DL model is measured by the difference between the estimated delay τ_e and the ground truth delay τ. The *reaction latency* is measured by the length of the trace required to obtain the model's output after the TDA starts, i.e., $t_e-\bar{t}$. In the following, we introduce the details of our proposed model $M(\cdot)$.

B. Model Design

We illustrate the structure of our solution in Fig. 3. Whereas the input trace provided to our model consists of measurements from various sensors created during simulations, in our solution, we first preprocess and clean the input data. The preprocessed data trace is then passed through a hierarchical LSTM (HLSTM) network, which provides temporal features periodically for further processing by a multi-tasking head. The multi-tasking head is designed as two separate modules, namely the classification module and the regression module, which perform respectively the delay attack detection and characterization. Once the TDA is confirmed by the classification module, an alert is sent to further characterize the TDA by activating the regression module to apply regression analysis on the sensor data. The details are given in the following.

Data pre-processing: The sensor measurements exhibit a high variance in magnitude over time. Since we want to train a continually running time-invariant model which can provide delay attack detection and characterization irrespective of the position of attack, the input data trace needs to be normalized over time to fit the model's input. Because of a TDA's negative impacts on the control system, the sensor measurements can sometimes deviate heavily from their expected behaviour. We use robust scaling, a numerical scaling technique whose

scaling statistics are based on percentiles and can thus deal with the existence of a few very large marginal outliers.

HLSTM backbone: For timely TDA detection and characterization, our model needs to satisfy the following practical requirements: (i) Able to handle long discrete-time data sequences from sensors; (ii) Suitable for continuous monitoring to give actionable information as soon as it is available.

Among DL learning models, recurrent neural networks (RNNs) have a distinguishing feature of internal state that remembers information from the past; they are thus preferred for sequence processing [42]. An LSTM network [43] is a particular RNN with "gates" to decide what information from the past states should be used to achieve the learning objectives. For dynamical systems, LSTM often outperforms conventional RNNs due to its capability to learn long-term dependencies among data [27]. We therefore adopt LSTM as a basis for the proposed solution.

However, LSTMs face the challenge of vanishing and exploding gradients [44], when working on long data sequences that may keep on growing, such as the sensor data streams in our problem. To solve the challenge, we propose a hierarchical LSTM model as illustrated in Fig. 4(a). Our model contains two LSTM levels with hierarchical connections to allow them to divide the input data sequence into shorter sub-sequences; the two levels together sharpen understanding about the complete sequence. The lower LSTM works strictly as a local feature extractor and provides input to the upper LSTM. The upper LSTM works on the sequence of local features provided by the lower LSTM to extract temporal features for the complete sequence. Essentially, the lower LSTM works on a fixed sub-sequence length and is reset by removing the links periodically, whenever the output is provided to the upper LSTM. The upper LSTM runs in a sequence-to-sequence fashion, to supply periodic features to dense layers. We do not use back connections (e.g., as in BiLSTM) to keep the LSTM compatible with online processing.

For the lower LSTM cell, denote the input, output, input cell state, and output cell state respectively for the ith timestep as $\{W_i^L, V_i^L, C_{in_i}^L, C_{out_i}^L\}$; for the upper LSTM cell, denote the input, output, input cell state, and output cell state for the jth timestep as $\{W_j^U, V_j^U, C_{in_j}^U, C_{out_j}^U\}$. Based on the functioning of a conventional LSTM cell, generally we have $\{V, C_{out}\} = \sigma(W, C_{in})$, where σ is the LSTM network. Let the initial LSTM cell state be C_0^L for the lower LSTM and the frequency of passing outputs to the upper LSTM be $1/\omega$, where ω is an integer (in time units). The layer functions are specified as:

$$\{V_i^L, C_{out_i}^L\} = \begin{cases} \sigma^L(W_i^L, C_0^L), & (i-1)\%\omega = 0 \\ \sigma^L(W_i^L, C_{out_{i-1}}^L), & \text{otherwise}, \end{cases} \quad (4)$$

$$\{V_j^U, C_{out_j}^U\} = \sigma^U(W_j^U, C_{out_{j-1}}^U), W_j^U = V_{\omega * j}^L, \quad (5)$$

where $\sigma^L(\cdot)$ and $\sigma^U(\cdot)$ are respectively the lower and upper LSTM layers. The lower LSTM is reset every ω time slots and the upper LSTM obtains outputs from the lower LSTM to generate the required features every ω time slots, as shown in Eq. (4) and (5). Moreover, for an input sequence of length l, the lower LSTM and the upper LSTM work on sub-sequences of length ω and l/ω respectively. To prevent vanishing gradients during training, we need to reduce the input sequence length of the upper LSTM; this requirement necessitates a lower bound for the value of ω. On the other hand, the frequency of outputs, $1/\omega$, determines the bottleneck reaction latency of our model and provides an upper bound for ω. In general, there exists a trade-off between higher and lower values of ω, and a suitable value for a given setting can be found by a search procedure. The impact of the choice of ω will be discussed in the experiments in Section V-C1.

Multi-Tasking head: To achieve both accurate TDA detection and characterization, we introduce a multi-tasking head with separate classification and regression modules, as shown in Fig. 4(b). The classification module first decides whether there is an attack or not. If the answer is affirmative, the regression module will output its result as an estimate of the malicious delay (i.e., the TDA value). This design is in contrast to the use of a regression module only, which estimates the malicious delay unconditionally – in the case of no attack, $\tau_e = 0$. Using a separate classification module can improve accuracy, since a continuous regression process will produce outputs that are be prone to continuous small errors.

We hypothesize that both modules on an abstract level focus on recognising a TDA from the input signal and only differ in the level of granularity that they offer. Thus a common set of temporal features provided by the HLSTM backbone can be used by both the modules to save computation. Detailed evaluations are presented in Section V-C3. Specifically, the multi-tasking head is defined as

$$V_C = \phi_C(D_C(V^U)),\ V_R = \phi_R(D_R(V^U)),$$

where V_C and V_R are respectively outputs of the classification and regression modules; $D_C(\cdot)$ and $D_R(\cdot)$ denote fully connected layers in the classification and regression, respectively; $\phi_C(\cdot)$ and $\phi_R(\cdot)$ are the activation functions in the classification and regression, respectively. Due to the discrete-time nature of the attack model, we round off the output V_R to the nearest integer. The outputs V_C and V_R respectively denote final answers for the TDA detection and characterization. In the following, we will first introduce the training process in the DL model. Then, we will give the details of interpreting the model outputs for final decisions.

