# Domain Adaptation of Deep Learning (D)DoS Attack Detection Algorithms for CPS and IoT

Vicky Ngo*, Mahsa Mohaghegh*, Roopak Sinha*

*Computer Science and Software Engineering Department, Auckland University of Technology, Auckland, New Zealand, Email: zvd0712@autuni.ac.nz, {mahsa.mohaghegh, rsinha}@aut.ac.nz

***Abstract*—Domain adaptation and transfer learning have had applications in computer vision and natural language processing, where machine learning models can be adapted for use in similar tasks across different domains. Not only domain adaptation allows existing detection models to leverage existing knowledge toward solving a more specific but similar problem, but it also addresses the lack of labelled training data since the model is already trained previously. Given that lack of data is an ongoing concern for (D)DoS detection models in cyber-physical systems and Internet-of-Things, domain adaptation of existing models is a promising solution. However, there is currently minimal research into adapting existing detection models into different detection domains and the challenges in doing so. In this study, we proposed a hypothesis on the minimum conditions required for domain adaptation of a detection model across different detection domains. To verify our hypothesis, we performed domain adaptation on two (D)DoS attack detection models in Internet-of-Things and cyber-physical systems, respectively. The two models were tested on a resource-constrained environment to evaluate their detection performance and computational overhead. The results have shown that our hypothesis holds in our test cases. We discover that the detection models architecture determines the scope to which the models can learn transferrable knowledge, which in turn affect their detection accuracy in different detection domain. Furthermore, low detection accuracy and occasional spiking in computational overhead mean that the models might not be well-suited for deployment in a real resource-constrained system. Given the limitations of our study, further work is required to validate how well our hypothesis generalizes to cybersecurity in Internet-of-Things and cyber-physical systems domains as a whole.**

***Index Terms*—Article submission, IEEE, IEEEtran, journal, LaTeX, paper, template, typesetting.**

## ACKNOWLEDGEMENT

Compute services provided by the ECMS Data Centre. Compute services managed by Bumjun Kim, Technical Coordinator - Information Tech bumjun.kim@aut.ac.nz and ICT. The data centre is part of the School of Engineering, Computer and Mathematical Sciences, Auckland University of Technology.

## I. INTRODUCTION

In 2006, Dr. Helen Gill from the United States National Science Foundation coined the term "cyber-physical system" (CPS) [1]. Specifically, CPS can be broadly defined as the integration of communication, control, and software components into physical processes. CPS has multiple industrial applications in smart grids, industrial control systems, and manufacturing processes [2]. In such cases, CPS operations tend to be continuous in nature and often have higher availability requirements. Therefore, interruptions to these processes can be detrimental to the system's operations a nd, in rare cases, can cause serious damage to the physical devices. As such, denial-of-service (DoS) and distributed-denial-of-service (DDoS) attacks are great concerns for CPS and must be prevented.

Internet-of-Things (IoT) can be considered a subset of CPS. By definition, IoT is the interconnection of physical objects, or "things," embedded with sensors, software, and other technologies that enable them to collect and exchange data over the internet [3]. IoT is often seen in smart homes, healthcare, and other industrial applications, which are commonly referred to as Industrial Internet-of-Things (IIoT). IoT infrastructure is susceptible to (D)DoS attacks that threaten IoT devices security [4]. While CPS and IoT are ultimately two different but related domains, common (D)DoS attack techniques in IoT have been shown to also overlap with (D)DoS attacks present in CPS [5].

In the literature, various (D)DoS detection algorithms based on ML and DL have been proposed for both IoT and CPS. However, a majority of detection algorithms in CPS did not consider the actual applications of these models in resource-constrained industrial systems, nor sufficiently address existing (D)DoS attack techniques targeting CPS [6]. According to [6], the lack of up-to-date datasets of current (D)DoS attack techniques to train detection models on is also a large factor. Transfer learning, or domain adaptation, of machine learning and deep learning detection models are common methods to address the lack of training data, mainly in natural language processing and computer vision, but also in cybersecurity. Given that CPS and IoT are related, it is possible to use domain adaptations of (D)DoS attack detection models in IoT for use in the CPS domain and vice versa. In this study, we propose the following research questions:

- RQ1: Are existing deep learning models capable of domain adaptation for detecting (D)DoS attacks in different detection domains (IoT, network, CPS)?
- RQ2: What challenges are present when adapting machine learning models to different datasets?
- RQ3: How are computational overheads affected when adapting machine learning models for detection in different domains?

Following these research questions, we proposed the

following hypothesis which this study seeks to verify, particularly about the training of models in a different detection domain.

**A deep learning (D)DoS detection model is capable of domain adaptation for cross-domain detection given that it satisfies the following two conditions**

- **C1:** The model is able to extract relevant features for training.
- **C2:** The detection algorithm source code provides sufficient soft coding and generalisation such that it can easily be adapted to datasets from different domains.

Through performing the experiments set out in this study, we made the following contributions.

- Verify the aforementioned hypothesis on domain adaptation of (D)DoS detection models in IoT and CPS domains.
- Explore existing challenges and points of consideration for domain adaptation in detection models.

This study is organized as follows. Sect. II describes existing (D)DoS detection algorithms and domain adaptation for CPS and IoT, and Sect. III lays out the controlled experiments protocol used in this study to verify our hypothesis. Sect. IV continues with our experiment design. Then, the implementation details are presented in Sect. V. Finally, Sect. **??** concludes the findings and future works.

## II. Background

### A. Deep Learning (D)DoS Attack Detection Algorithms in CPS

Deep learning algorithms are known to achieve better accuracy than machine learning algorithms by utilising neural networks. In recent years, researchers have started to consider machine learning (ML) and deep learning (DL) algorithms to assist in cyberattack detection in CPS [7]. Particularly, there has been an increase in using deep neural network algorithms for attack detection in CPS, such as RNN, CNN, and autoencoder [8]. In another systematic mapping study of state-of-the-art deep learning models for industrial control systems, [6] also found similar results were found where CNN, autoencoders, and LSTM (a type of RNN) were used more often.