C. Training Process

There are two main challenges in training our multi-tasking HLSTM: (i) The DL model must output as early as possible, instead of doing so only after analyzing a long forensic data trace; (ii) The dataset compositions required by the classification and regression conflict due to different training targets. In the following, we present our training process that overcomes these challenges.

1) Sliding window based input data processing: To allow the DL model to process an input sensor data stream incrementally, as soon as the current prefix becomes available, thus giving results as early as possible, under variable TDA launch time in the training dataset, we propose creating small sliding windows of a fixed trace length $\lambda(\lambda < T)$ for training,

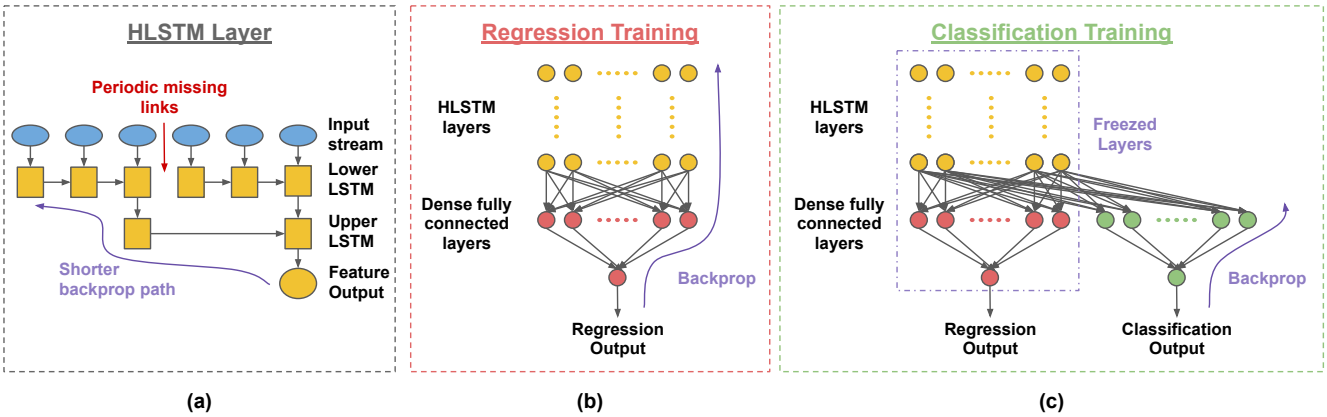


Fig. 4: (a) Proposed hierarchical LSTM architecture. (b) & (c) The overall learning model and the training procedure.

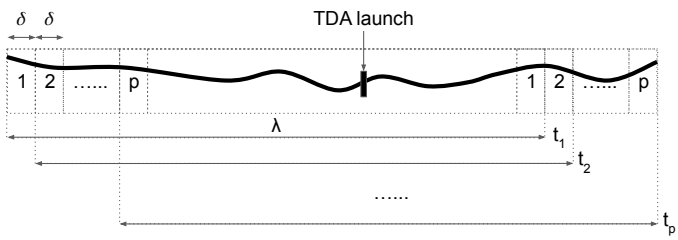


Fig. 5: Illustration of the sliding window approach of extracting training sequences from a simulation system run.

as shown in Fig. 5. This method also increases the speed of dataset generation, since we can create multiple such traces from a single run of the system (in our case, a simulation of the PPCS). In the sliding window approach, the window can move in either direction with a stride of $\delta(\delta > 0)$ to extract traces from a log of the system run. In generating the training data, we ensure that the window covers the TDA launch time point and contains a substantial amount of data both before and after this point, so that appropriate features, spanning both cases of pre- and post-attack, can be learned in the DL model.

2) Asynchronous model training: The regression head in our model needs to give a fine-grained characterization of the TDA value and should be trained on a dataset with a balanced number of examples for a wide distribution of the TDA values (including $\tau = 0$). On the other hand, the classification head differentiates between the presence or absence of TDA; it should be trained on a dataset that has a balanced number of examples for both cases. Thus, the dataset compositions required by the two heads are different, which raises a challenge for training an overall model for them together.

To solve the challenge, we propose an asynchronous training protocol as shown in Fig. 4(b). We first create the model with only the regression head and train it end-to-end using a dataset with a uniform distribution of malicious delay values. Next, we include the classification head and freeze the weights of both the regression head and the shared HLSTM backbone. Then, we train the classification head, for which the input is the features created by the HLSTM (since the error will not backpropagate beyond them) using a dataset with a balanced mix of attack and no-attack samples. Finally, all the trained weights are put together to give the overall model.

D. Strategies on Interpreting Model Output

Our model provides periodic outputs every few times slots. Each output tells whether a TDA is indicated or not and, if so, a characterization of the TDA. To control between false positives and false negatives, as well as the accuracy of the estimated delay, we devise different strategies for interpreting multiple outputs from different time slots to arrive at the final learning result. This interpretability of our solution is independent of the original model training and does not require any form of retraining for the same. This final result, denoted as τ_e, is defined as

$$\tau_e = \begin{cases} S_R(V_R), & S_C(V_C) = \text{True}, \\ 0, & \text{otherwise}, \end{cases} \tag{6}$$

where $S_R(\cdot)$ and $S_C(\cdot)$ denote respectively the regression and classification functions. Note that if the classification result is negative (i.e., no attack), the final result τ_e is zero. Since no standard form of interpretation strategies exist in the literature that fuse multiple outputs from different time slots, in the following, we present key details of our strategies.

1) Classification strategies $S_C(\cdot)$: In the TDA detection, the classification module gives a binary result (i.e., presence or absence of attack) every ω time slots (e.g., $\omega = 15$). Rather than alerting the system operator immediately on any positive classification, a more prudent approach is to wait until n consecutive positive results before raising the alert. This strategy increases the confidence threshold required for action; i.e., it increases the model's *specificity*. The increased specificity comes at a price of a higher chance of missing a real TDA, i.e., a reduced model *sensitivity*. Additionally, since our model produces an output every ω time slots, waiting for $n > 1$ consecutive alerts necessarily increases the reaction latency. It is interesting to investigate the trade-off among these three performance aspects based on specific system requirements.