False data injection and malware are two common DDoS attacks in CPS. The anomaly detection approach has been considered highly effective in detecting these attacks, which have also been employed in various deep learning detection models [8]. A smaller number of algorithms use a signature based approach, with a small percentage taking advantage of both approaches [8]. An anomaly detection model learns the normal behaviour of the chosen systems, and classifies other anomalies and deviations as potential attacks. Thus, detection models following this approach are capable of detecting a wider range of attacks, including zero-day attacks.

### B. Deep Learning (D)DoS Attack Detection Algorithms in IoT

Deep learning detection models also have various applications in IoT and smart homes. It is worth noting that the (D)DOS attack types present in the IoT domain are vastly different compared to (D)DoS attack in CPS. Particularly, attacks in IoT include application layer attacks, botnet attacks, and transport layer attacks. These attacks are common not only in IoT, but also target web applications and computer networks [9]. In a survey of recent attack detection models in IoT by [10], variations of neural networks and RNNs are most common in addressing these attacks. We also discovered that the majority of the models reported in [10] use a semi-supervised training approach, with both labelled and unlabelled data used to train the detection models. Furthermore, the three algorithms CNN, ANN, LSTM are shown to have returned the best detection performance in terms of accuracy for the IoT domain [11].

### C. Domain Adaptation of Deep Learning (D)DoS Algorithms

Traditional deep learning algorithms assume that the feature space of the training data (source domain) is the same as the feature space of the data on which the model is applied (target domain) [12]. In fact, common practise usually splits a chosen dataset into training, testing, and validation sets to train and evaluate a machine learning model, whereas the feature spaces of all three sets are the same. However, this assumption may limit the applicability of deep learning models because the amount of data, in the target domain may have different feature spaces, especially for unlabelled data [12].Therefore, domain adaptation of deep learning models for cybersecurity is needed, where data is scarce [12]. Domain adaptation is interchangeable with transfer learning.

Domain adaptation is considered a subset of transfer learning [13]. The term is interchangeable with transductive transfer learning, where both the source and target tasks are the same [13]. In this case, the data from the source domain is labelled, whereas the data from the target domain are not. According to [13], two types of domain adaptation exist:

- The feature spaces between the source and target domains are different.
- The feature spaces between domains are the same, but the marginal distribution of the input data is different.

In our study, we primarily consider the first case, where the feature spaces of the source and target domains are different. This allows us to address domain adaptation through using datasets from different domains for testing and training. We conducted a systematic search of existing studies for transfer learning and its subset domain adaptation in the sections that follow. The following sample search string was used in our search.

*"domain adaptation" AND "deep learning" AND (cyber physical system OR industrial control system) AND (ddos OR dos OR denial-of-service)*

As seen in Figure 1, there is a lack of research into domain adaptation for deep learning detection algorithms in cyber physical systems. This presents an urgency to address this research gap.

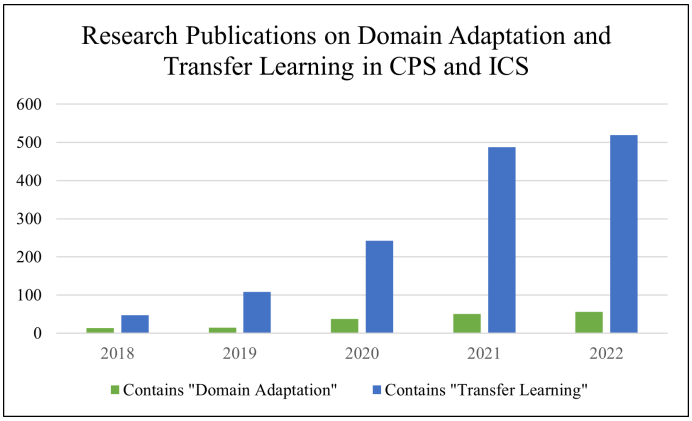


Fig. 1. Research publications in attack detection using ML & DL

According to [14], deep generative models have been applied for domain adaptation in Industrial Internet of Things (IIoT). These include generative adversarial networks (GANs), autoregressive models (ARs), variational autoencoder (VAE) [14], [15]. Below, we discuss some recent works on this topic.

## III. Research Method

This section discusses the research approach and strategy that we used to test our hypothesis, as described in I.

### A. Choosing a Research Strategy

A research strategy is described as methods used to systematically produce a solution to a given research problem [16]. It is important to establish the chosen research plan and methodology clearly before conducting the actual research work. Likewise, selecting an appropriate research strategy is an equally important task. Below, we examined six different research approaches while considering how to design and evaluate our hypothesis [17].

- **Controlled Experiments**: A controlled experiment involves investigating a testable hypothesis where the relationship between the independent and dependent variables is explored [17]. This is done by making controlled changes to the independent variables and observing the effect these changes have on the dependent variables. It is important that the experiments be conducted in a controlled environment, where other external factors aside from the independent variables must not be allowed to influence the experiment results. This is to ensure the validity of the cause-effect relationship established upon observation of the dependent variables [17].
- **Case Studies**: A case study can be understood as an empirical enquiry that seeks to explain why a certain phenomenon occurs within a real-life context and reveal any relevant cause-and-effect relationships [17]. Case study research uses purposive sampling to select the most relevant cases to the study proposition, but it is also more open to researcher bias [17].
- **Survey Research**: Survey research is a quantitative method to collect information from a representative sample, from which generalisations can be drawn and applied to the wider population [17]. Survey research often involves the use of questionnaires and different sampling methods to collect data. However, it is important to also control for sampling bias to ensure that the selected sample is representative of the larger population [17].
- **Ethnographies**: Ethnography focuses on field observations of a target community in their own environment, and thus has a more sociological focus [17]. While no pre-existing theories are assumed for ethnography research, it is crucial to avoid preconceptions. In the context of software engineering, ethnography would be helpful for studies focusing on aspects of particular technical communities, such as work practises [17].
- **Action Research**: In action research, the researcher studies methods to solve an existing problem, while also documenting and analysing the process of solving that problem [17]. Therefore, action research tends to imply organisational commitment that comes with implementing the solution proposed and observing its impacts [17].
- **Design Science**: Design science approach involves the study and creation of an artefact for a specified problem domain [18]. The artefact must be rigorously evaluated, and proven to solve the specified problem in a more effective manner that is of interest to the general population [18]. Additionally, the design science research must be communicated effectively to both technical and managerial audiences.