2) Regression strategies $S_R(\cdot)$: The regression head is responsible for characterizing the TDA value after the classification head declares an attack. As the length of the input data trace prefix received increases, more data becomes available

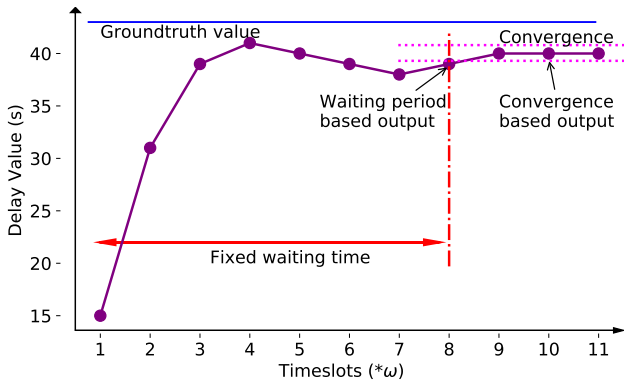


Fig. 6: Example of the two regression strategies.

for the regression and therefore, intuitively, the regression's accuracy is expected to increase, i.e., consecutive regression module outputs will converge close to the ground truth value. Moreover, since a TDA's impacts likely intensify over time, fusing multiple outputs of the regression module to obtain an overall characterization result can provide better accuracy than using any single output. Based on these observations, we propose two strategies for interpreting multiple consecutive regression module outputs.

Waiting-time based strategy: A straightforward way to increase accuracy is to wait for some predefined number of time slots before giving the final result. The longer we wait, likely the more accurate the prediction will be. The drawback of a longer wait is, however, a longer reaction time, i.e., potential efforts to counteract a TDA will be delayed.

Convergence-based strategy: As mentioned, the regression outputs will likely converge to some value near the ground truth. Hence, instead of fixing the waiting time, we can monitor the convergence process and give the final characterization result when a convergence threshold is reached.

An example of regression outputs using the two proposed strategies is shown in Fig. 6. We can observe that consecutive regression module outputs converge towards the ground truth value. In general, waiting for more time slots leads to more accurate final results. A detailed performance comparison between the two strategies will be discussed in Section V-C2.

V. Evaluation

We now evaluate the performance of the proposed DL model for the PPCS and AGC in power grids. We first describe the experiment settings, model settings, and evaluation metrics. We will then present experimental results to illustrate the performance of our approach under the diverse settings and compare the performance of our approach with that of several major traditional ML techniques.

A. Experiment and Model Settings

We generate the dataset in the PPCS as shown in Fig. 1 using Thermopower [38]. The sensor measurements are subject to random additive zero-mean Gaussian noises. The simulation starts at $t = 200s$ and terminates at $t = 1500s$ or earlier if the system crashes due to the attack. The data collection frequency

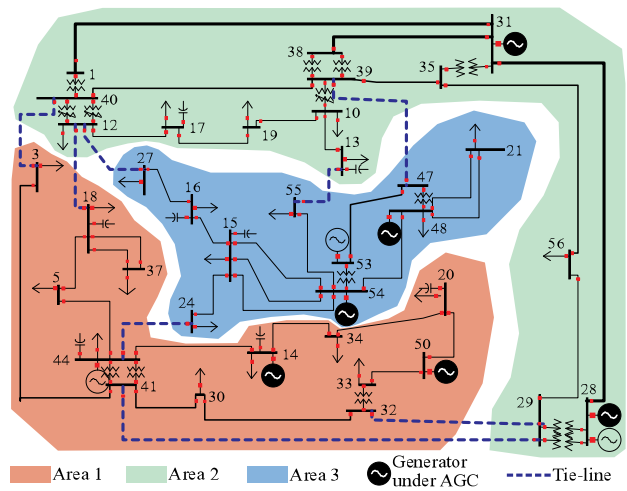


Fig. 7: A three-area 37-bus system with AGC controller [45].

is 0.5 (i.e. 1 time slot is equivalent to 2 seconds). The dataset contains 20,000 traces of three sensor measurements, TM, PR and GE, where 10,000 traces consist of a roughly even balance between the cases of attacks and no attacks for the classification training and the remaining 10,000 traces contain TDA of different delay values for the regression training. For each attack, the delay τ and the TDA launch time $\bar{t}$ are integers generated uniformly at random between 0s and 50s as well as 800s and 1200s, respectively. For each dataset, we use a split of 60% for training, 10% for validation, and 30% for testing. For data augmentation during training, we use the sliding window method as detailed in Section IV-C1. We use the sliding window of size $\lambda = 600$ and stride $\delta = \omega$ (ω is a searchable hyperparameter) for creating shifted copies from the training dataset. The final results are performed on the combined testing dataset for both classification and regression.

The evaluations of AGC are conducted in a three-area 37-bus power system.[2] The experiments use PowerWorld [41], an industry strength power grid simulator. As shown in Fig. 7, the complete system is divided into three areas, which are connected with each other through eight different tie-lines (the dashed lines in the figure). The eight tie-line flow measurements are representative of each area's power export, and they are used as our model input.

The simulation for AGC starts at $t = 20s$ and terminates at $t = 320s$, with a data collection frequency of 0.25 (i.e., 1 time slot is equivalent to 4 seconds). The AGC dataset contains a smaller trace length compared with the PPCS dataset, as it is more sensitive to the delay attack and gets unstable very quickly when subject to these attacks. The dataset contains 6,000 traces of eight sensor measurements, $Tieline_1$ to $Tieline_8$, divided into 3,000 traces each for classification and regression training respectively, similar to the PPCS case. For each attack, the delay τ and the TDA launch time $\bar{t}$ are integers generated uniformly at random

[2]We use the 37-bus system as a representative AGC throughout this paper. It is a test system [46]. Its scale corresponds to a small to mid-scale grid in real life. According to our rough count based on a grid topology database (http://bit.ly/2vRH5Nd), a major fraction of 130 national grids consist of fewer than 37 buses.

between 0s and 10s and between 100s and 220s in two separate settings. Again, a smaller range of delay attack values is used for the AGC dataset, for the same reason as before. This also dictates the total size of the dataset used, as a smaller range of possible delay values reduces the size of the dataset required. We use a sliding window of size $\lambda = 100$ and stride $\delta = \omega$ (ω is a searchable hyperparameter) for creating shifted copies from the training dataset. The dataset split as well as sensor measurement noises are the same as those for PPCS.

The DL model contains two LSTM cells stacked together at the lower level, each with 256 hidden units. The upper level is relatively complex since it focuses on extracting temporal features from the complete sequence and contains two LSTM cells stacked together, each with 512 hidden units. The output of the upper LSTM is passed through two different dense fully connected layers (for regression and classification, respectively), each containing 512 hidden units. All the layers have a ReLU activation function at the output, except for the classification branch output, which has a Sigmoid activation function. The classification and regression module are respectively trained using binary cross-entropy loss and mean squared error loss. We use an Adam optimizer for the training with a learning rate of 0.001. We also add a dropout of 0.1 between the hidden layers to prevent over-fitting.