Through careful evaluation of the six proposed research approaches, we considered the controlled experiment method to be the most fitting to evaluate our hypothesis.

### B. Controlled Experiments

A controlled experiment is an investigation of a testable hypothesis where the effect of manipulating independent variables on dependent variables are evaluated [17]. Controlled experiments allow us to determine whether a cause-effect relationship exists between these variables, and how they are related to each other. As such, the precondition of a controlled experiment is a clear and testable hypothesis, which will guide the subsequent steps in the experiment as well as deciding what variables are to be included. In a controlled experiment, there are three important types of variables:

- **Independent Variables**: Independent variables are those that will be changed across the experiment to observe the effect that these changes have on the dependent variables. In our experiments, the independent variables are the models to be trained, and the datasets that each model was trained and evaluated with.
- **Dependent Variables**: Dependent variables are variables whose value depends upon changes made by independent variables. We seek to find if a cause-effect relationship exists between the dependent and independent variables. These are the performance metrics of the trained models upon evaluation, including computational overhead.

- **Controlled Variables**: Any variables other than the chosen independent variables that can affect the dependent variables are considered controlled variables. It is crucial not to let the controlled variables affect the experiment results to ensure a fair experiment. In this study, the main controlled variables include the consistency of the datasets and the evaluation environment used. The extent to which we modify the chosen models' source code must also be controlled to reliably adapt the models without affecting their core architecture and intended purposes.

Additionally, the method in which we evaluate the experiment empirical validity is also of importance. [17] identify the following four criteria for validity:

- *Construct validity* focuses on whether theoretical constructs and abstract definitions are interpreted and measured correctly. For example, different researchers may have different interpretations of the terms "detection performance" and "computational overhead." As a result, we have explicitly defined the meanings of these two terms in Sect. IV.
- *Internal validity* focuses on the study design and whether the results follow from the data. Allowing uncontrolled variables outside independent variables to affect dependent variables is a common threat to internal validity.
- *External validity* focuses on whether the generalisation of results and subsequent claims are justified. Specifically, how do the results of domain adaptation experiments on our chosen models generalise to the hypothesis, and whether such generalisation is justified?
- *Reliability* focuses on whether the experiment is reproducible and whether the same results can be achieved if other researchers replicate the study design. To maximise reproducibility, we have described the experiment's implementation in detail in Sect V, as well as made the source code we used and other technical details publicly available in [19].

## IV. Methodology

The purpose of this study is to test the capabilities of existing models for domain adaptation. Therefore, we will only be making minor changes to the source code as necessary to adapt them in our test environments. Major changes to the functions and model architecture to adapt the models to different domains are out of scope for this study, in which case we will evaluate the models' best performance in those domains instead. We choose two models: one for DDoS detection in an IoT network and another for DDoS detection in an ICS network. Two datasets for (D)DoS attacks in each domain are chosen as the baselines for comparison. We will then train and test each model's performance on each dataset to evaluate its capabilities for domain adaptation. Further details on the experiment procedure, models, and dataset selection are described below, with implementation details discussed in Sect. V.

### A. Models Selection

It is known that RNN (including LSTM), CNN, and autoencoders are frequently used for deep learning attack detection models [8]. Furthermore, GAN models are also very flexible for domain adaptation. As such, we used the following model selection criteria for our experiments:

- The model must either be RNN, LSTM, CNN, autoencoder, GAN, or similar hybrid model.
- The model implementation must have been published as a research paper to verify credibility. Only models that were published or last updated within 2019 to 2022 are considered, as older models are most likely to be outdated.
- The models must be applicable for detection in either CPS or network/IoT.
- The model source code must be available to reproduce its performance.

Using these search criteria, we have chosen the following two models: LUCID and MAD-GAN.

**LUCID** is a CNN-based (D)DoS detection model that can be deployed in online resource-constrained environments, which has been tested in edge computing [20]. The LUCID architecture is composed of a novel data preprocessing algorithm and the LUCID model itself. The main LUCID training objective is to minimise its cost function by iteratively updating all the model's weights and biases, also known as trainable parameters. To optimise accuracy, LUCID performs a grid-search through a set of hyperparameters with F1 as the evaluation metric. The training continues indefinitely for each point in the grid until the loss does not decrease for a consecutive 25 times. The F1 score is then saved before the model moves on to the next point.

The LUCID model is tightly implemented around the TCP/IP protocol, and thus will be able to label and train on data as long as the attackers and victims can be identified with the following 5-tuple: source IP, source port, destination IP, destination port, protocol [21]. The labelled samples are then compiled and balanced so that the number of benign and malicious samples is the same before they are divided into training, validation, and test sets. The percentages are 81%, 9%, 10% respectively. LUCID will then train multiple models on various hyperparameters before output the model with best F1-score. The model has been made to be used in both real-time detection and offline detection on.pcap files [21]. That means that we can adapt the LUCID model to any dataset as long as LUCID is able to label training data correctly using the aforementioned 5-tuple.

We use the latest version, updated on June 24, 2022, which can be found in [21]. At the time of publication, the LUCID model was trained on the CICIDS2017 dataset, and evaluated across another three datasets namely ISCX2012, CSECIC2018, and a combined version of all three datasets called UNB201X [20]. All datasets came from the University of New Brunswick.

**MAD-GAN** is a GAN model incorporating LSTM-RNN for (D)DoS detection in CPS [22], [23]. The MAD-GAN model consists of a generator and a discriminator as two

LSTM-RNNs (Long Short Term Memory Recurrent Neural Networks). The MAD-GAN model trains the generator and discriminator on the normal behaviours of devices in the CPS network in an unsupervised manner. The generator learns to reconstruct data similar to the CPS network's normal behaviours, while the discriminator learns to classify between real and generated data. Both the generator and discriminator generate losses during their training, which function as feedback to improve their performance. In other words, the generator and discriminator train each other based on the discriminator's loss and the generator's loss. Overall, the MAD-GAN anomaly detection approach consists of two parts: discriminator-based anomaly detection and reconstruction-based anomaly detection.

The MAD-GAN model is initialised using command-line input or a.txt file containing the values for selected parameters. The model is then trained on common behaviours like.csv files and numpy arrays (.npy files).Testing and evaluation are then performed on attack traffic recorded in either .csv or .npy formats. At the time of publication, the MAD-GAN model was trained and tested on the 2016 version of the SWaT dataset, along with KDD99 and WADI datasets [22].

### B. Data Selection and Balancing

For this research, we prioritise using datasets supported by the two chosen models, which are CICDDoS2019 [24], [25] and SWaT [26], [27] datasets. CICDDoS2019 contains (D)DoS attacks commonly found in computer networks and IoT devices [24]. It is frequently used by deep learning researchers for (D)DoS detection in networks or the IoT. Meanwhile, the SWaT dataset contains attacks targeting CPS, including false data injection attacks [27]. The SWaT dataset was collected from a real water treatment testbed and, as such, is also commonly used for the development of deep learning models for attack detection in the CPS environment and relevant domains. Both datasets are sufficiently large for our experiment.

One disadvantage of the CICDDoS 2019 dataset is that it contains 99.9% attack traffic and only 0.1 percent benign traffic [28]. Since both LUCID and MAD-GAN requires large numbers of benign traffic for training, we have decided to include benign traffic from the CIC2017 dataset in addition to CICDDoS2019. Both datasets came from the University of New Brunswick, and the benign traffic in both of them were generated in the same manners. This method works to our advantage because LUCID also supports the CIC2017 dataset in addition to CICDDoS2019. We shall refer to the combined dataset from CICIDS2017 and CICDDoS2019 as the **CICDDoS dataset** collectively throughout this paper.

For the SWaT dataset, we chose the 2019 version of the dataset, which contains historian exfiltration and sensor disruption attacks targeting the SCADA and engineering workstations.

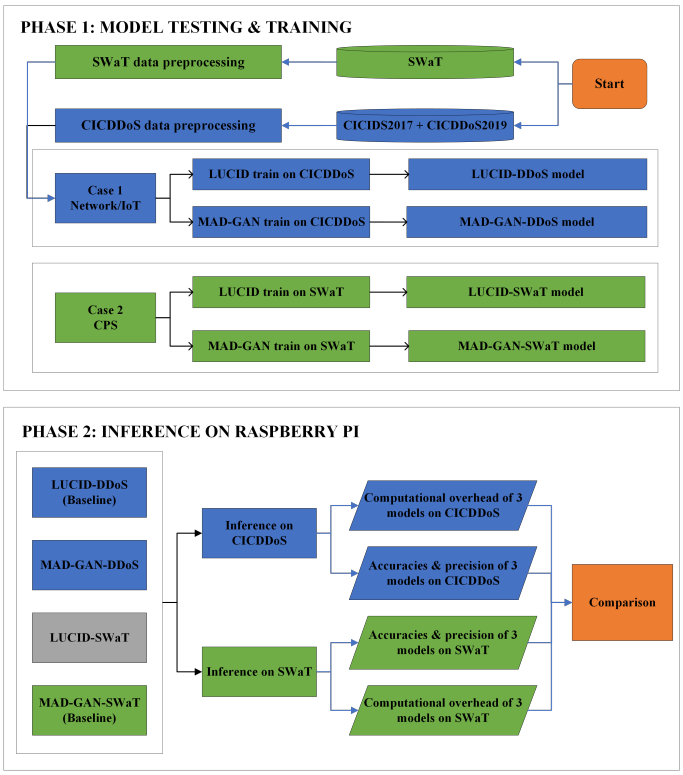


Fig. 2. Experiment Design

### C. Experiment Design

The experiment design is composed of two phases, with the first phase for training and testing the two models and the second phase for evaluating the two models in resource-constrained environments. This is better represented in 2.

For phase 1, the first step involves collecting the required datasets, which are CICDDoS and SWaT. During this step, we identify the types of attacks and the traffic traces they cover in each dataset. Then, we select the appropriate traffic traces for the experiment. This step is necessary due to the fact that it's impractical to train both models over such large datasets where resource and time are our constraints. We will use the metrics of accuracy (Acc), precision (Pre), recall (Rec), and F1 scores (F1) to compare the models' performances. These metrics are defined as follows.

$$Acc = \frac{TP + TN}{TP + FP + TN + FN} \tag{1}$$

$$Pre = \frac{TP}{TP + FP} \tag{2}$$

$$Rec = \frac{TP}{TP + FN} \tag{3}$$

$$F1 = 2 \times \frac{Pre \times Rec}{Pre + Rec} \tag{4}$$

Using the data that we have processed, we propose two test cases. The first test case involves training both models on the CICDDoS datasets. The results of this test case will

be two models for (D)DoS detection in network and IoT environments, which we will refer to as LUCID-DDoS2019 and MAD-GAN-DDoS throughout the rest of this study. LUCID-DDoS will serve as the baseline against which MAD-GAN-DDoS performance will be compared. This is because the original LUCID model was developed for detection in the network and IoT domain [20], making it best suited for the task.