B. Evaluation Metrics

We evaluate the model performance for both TDA detection and characterization. The performance of the classification module can be analyzed using an error matrix in which true positives (TP) and true negatives (TN) represent correct detection of the TDA, false negatives (FN) are TDAs that are missed by the learning, false positives (FP) are wrongly detected TDAs that are in fact not present. The model's sensitivity and specificity respectively correspond to the avoidance of FNs and FPs. We define the classification accuracy ACC as:

$$ACC = \frac{TP + TN}{TP + TN + FP + FN}.$$

For the regression module's performance, its error is quantified in terms of mean absolute error (MAE) and root mean square error (RMSE), which are defined respectively as:

$$MAE = \frac{1}{m}\sum_{i=1}^{m} |\tau^i - \tau_e^i|,\ RMSE = \sqrt{\frac{1}{m}\sum_{i=1}^{m}(\tau^i - \tau_e^i)^2},$$

where τ^i and τ_e^i represent respectively the groundtruth and estimated malicious delay values for the ith test dataset trace and m is the size of the test dataset. The average time required by the model to characterize the TDA is given as the reaction latency; it is the average time (in seconds) from the launch of the attack to the availability of the learning result, defined as:

$$T_{avg} = \frac{1}{f} \cdot \frac{1}{m}\sum_{i=1}^{m}(t_e^i - \bar{t}^i),$$

where $\bar{t}^i$ denotes the attack launch time and t_e^i denotes the time slot of the result for the ith test case, and f denotes the data collection frequency (i.e., 0.5 for PPCS and 0.25 for AGC).

C. Performance Evaluations

1) Model parameter settings: There are two major hyperparameters involved in our architecture design: the depth of the model and the frequency of inputs to the upper LSTM (i.e., $1/\omega$). For the model depth, we can increase the layers of either the lower or upper LSTM. We illustrate the results in Table I and II to show the impact of different depth values. As expected, the performance of the model first increases with an increase in depth; however, it can saturate and start decreasing after an optimal value of depth. This can be explained by the commonly observed phenomenon of vanishing gradient and overfitting. For PPCS, we can notice from Table I that the depth of two in the lower LSTM achieves a good balance between performance and the number of parameters, whereas for the upper LSTM, the depth of three achieves better performance at the cost of a much larger number of model parameters (i.e., more than 7 million). Moreover, the upper LSTM's computational time complexity is much higher than that of the lower LSTM due to the feature input length and the number of hidden units. Similarly, for the AGC, note from Table II that the depth of two in both the lower LSTM and the upper LSTM achieves the best performance.

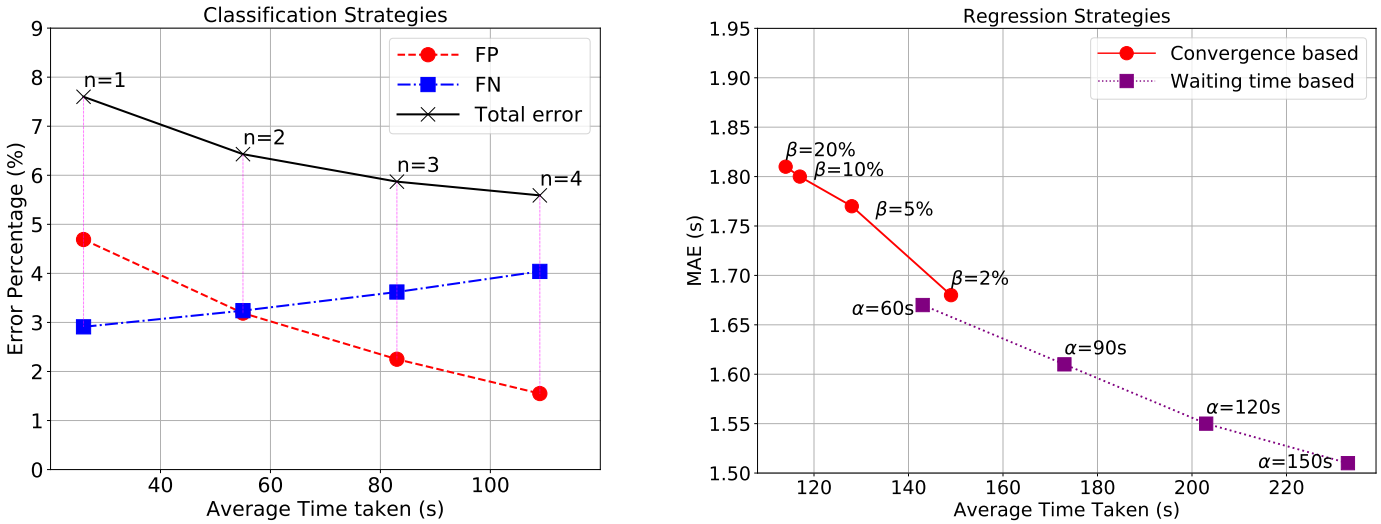
TABLE I: Performance comparison across varying model depth values for PPCS.

Depth		Regression (s)		Classification (%)			No. of
Lower	Upper	MAE	RMSE	ACC	FP	FN	Param.(m)
1	2	2.11	5.83	90.56	7.4	2.1	4.47
2	1	2.44	5.09	88.31	9.8	1.9	2.89
2	2	2.03	5.48	92.39	**4.7**	2.9	4.99
2	3	**1.81**	**4.27**	**92.63**	4.8	**2.6**	7.09
3	2	2.65	5.97	88.59	8.8	2.6	5.52

TABLE II: Performance comparison across varying model depth values for AGC.

Depth		Regression (s)		Classification (%)		No. of
Lower	Upper	MAE	RMSE	ACC	FN	Param.(m)
1	2	0.59	1.21	97.88	2.12	4.47
2	1	0.84	2.69	**98.89**	**1.11**	2.89
2	2	**0.49**	**0.98**	98.49	1.51	4.99
2	3	2.37	3.66	63.01	36.69	7.09
3	2	0.71	1.65	93.03	6.97	5.52

An interesting trend that can be noticed for the AGC, which is not present in the PPCS results, is the lack of False Positive (FP) examples. After experimenting with various variations of our model, we found that our learning is free of any FP (i.e., false alarms) in the AGC dataset. This can be explained by the self-correcting nature of the AGC. For some smaller TDA values, the divergence caused by the attack is mitigated internally by standard frequency control in power grids and thus simulations with smaller TDA values can behave similarly to simulations without any TDA. Since our model learns to rely on signal divergence for detecting the TDA, our classification never causes false alarms because no divergence is noticed in those cases. Instead, all the detection error manifests as False Negatives (or missed attacks), where the divergence is sometimes automatically corrected by the system before it ever gets detected by our model. Since the PPCS does not have a



(a) (b)

Fig. 8: Performance of different output interpretation strategies: (a) Classification performance under different n. (b) Regression performance under different strategies.

similar internal method of correction to smaller delay values, this result sets the two systems apart.