The second test case involves training LUCID and MAD-GAN on the SWaT dataset, which includes (D)DoS attacks in the CPS environment. Similar to case 1, the goal of case 2 is to also produce two models for (D)DoS attack detection in the CPS domain. We will refer to these two models as LUCID-SWaT and MAD-GAN-SWaT throughout the rest of this study, respectively. Given that MAD-GAN has been proposed for detection in the CPS domain [22], MAD-GAN-SWaT performance will serve as the comparison baseline for case 2.

It is expected that we will have four models trained by the time we reach phase 2 of the experiment, which are LUCID-DDoS2019, MAD-GAN-DDoS, LUCID-SWaT, and MAD-GAN-SWaT. However, we discovered during the implementation that the LUCID model was not able to train on the SWaT dataset efficiently, as reported in Sect. V. To reflect this, the experiment design in Fig. 2 shows a grayed-out LUCID-SWaT model. Therefore, only the three models, LUCID-DDoS, MAD-GAN-DDoS, and MAD-GAN-SWaT will be deployed on the Raspberry Pi for evaluation. The purpose of phase 2 is twofold:

- To test each model's performance when given a detection task outside its intended detection domain. If a model is capable of domain adaptation, its detection performance is only expected to decrease by a small percentage. Otherwise, that model will identify (D)DoS attacks as benign traffic.
- To test the effect of domain adaptation on each model's computational overhead when deployed in a resource-constrained environment. For this, we will compare each model's performance when evaluated against familiar data from the same dataset versus unfamiliar data from a different domain. This allows us to dismiss any difference from evaluating the models on a GPU versus the same task on a Raspberry Pi.

When deployed on the Raspberry Pi, we will feed each model with pre-recorded traffic traces in the .pcap and .csv format from both CICDDoS and SWaT datasets. All the traffic traces are unseen data. During our comparison, we will observe any changes to the detection performance, specifically in terms of accuracy, precision, recall, and F1 score. We will also measure the time taken to perform the detection task and the computational overhead of the models. In our study, we define computational overhead as the CPU and Random Access Memory (RAM) usages of each model when detecting DDoS attacks on the Raspberry Pi.

TABLE I
LUCID & MAD-GAN TRAINING DEPENDENCIES

| LUCID | MAD-GAN |
|---|---|
| Python 3.9.13 | Python 3.6.13 |
| numpy 1.23.3 | numpy 1.19.2 |
| scipy 1.9.1 | scipy 1.1.0 |
| tensorflow 2.4.1 | tensorflow 1.15.1 |
| scikit-learn v1.1.3 | scikit-learn 0.19.1 |
| tshark 3.6.2 | pandas 0.22.0 |
| pyshark 0.5.3 matplotlib 2.1.1 | keras 2.1.2 |
| | bleach 1.5.0 |

## V. IMPLEMENTATION

### A. Hardware and Software

Both models were developed using the TensorFlow machine learning libraries. For our training and testing process in phase 1, we used a GeForce RTX 3080 graphics card on an Ubuntu 22.04 machine with 1 TB data storage and 128GB RAM. The processor is a Xeon W-2265 3.5 12C 24T 3.5 4.8 GHZ 19.25 MB cache, with 12 CPU cores and 24 threads. We used a Raspberry Pi 4 Model B Rev 1.4 with 7.6GB RAM for the second phase.

Because LUCID and MAD-GAN each used different versions of TensorFlow and the related dependencies, certain commands required older versions of the software. As such, we created two separate Anaconda environments for each model to resolve this difference. Doing so will also allow us to monitor the experiments on each model more effectively. Table I shows the core dependencies we installed for each model.

### B. Datasets

The CICDDoS2019 dataset collects (D)DoS attack traffic data over 2 days, with 7 attacks in the first day and 12 attacks in the second day, totalling up to 13 different types of attack [25]. The packet captures in this dataset are publicly available in .pcap format, including full payloads. They were generated by using profiles representing abstract behaviours of human interactions with the testbed, allowing for the generation of naturalistic benign background traffic.

We select traffic from each of the 13 types of attacks from both days and an equivalent amount of benign traffic. As for the CIC2017 dataset, we select only the traffic collected on Monday, July 3rd that is fully benign. Combining this dataset portion with the CICDDoS2019 dataset gives us the CICDDoS dataset that will be used for training and testing the models throughout this paper.

With the SWaT dataset, we selected the latest data collected in December 2019, both .pcap and .csv formats.

### C. Implementation of Phase 1 - Case 1: (D)DoS in Network/IoT

In case 1, we trained and tested both the LUCID and MAD-GAN models on the CICDDoS2019 datasets. Because LUCID was developed for network detection, its performance in this phase will be the baseline for comparison. This case yielded

TABLE II
LUCID ADJUSTED HYPERPARAMETERS

| Hyperparameters | Chosen Configuration |
|---|---|
| Learning Rate | LR = [0.1] |
| Convolutional filters | k = [1,2,4,8,16,32] |
| Packets per samples | n = 100 |
| Time windows | t = 100 |
| Epoch | e = 50 |
| Regularization | regularization = r1 |
| Dropout rate | d = 0.5 |

two models trained for (D)DoS detection in networks and IoT.

**LUCID-DDoS2019.** For LUCID, we combined both benign and attack traffic from Monday in the CICIDS2017 dataset, in addition to the attack traffic from the CICDDoS2019 dataset. The LUCID model will be given the processed training data along with a set of hyperparameters as arrays. LUCID will then train using different combinations of the hyperparameters before selecting the one with the best F1-score. As a result, this can lead to high memory and computing resource consumption, which we encountered a few times when using the default configurations. To address this issue while maintaining high accuracy, we have made some changes to the training hyperparameters, as shown in Table II.