For the frequency of providing inputs to the upper LSTM in the PPCS, we increase ω from 10 to 25, in steps of 5, to observe its impact. The results are shown in Table III. Similarly, for the AGC, the value of ω is increased from 3 to 6, in steps of 1, and the final results are provided in Table IV. From Tables III and IV, note that a higher frequency of passing outputs to the upper LSTM (i.e., a smaller ω) gives faster final results. For the accuracy comparison, notice the trade-off discussed in Section IV-B due to which ω values at both extremes experience performance degradation. We see that $\omega = 15$ for the PPCS and $\omega = 4$ for the AGC achieves the best accuracy, in a reasonable amount of time. We use these ω values for further evaluations.

TABLE III: Performance across different values of ω for PPCS.

Value of ω	Regression (s)			Classification only (%)		
	MAE	RMSE	T_{avg}	ACC	FP	FN
10	2.61	6.07	**117**	90.54	6.1	3.3
15	**2.03**	**5.48**	128	**92.39**	**4.7**	**2.9**
20	2.14	6.21	145	91.12	5.4	3.5
25	3.14	6.91	178	86.45	7.4	5.3

TABLE IV: Performance across different values of ω for AGC.

Value of ω	Regression (s)			Classification only (%)	
	MAE	RMSE	T_{avg}	ACC	FN
3	0.82	1.52	**47**	97.41	2.51
4	**0.49**	**0.98**	49	**98.49**	**1.51**
5	0.70	1.50	61	98.20	1.80
6	1.13	1.98	67	88.20	11.80

2) Output interpretation analysis: We now compare the performance between various strategies of interpreting the model output. While we only use the PPCS results here for analysis, our solution shows a similar trend for the case of AGC too. We first consider different strategies of interpreting outputs of the classification module. We draw Fig. 8(a) for different values of n, where positive detection is concluded finally from n consecutive positive classification module results. From Fig. 8(a), we see that a larger n uses more time to conclude detection, but the accuracy is improved as more information is available in longer traces. Moreover, the specificity of the model increases whereas its sensitivity decreases when n is increased. The reason is that with a larger n, we are more conservative in confirming an attack, which helps to avoid false positives. In this set of experiments, $n = 2$ or $n = 3$ can achieve a good balance between the error rate and the reaction latency.

Next, we evaluate the performance of different regression strategies under different parameter settings. The results are shown in Fig. 8(b). To focus on the regression strategy, we fix the classification strategy to $n = 3$ for all the results in Fig. 8(b). In the waiting-time based strategy, α corresponds to a fixed waiting time for the regression after the TDA is detected. In the convergence-based strategy, β corresponds to a percentage convergence criterion i.e. the difference in the results of the two consecutive regression outputs is within the percentage specified. From Fig. 8(b), we see that when we have a longer waiting time α, we achieve a smaller error but require a longer time for obtaining the result. For the convergence criterion, a larger β can cause a higher error rate but a shorter characterization time. It is because a looser criterion is easier to satisfy (i.e., takes less time) but the converged result may be farther from the optimal. The performance of the convergence-based strategy further emphasizes that outputs provided by our DL model converge towards the ground truth over time. Moreover, the convergence-based strategy achieves a good balance between the error rate and reaction latency, and it also allows variable reaction latency for each individual test case.

3) Comparison with traditional approaches: We now compare the performance of different learning-based approaches.

We start with kNN and RF as conventional ML-based solutions. Since these algorithms do not give continuous outputs as required by our settings, their evaluations are based on their final outputs obtained after processing each entire input trace. For both kNN and RF, we use separately trained models for regression and classification. We also compare the proposed HLSTM backbone by replacing it with a vanilla LSTM and another traditional LSTM model, which we call accumulate LSTM (as used in [29]). Accumulate LSTM simply concatenates input traces, one for every ω time slot, and then feeds the result as a single input feature vector to the LSTM cell. In other words, accumulate LSTM replaces the lower LSTM in our model with a simple concatenation of the input signal values. Both the vanilla LSTM and the accumulate LSTM have the same configuration as our upper LSTM and are trained till convergence. Moreover, we also analyse the performance of the multi-tasking head. The results are shown in Tables V and VI for the PPCS and AGC respectively, where metrics for both the TDA detection and characterization are shown.

TABLE V: Performance comparison with traditional approaches for PPCS.

Approach	Regression (s)			Classification (%)		
	MAE	RMSE	T_{avg}	ACC	FP	FN
kNN	6.23	9.48	300	72.6	11.8	15.6
RF	6.44	10.32	300	80.82	5.2	13.9
Lou et al. [29]	3.73	6.84	300	–	–	–
Van. LSTM+Multi-Task	4.17	8.76	136	85.21	9.7	5.1
Acc. LSTM+Multi-Task	2.76	6.03	168	89.76	7.1	3.3
1 HLSTM+Regression	3.51	7.46	148	–	–	–
2 HLSTMs+Multi-Task	**2.02**	5.53	**124**	**93.03**	**3.7**	3.2
1 HLSTM+Multi-Task	2.03	**5.48**	128	92.39	4.7	**2.9**

TABLE VI: Performance comparison with traditional approaches for AGC.

Approach	Regression (s)			Classification (%)	
	MAE	RMSE	T_{avg}	ACC	FN
kNN	3.27	5.02	200	71.29	28.71
RF	3.41	4.10	200	74.57	25.43
Lou et al. [29]	1.34	2.17	200	–	–
Van. LSTM+Multi-Task	1.74	3.84	63	81.07	18.93
Acc. LSTM+Multi-Task	0.77	1.42	65	97.71	2.29
1 HLSTM+Regression	1.07	1.84	81	–	–
2 HLSTMs+Multi-Task	**0.47**	**0.91**	57	**99.09**	**0.91**
1 HLSTM+Multi-Task	0.49	0.98	**49**	98.49	1.51

We can see that our model works much better than kNN, RF, and the DL model in [29], in the regression and classification performance as well as the reaction latency. Moreover, as presented in [12], [13], it takes around 200s in the PPCS and around 100s after an attack launches in the AGC for the TDA to become harmful to the system. Hence, the average time needed to obtain the result, $T_{avg} = 128s$ in the PPCS or $T_{avg} = 49s$ in the AGC, is beneficial for the subject systems. This result is obtained when $n = 1$ and $\beta = 2\%$. We can decrease T_{avg} further by increasing the regression error a bit as shown in Fig. 8(b) (e.g., for the PPCS with $\beta = 20\%$, we obtain $T_{avg} = 65s$ and $MAE = 2.56$). Additionally, comparison with different traditional backbones show the superiority of our HLSTM architecture and comparison with the regression only model shows the importance of our multi-tasking head.