Firstly, we set the two hyperparameters n = 100 and t = 100. This allows for the model to achieve the highest F1-score possible despite higher memory, as shown in previous test cases by [20]. Secondly, we fixed the learning rate to 0.01 instead of an array of multiple learning rates. The learning rate of 0.01 was shown in [20] to produce the highest F1-score, which was the basis of our decision. Thirdly, we reduced the number of default epochs from 100 to 50. And finally, we removed k = 64 from the hyperparameters array, which will reduce the number of convolutions needed. More information on other hyperparameters can be found in [20], [21].

**MAD-GAN-DDoS.** The MAD-GAN model's training and testing procedures require two .txt files containing the training and testing parameters for any new dataset. Additionally, a training function and a testing function customised for that particular dataset must also be added to the source code. Naturally, we must also add these two functions to the CICDDoS2019 dataset in addition to the parameter files. We have observed that the existing training and testing functions for SWAT, WADI, and KDD99 datasets follow a similar format, which we adopted for the CICIDS2017 and CICDDoS2019 dataset. We used only Monday traffic CICIDS2017 which is all benign, to train the model. Then, we selected attack traffic for all types of attacks present in the CICDDoS2019 dataset for training. During the data preprocessing processes, we removed non-numerical columns and replaced "NaN" and "Infinity" values with 0 instead. The model was trained over 100 epochs as per the default settings.
The architecture of MAD-GAN offers two methods for anomaly detection [22]. The first method involves using only the discriminator for direct anomaly detection [22]. The second method involves using both the generator and discriminator for detection [22]. However, we have observed that the second detection method did not appear to converge as expected. Similar issues have been previously raised in 2019, but no solutions were available [23]. As such, we have decided to only use the discriminator to test and evaluate the MAD-GAN model.

### *D. Implementation of Phase 1- Case 2: (D)DoS in CPS*

In case 2, we trained and tested both the LUCID and MAD-GAN models on the SWaT datasets. We chose data from December 2019, which includes both benign and malicious traffic in.csv and.pcap formats with full payloads. The dataset can be obtained by contacting the authors [26].

**LUCID-SWaT.** LUCID can adapt to any dataset with a simple change to the source code [21]. To do so, we first identify the IP addresses of the attacker machines and victim machines in the SWaT dataset. This information is crucial for LUCID data preprocessing algorithm to label benign and attack traffic accurately before training [21]. Then, we feed the traffic data to the data preprocessing algorithm which splits them into train/validation/test set with respective percentages being 81%/,9%,10%.

However, we have discovered that LUCID's dependency on source and destination IP addresses of packets to correctly label benign and (D)DoS traffic proved to be a difficulty in adapting the model to the SWaT dataset. As the SWaT dataset features mainly false data injection attacks, packets are hijacked as they communicate between the SCADA and PLCs of the testbed. This means that there are no definite attackers' IP addresses to help LUCID identify attacks and benign traffic for training. Furthermore, false data injection attacks and other (D)DoS attacks in the CPS domain typically require the detection model to analyse packet payloads, which LUCID does not consider in its algorithm. The design of the LUCID detection model only works for common (D)DoS attacks using the TCP/IP protocol with definite attacks and victims' IP addresses. As a result, the current LUCID model cannot be trained to detect application-layer protocol-based attacks, such as the SWaT testbed's Common Industrial Protocol (CIP) [21].Training a LUCID-SWaT model without modification to the source code is not possible at this stage.

To conclude, the design of the LUCID model violates condition C1 in our hypothesis in regard to adapting it to the SWaT dataset. Therefore, this outcome still aligns with our initial hypothesis. Improving the LUCID model for detection in CPS through further modification of the source code is out of scope for this study, which we will leave for future work.

**MAD-GAN-SWaT.** In this particular test, we train the MAD-GAN model on the 2019 version of the SWaT dataset, which is smaller and has fewer attacks compared to the

2016 version. While the detection technique is the same, we encountered multiple issues when training MAD-GAN with the 2019 version of the SWaT dataset. To identify the core problems, we trained two identical MAD-GAN models in parallel on each version of the SWaT dataset and made comparisons. This brings to our attention the fact that the small sample sizes of the 2019 SWaT dataset have led to negative values in multiple calculations and array inputs, causing the training errors. In fact, a dataset at least three times larger than the SWaT 2019 dataset is needed for the MAD-GAN model to train properly. In fact, the SWaT 2016 dataset is approximately 40 times larger than the SWaT 2019 datasets.

To solve this problem, we extended the sample sizes by duplicating existing data in the SWaT 2019 dataset for both normal and attack behaviours. In principle, this solution discards any assumptions of independence in the data samples and, at the same time, leads to model overfitting. We considered combining parts of the SWAT 2016 dataset with the SWAT 2019 dataset as padding, but this solution was ruled out for the following reasons:

- The measurements in the SWaT 2016 dataset are inconsistent with the measurements recorded in the SWaT 2019 datasets. This might have been caused by varying test bed states as a result of a different set of attacks being conducted in the two recording instances.
- If we combine both datasets so that MAD-GAN can train without problems, the minimum number of samples in the SWaT 2016 dataset will be greater than the number of samples in the SWaT 2019 dataset. Taking into consideration the inconsistency presented in the two dataset versions, it would be more efficient to simply train the model with the SWaT 2016 dataset instead of merging. However, this does not align with our objectives.

Similar to the case of MAD-GAN-DDoS, we only performed detection with the trained discriminator of the model, due to the fact that the model was not able to converge when using both the discriminator and the generator. The outcomes of the experiment in case 2 are those of the MAD-GAN-SWaT model only. While this is different from our expected outcomes in the experiment, designed as per Sect. IV, these outcomes still align with our initial hypothesis.