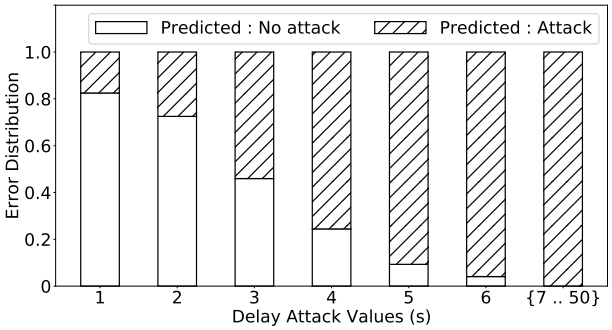


Fig. 9: TDA detection error distribution for PPCS.

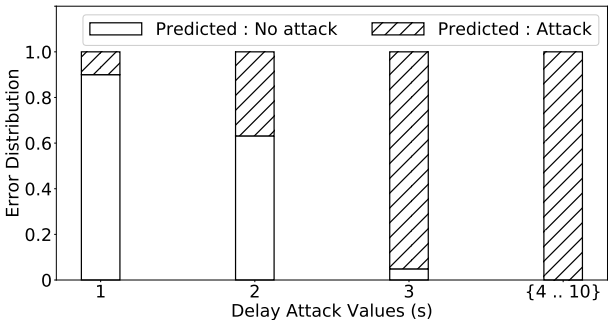


Fig. 10: TDA detection error distribution for AGC.

Notice also that we can achieve similar performance with less computational resources by using only one common HLSTM backbone instead of two separate ones. This supports our initial hypothesis that the temporal features provided by a common backbone can be used for both detection and characterization of the attack.

4) Sensitivity analysis: Our classification head achieves over 92% accuracy in PPCS. The error is divided into 4.7% FN and 2.9% FP. To better understand how well our model can perform in TDA detection, we further analyze the FN distribution for different delay values of τ. Fig. 9 shows the classification distribution under different delay values from 1s to 50s. We see from Fig. 9 that our model can detect all the TDAs when $\tau > 6s$, which is expected as higher delay values can cause visible divergence in the signal. When τ is between 4s to 6s, around 20% to 5% TDAs are missed. When $\tau < 4s$, the missed detection rate increases sharply, reaching as high as 80% for $\tau = 1$.

The sharp increase in FN for lower values of τ can be attributed to the fact that these small delay values cause no significant effects on the system. Consequently, the signal remains unaffected, and since we use a data-driven solution, our model has no means of identifying the existence of an attack. However, missing weak attacks that have no significant impacts is acceptable. By the conclusion drawn in [12], [13], the PPCS can tolerate up to $\tau = 12s$ of malicious delay with no harm to the normal operation of the system. Also the impact of FP is limited regarding system safety or stability, since the triggered mitigation in those cases will be of small strengths, as detailed in prior work [12], [13]. A sensitivity analysis of the AGC problem also renders similar results, as shown in Fig. 10. However, due to the lower tolerance of AGC against

delay attacks [12], [13], the distribution of FN is also across smaller values of τ. More than 80% of FN is shared between $\tau = 1$ and $\tau = 2$, and all delays for $\tau > 3$ are detected by our solution for the AGC.

VI. Conclusion

In this paper, we proposed a learning based solution that analyzes relevant sensor outputs in a closed-loop CPS to simultaneously detect and characterize time delay attacks. Our solution focuses on practicality for real systems and is configurable to different objectives based on users' specific requirements. Our model outperforms conventional ML algorithms like kNNs, RF, and other DL models. It achieves 92% and 98% accuracy in detecting TDAs in the cases of PPCS and AGC, respectively. It also achieves an MAE of almost two seconds and half a second in the PPCS and AGC, respectively, in terms of delay value characterization.

Since we have designed our model to continuously monitor the subject system in real-time settings, it is interesting for future research to investigate deployment issues on embedded devices and create a hardware plugin for protecting a CPS against the TDA. For this purpose, one will likely need to reduce the model size and perform optimizations specific to the hardware platform in question. Also, while our work applies to TDAs against a single control signal, in real systems multiple control signals might be subject to the TDA at the same time. The problem becomes more complicated as malicious delays in multiple control signals can have a compounding effect. It will be interesting to study how our proposed method can be adapted to these further scenarios.

Acknowledgement

This research was supported in part by the National Research Foundation, Prime Minister's Office, Singapore under its Campus for Research Excellence and Technological Enterprise (CREATE) programme, and in part by the National Research Foundation, Singapore, and the Energy Market Authority, under its Energy Programme (EP Award no. NRF2017EWT-EP003-061). This publication was made possible in part by NPRP grant # [NPRP10-0208-170408] from the Qatar National Research Fund (a member of Qatar Foundation). The findings herein reflect the work, and are solely the responsibility of the author[s].

References

[1] H. Farhangi, "The path of the smart grid," *IEEE power and energy magazine*, vol. 8, no. 1, pp. 18–28, 2009.

[2] H. He and J. Yan, "Cyber-physical attacks and defences in the smart grid: a survey," *IET Cyber-Physical Systems: Theory & Applications*, vol. 1, no. 1, pp. 13–27, 2016.

[3] Y. Zhang and V. Paxson, "Detecting stepping stones." *Proc. of USENIX Security Symposium*, 2000.

[4] S. Viswanathan, R. Tan, and D. K. Yau, "Exploiting power grid for accurate and secure clock synchronization in industrial IoT," *IEEE RTSS*, pp. 146–156, 2016.

[5] S. Karnouskos, "Stuxnet worm impact on industrial cyber-physical system security," *IEEE IECON*, pp. 4490–4494, 2011.

[6] A. A. Cárdenas, S. Amin, and S. Sastry, "Research challenges for the security of control systems." *HotSec*, 2008.

[7] S. Liu, X. P. Liu, and A. El Saddik, "Denial-of-service (DoS) attacks on load frequency control in smart grids," *IEEE ISGT*, 2013.

[8] Y. Liu, P. Ning, and M. K. Reiter, "False data injection attacks against state estimation in electric power grids," *ACM CCS*, pp. 21–32, 2009.

[9] S. Amin, A. A. Cárdenas, and S. S. Sastry, "Safe and secure networked control systems under denial-of-service attacks," *Hybrid Systems: Computation and Control*, pp. 31–45, 2009.

[10] B. Q. M. AL-Musawi, "Mitigating dos/ddos attacks using iptables," *IJET*, vol. 12, no. 3, pp. 101–111, 2012.

[11] A. Sargolzaei, K. K. Yen, and M. Abdelghani, "Time-delay switch attack on load frequency control in smart grid," *Advances in Communication Technology*, vol. 5, pp. 55–64, 2013.

[12] X. Lou, C. Tran, R. Tan, D. K. Y. Yau, Z. T. Kalbarczyk, A. K. Banerjee, and P. Ganesh, "Assessing and mitigating impact of time delay attack: case study for power grid controls," *IEEE JSAC*, vol. 38, no. 1, pp. 141–155, 2020.

[13] X. Lou, C. Tran, R. Tan, D. K. Yau, and Z. T. Kalbarczyk, "Assessing and mitigating impact of time delay attack: a case study for power grid frequency control," in *Proceedings of the 10th ACM/IEEE International Conference on Cyber-Physical Systems*, 2019, pp. 207–216.

[14] A. Sargolzaei, K. Yen, and M. N. Abdelghani, "Delayed inputs attack on load frequency control in smart grid," *IEEE ISGT*, 2014.

[15] W. Michiels and S.-I. Niculescu, *Stability, control, and computation for time-delay systems: an eigenvalue-based approach*. SIAM, 2014.

[16] H. Ali and D. Dasgupta, "Effects of time delays in the electric power grid," in *International Conference on Critical Infrastructure Protection*. Springer, 2012, pp. 139–154.

[17] R. Annessi, J. Fabini, F. Iglesias, and T. Zseby, "Encryption is futile: Delay attacks on high-precision clock synchronization," *arXiv:1811.08569*.

[18] N. J. Higham, *Accuracy and stability of numerical algorithms*. SIAM, 2002, vol. 80.

[19] R. J. De Boer, "Which of our modeling predictions are robust?" *PLoS computational biology*, vol. 8, e1002593, 2012.

[20] A. Sargolzaei, K. K. Yen, M. N. Abdelghani, S. Sargolzaei, and B. Carbunar, "Resilient design of networked control systems under time delay switch attacks, application in smart grid," *IEEE Access*, vol. 5, pp. 15 901–15 912, 2017.

[21] M. Shafique and N. Iqbal, "Load frequency resilient control of power system against delayed input cyber attack," *IEEE RAEE*, 2015.

[22] F. Pasqualetti, F. Dörfler, and F. Bullo, "Cyber-physical attacks in power networks: Models, fundamental limitations and monitor design," *IEEE CDC-ECC*, pp. 2195–2201, 2011.

[23] A. Sargolzaei, K. K. Yen, and M. N. Abdelghani, "Preventing time-delay switch attack on load frequency control in distributed power systems," *IEEE Transactions on Smart Grid*, vol. 7, no. 2, pp. 1176–1185, 2015.

[24] S. Ben Atia, A. Messaoud, and R. Ben Abdennour, "An online identification algorithm of unknown time-varying delay and internal multimodel control for discrete non-linear systems," *Mathematical and Computer Modelling of Dynamical Systems*, vol. 24, no. 1, pp. 26–43, 2018.

[25] Y. Deng, V. Léchappé, S. Rouquet, E. Moulay, and F. Plestan, "Super-twisting algorithm based time-varying delay estimation with external signal," *IEEE Transactions on Industrial Electronics*, 2019.

[26] E. Korkmaz, M. Davis, A. Dolgikh, and V. Skormin, "Detection and mitigation of time delay injection attacks on industrial control systems with plcs," in *International Conference on Mathematical Methods, Models, and Architectures for Computer Network Security*. Springer, 2017, pp. 62–74.

[27] R. Pascanu, T. Mikolov, and Y. Bengio, "Understanding the exploding gradient problem," *CoRR, abs/1211.5063*, vol. 2, 2012.

[28] D. Li, D. Chen, J. Goh, and S.-k. Ng, "Anomaly detection with generative adversarial networks for multivariate time series," *arXiv:1809.04758*.

[29] X. Lou, C. Tran, D. K.Y. Yau, R. Tan, H. Ng, T. Zhengjia Fu, and M. Winslett, "Learning-based time delay attack characterization for cyber-physical systems," *IEEE SmartGridComm - Workshop: AI in Energy Systems (Invited Paper)*, 2019.

[30] C. Feng, T. Li, and D. Chana, "Multi-level anomaly detection in industrial control systems via package signatures and lstm networks," *IEEE/IFIP DSN*, pp. 261–272, 2017.

[31] J. Inoue, Y. Yamagata, Y. Chen, C. M. Poskitt, and J. Sun, "Anomaly detection for a water treatment system using unsupervised machine learning," *IEEE ICDMW*, pp. 1058–1065, 2017.

[32] J. Goh, S. Adepu, M. Tan, and Z. S. Lee, "Anomaly detection in cyber physical systems using recurrent neural networks," *IEEE HASE*, pp. 140–145, 2017.

[33] D. Ding, Q.-L. Han, Y. Xiang, X. Ge, and X.-M. Zhang, "A survey on security control and attack detection for industrial cyber-physical systems," *Neurocomputing*, vol. 275, pp. 1674–1683, 2018.

[34] R. Mitchell and I.-R. Chen, "A survey of intrusion detection techniques for cyber-physical systems," *ACM Computing Surveys*, vol. 46, no. 4, pp. 55:1–55:29, 2014.

[35] C. S. Wickramasinghe, D. L. Marino, K. Amarasinghe, and M. Manic, "Generalization of deep learning for cyber-physical system security: A survey," *IEEE IECON*, pp. 745–751, 2018.

[36] M. Schetzen, *Linear time-invariant systems*. J Wiley-Interscience, 2003.

[37] A. Abbasspour, A. Sargolzaei, M. Victorio, and N. Khoshavi, "A neural network-based approach for detection of time delay switch attack on networked control systems," *Procedia Computer Science*, vol. 168, pp. 279–288, 2020.

[38] F. Casella and A. Leva, "Modelling of thermo-hydraulic power generation processes using modelica," *Mathematical and Computer Modelling of Dynamical Systems*, vol. 12, no. 1, pp. 19–33, 2006.

[39] P. Fritzson, P. Aronsson, A. Pop, H. Lundvall, K. Nystrom, L. Saldamli, D. Broman, and A. Sandholm, "Openmodelica-a free open-source environment for system modeling, simulation, and teaching," *IEEE CACSD*, pp. 1588–1595, 2006.