### *E. Implementation of Phase 2: Models Evaluation on Raspberry Pi 4*

Following the experiments performed in case 1 and case 2, we made copies of the 3 trained models, namely LUCID-DDoS, MAD-GAN-DDoS, MAD-GAN-SWaT, and evaluate their performance in a development environment in a Raspberry Pi 3B+. It should be noted that LUCID was run in .pcap files inference mode, as opposed to performing detection on the testing set during Phase 1.

To test their performance, each model will be given recorded .pcap and .csv files from the SWaT and CICDoS datasets. While it is more efficient to test the models in real-time traffic inference, there are several hardware limitations in our experiments:

TABLE III
LUCID & MAD-GAN RASPBERY PI TESTING DEPENDENCIES

| **LUCID** | **MAD-GAN** |
|---|---|
| Python 3.9.13 | Python 3.10.6 |
| numpy 1.23.3 | numpy 1.19.2 |
| scipy 1.9.1 | scipy 1.1.0 |
| tensorflow 2.4.1 | tensorflow 1.15.1 |
| scikit-learn 1.1.3 | scikit-learn 0.19.1 |
| tshark 3.6.2 | pandas 0.22.0 |
| pyshark 0.5.3 | matplotlib 2.1.1 |
| | keras 2.1.2 |
| | bleach 1.5.0 |

- The Raspberry Pi 4 does not have sufficient hardware capabilities to run the models in real-time mode without technical difficulties.
- We currently do not have a sufficient CPS network and computer network to generate normal and attack traffic realistically.
- While the LUCID model has an option for live inference, the MAD-GAN model did not feature any real-time detection function. Given the hardware limitations, it is fairer to test variations of both models with pre-recorded traffic.

Because the two MAD-GAN models and the LUCID-DDoS model require different dependencies, we used two SD cards of sizes 32 GB and 16 GB to set up two different environments for MAD-GANs and LUCID, respectively. When setting up the environment for testing each model on the SD cards, we used miniforge instead of miniconda3 as the former is more compatible with the Raspberry Pi operating system. However, we discovered that it was not possible to install the dependencies required by MAD-GAN on the Raspberry Pi operating system, similar to the testing phase. Therefore, we decided to use the latest version of MAD-GAN core dependencies instead, and modify the anomaly detection scripts so that they would suit the new environment. We removed an unnecessary import statement for a deprecated function from scipy v1.1.0, which was removed in the newer version. Then, we used the tf.compat module to allow deprecated functions from Tensorflow v1.X to run smoothly on Tensorflow v2.X. The dependencies used by the Raspberry Pi are listed in Table III. Note that we only modified the script for running anomaly detection on the newer dependencies, with compatibility in mind. We did not upgrade the MAD-GAN model for it to run entirely on the newer Tensorflow or other packages. We also did not modify the training script, as it is irrelevant in Phase 2, although this could possibly be part of our future works. For the dependencies used in training MAD-GAN, please refer to Table I. Modifying the scripts has no impact on the trained models.

Below, we detail the test cases to be carried out on the Raspberry Pi. Note that the CICDDoS dataset includes both benign and attack traffic, combined from CICIDS2017 and CICDDoS2019. During phase 2, the testing data will have more attack samples than benign samples.

- 1) LUCID-DDoS on CICIDS2017 (benign)

- 2) LUCID-DDoS on CICDDoS2019 (attack)
- 3) LUCID-DDoS on SWaT (benign)
- 4) LUCID-DDoS on SWaT (attack)
- 5) MAD-GAN-DDoS on CICDDoS
- 6) MAD-GAN-DDoS on SWaT
- 7) MAD-GAN-SWaT on CICDDoS
- 8) MAD-GAN-SWaT on SWaT

It is important to note that during evaluation, LUCID only output is the DDoS rate, which depends entirely on the data LUCID trained on. Therefore, it is possible for LUCID to mistakenly classify benign traffic as DDoS traffic. Therefore, we evaluated LUCID-DDoS on datasets that contain either entirely benign traffic or entirely malicious traffic. This would help us roughly evaluate LUCID's accuracy and false positive rate.

Meanwhile, the MAD-GAN models were trained and tested on labelled samples stored in .csv formats, allowing the models to calculate their own accuracy by comparing their predictions to the correct labels. As such, MAD-GAN can be evaluated on a dataset that includes both benign and DDoS traffic.

## VI. Phase 1 - Training Results Analysis

In this section, we present the results obtained from training LUCID and MAD-GAN models on different datasets.

**MAD-GAN models.** The training results of MAD-GAN models were evaluated in terms of the discriminator's loss and generator's loss over each epoch they were trained on. The GAN model design dictates that as the discriminator's loss increases, the generator's loss should decrease, and vice versa, as both the discriminator and generator are training each other. Generally speaking, the two losses should eventually converge toward an optimal number after considerable training time. Both models were trained over 100 epochs.

Fig. 3 shows the discriminators' and generators' losses for both MAD-GAN-DDoS and MAD-GAN-SWaT models during training. The MAD-GAN-DDoS shows a continuous diverging pattern, which suggests that the MAD-GAN-DDoS model's discriminator has become too good at discriminating data produced by the generator, leading to increasing generator loss. There is a larger difference between the discriminator's loss and the generator's loss for the MAD-GAN-DDoS model compared to that of the MAD-GAN-SWaT model. At the same time, the MAD-GAN-SWaT model was shown to have gradually diverging patterns between each of its three peaks, which converge slightly near the end.

The average loss over 100 epochs for MAD-GAN-DDoS and MAD-GAN-SWaT are shown in Table IV. We can also observe that MAD-GAN-SWaT has much lower generator loss compared to MAD-GAN DDoS, and that MAD-GAN-DDoS discriminator loss is also very low. This further proves that MAD-GAN-DDoS discriminator's high performance did not provide sufficient feedbacks to its generator, leading to increasing generator loss. Overall, this has shown that the MAD-GAN model has a more balanced training results when trained on the SWaT dataset compared to training on the CICDDoS dataset.