[40] P. Kundur, "Power system stability and control. mcgrawhill, london, 1994," 2012.

[41] P. Simulator, "Version 10.0 scopf," *PVQV, PowerWorld Corporation, Champaign, IL*, vol. 61820, 2005.

[42] Z. C. Lipton, "A critical review of recurrent neural networks for sequence learning," *arXiv:1506.00019*.

[43] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[44] S. Hochreiter, Y. Bengio, P. Frasconi, and J. Schmidhuber, "Gradient flow in recurrent nets: the difficulty of learning long-term dependencies," *A field guide to dynamical recurrent neural networks, IEEE Press*, 2001.

[45] R. Tan, H. H. Nguyen, E. Y. Foo, X. Dong, D. K. Yau, Z. Kalbarczyk, R. K. Iyer, and H. B. Gooi, "Optimal false data injection attack against automatic generation control in power grids," in *2016 ACM/IEEE 7th International Conference on Cyber-Physical Systems (ICCPS)*. IEEE, 2016, pp. 1–10.

[46] J. D. Glover, M. S. Sarma, and T. Overbye, *Power system analysis & design, SI version*. Cengage Learning, 2012.

Prakhar Ganesh received the B.Tech. degree in computer science and engineering from the Indian Institute of Technology (IIT) Delhi, India, in 2019. He is currently a Research Engineer with the Advanced Digital Sciences Center (ADSC), a research center established by the University of Illinois at Urbana Champaign (UIUC) in Singapore. His research interests include faster and interpretable deep learning models robust to real world applications.

Xin Lou is a Research Scientist at Advanced Digital Sciences Center (ADSC), a research center established by the University of Illinois at Urbana Champaign (UIUC) in Singapore. He is also a research affiliate at the Coordinated Science Laboratory (CSL) at UIUC. He was a postdoctoral Researcher (2016-2018) at ADSC. He received the Ph.D. (2016) degree in computer science from City University of Hong Kong, Hong Kong SAR and B.E. (2005) degree in telecommunication engineering from Sichuan University, China. His research interests include cyber-physical systems, deep learning for sensing and computing, nonlinear optimization and distributed algorithms.

Yao Chen is a research scientist in the Advanced Digital Sciences Center, Singapore, which is a research institute of University of Illinois at Urbana-Champaign (UIUC). He is also a research affiliate at the Coordinated Science Laboratory (CSL) at UIUC. He received the B.S. and Ph.D. degree from Nankai University, Tianjin, China in 2010 and 2016, respectively. His research interests include reconfigurable computing, high level synthesis, high performance computing and machine learning.

Rui Tan (M'08-SM'18) is an Assistant Professor at School of Computer Science and Engineering, Nanyang Technological University, Singapore. Previously, he was a Research Scientist (2012-2015) and a Senior Research Scientist (2015) at Advanced Digital Sciences Center, a Singapore-based research center of University of Illinois at Urbana-Champaign (UIUC), a Principle Research Affiliate (2012-2015) at Coordinated Science Lab of UIUC, and a postdoctoral Research Associate (2010-2012) at Michigan State University. He received the Ph.D. (2010) degree in computer science from City University of Hong Kong, the B.S. (2004) and M.S. (2007) degrees from Shanghai Jiao Tong University. His research interests include cyberphysical systems, sensor networks, and ubiquitous computing systems. He received the Best Paper Awards from IPSN'17, CPSR-SG'17, Best Paper Runner-Ups from IEEE PerCom'13 and IPSN'14.

David K.Y. Yau received the B.Sc. from the Chinese University of Hong Kong, and M.S. and Ph.D. from the University of Texas at Austin, all in computer science. He has been Professor at Singapore University of Technology and Design since 2013. Since 2010, he has been Distinguished Scientist at the Advanced Digital Sciences Centre, Singapore. He was Associate Professor of Computer Science at Purdue University (West Lafayette). He received an NSF CAREER award. He won Best Paper award in 2017 ACM/IEEE IPSN and 2010 IEEE MFI. His papers in 2008 IEEE MASS, 2013 IEEE PerCom, 2013 IEEE CPSNA, and 2013 ACM BuildSys were Best Paper finalists. His research interests include cyber-physical system and network security/privacy, wireless sensor networks, smart grid IT, and quality of service. He serves as Associate Editor of IEEE Trans. Network Science and Engineering and ACM Trans. Sensor Networks. He was Associate Editor of IEEE Trans. Smart Grid, Special Section on Smart Grid CyberPhysical Security (2017), IEEE/ACM Trans. Networking (2004-09), and Springer Networking Science (2012-2013); Vice General Chair (2006), TPC co-Chair (2007), and TPC Area Chair (2011) of IEEE ICNP; TPC co-Chair (2006) and Steering Committee member (2007-09) of IEEE IWQoS; TPC Track co-Chair of 2012 IEEE ICDCS; and Organizing Committee member of 2014 IEEE SECON.

Deming Chen received his B.S. degree in computer science from University of Pittsburgh, PA, USA in 1995 and his M.S. and Ph.D. degrees in computer science from the University of California at Los Angeles in 2001 and 2005, respectively. He is the Abel Bliss Endowed Professor of Engineering at University of Illinois at UrbanaChampaign. Dr. Chen is an IEEE Fellow, an ACM Distinguished Speaker, and the Editor-in-Chief of ACM Transactions on Reconfigurable Technology and Systems (TRETS). His current research interests include system-level and high-level synthesis, machine learning, GPU and reconfigurable computing, computational genomics, and hardware security.

Marianne Winslett received the Ph.D. degree in computer science from Stanford University, Stanford, CA, USA, in 1987. She has been an Assistant, Associate, full, Adjunct, and Research Professor with the Department of Computer Science, University of Illinois at UrbanaChampaign, Urbana, IL, USA, since 1987, where she is currently a Research Professor Emerita. She served as the Director of the Advanced Digital Sciences Center in Singapore, where she is now a Distinguished Principal Research Scientist. She is also a venture partner at R3i Ventures. Her research interests include information management and security. Dr. Winslett is a fellow of ACM. She received the Presidential Young Investigator Award from the U.S. National Science Foundation in 1989. She served as an SIGMOD Vice Chair from 2001 to 2005. She has served on the editorial boards of ACM Transactions on Database Systems, IEEE Transactions on Knowledge and Data Engineering, the Very Large Data Bases Journal, ACM Transactions on Information and System Security, and ACM Transactions on the Web, where she also served as co-Editor-in-Chief.