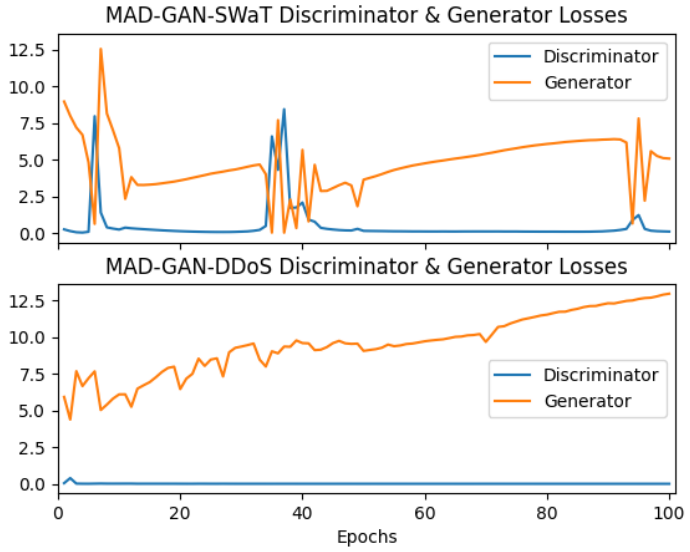


Fig. 3. MAD-GAN Models Training Discriminator & Generator Loss

TABLE IV
MAD-GANs Average Discriminator & Generator loss

| | MAD-GAN-SWaT | MAD-GAN-DDoS |
|---|---|---|
| Discriminator loss | 0.505115 | 0.010799 |
| Generator loss | 4.715256 | 9.491622 |

TABLE V
LUCID-DDoS Training Results

| | Accuracy | F1 score | Precision | Recall |
|---|---|---|---|---|
| Training | 99.42% | 99.43% | 99.19% | 99.67% |
| Testing | 99.42% | 99.42% | 99.20% | 99.63% |

**LUCID model**. After data processing, we extracted 872,590 samples, with 436295/436295 benign/DDoS distribution. The train/validation/test sizes are 706797/78534/87259 samples, which are equivalent to 5297712/583791/653940 packets, respectively. Out of multiples models that the LUCID algorithm trained, the model with the best F1 score on the validation set was chosen. Its performance on the training and testing sets are shown in Table V.

We can observe that LUCID achieved a high detection performance on both the training and testing datasets.

## VII. Phase 2 - Evaluation on Raspberry Pi

### A. MAD-GAN Models

We recorded the predicted DDoS percentage in the dataset when testing with LUCID. On the other hand, MAD-GAN-DDoS and MAD-GAN-SWaT use the standard metrics, namely accuracy (Acc), precision (Pre), recall (Rec), and F1 scores, to evaluate the anomaly detection based on the labelled data.

TABLE VI
MAD-GANs Evaluation Results

| Test Case | Accuracy | F1 score | Precision | Recall | Time (s) |
|---|---|---|---|---|---|
| MAD-GAN-SWaT on SWaT | 55.39% | 28.77% | 37.09% | 27.43% | 1862 |
| MAD-GAN-DDoS on SWaT | 53.91% | 28.53% | 42.24% | 30.26% | 1955 |
| MAD-GAN-SWaT on CICDDoS | 35.98% | 97.56% | 35.98% | 46.35% | 1931 |
| MAD-GAN-DDoS on CICDDoS | 40.16% | 98.83% | 40.16% | 50.60% | 1931 |

Both MAD-GAN-DDoS and MAD-GAN-SWaT were evaluated with attack data from the SWaT and CICDDoS datasets only, as the two models were already trained on benign data. For each dataset, we collected 10,000 samples (rows) for evaluation. For each test case, there are three types of epochs used: sample-wise epoch, statistic-based epoch, and logits-based epoch [23]. The average results from the three epoch types for four test cases are reported in Table VI. Below, we compare both MAD-GAN-DDoS and MAD-GAN-SWaT performance on each dataset.

We observed that both MAD-GAN-SWaT and MAD-GAN-DDoS achieved near-perfect precision when tested on the CICDDoS dataset. On the same dataset, both accuracy and recall are the same regardless of the types of epochs, which never exceed 50%. As such, the F1 score is also low since it is the harmonic mean of precision and recall. This suggested that the number of false positives must be close to zero, in contrast to the higher number of false negative predictions. As a result, the models returned high precision but low accuracy. Since the accuracy and recall are also exactly the same, it is likely that the number of true negatives must also be approximating zero. In other words, this suggested that our test dataset for CICDDoS was imbalanced, with overwhelmingly more attack traffic than benign traffic. In such cases, a true positive result is more common than a true negative result. However, we must consider that the recall and accuracy are also not very high. This suggested that the number of false negatives must be greater than the number of true positives for both MAD-GAN-DDoS and MAD-GAN-SWaT, despite the former having a slightly better recall rate than the latter. In conclusion, MAD-GAN-DDoS and MAD-GAN-SWaT cannot detect attacks in the CICDDoS dataset very well.

When tested on the SWaT dataset, both models returned higher accuracy, but significantly lower precision when compared with previous testing results on the CICDDoS2019 dataset. The recall rates of MAD-GAN-SWaT and MAD-GAN-DDoS for all types of epochs do not exceed 50.35%. Due to the low precision, the F1 scores are also lower when testing MAD-GAN models on the SWaT datasets. Again, it is likely that the high number of false positives caused the low precision shown in Table VI. Interestingly, MAD-GAN-DDoS average performance on the SWaT dataset was on par with MAD-GAN-SWaT performance on the same dataset. Nevertheless, both MAD-GAN-DDoS and MAD-GAN-SWaT do not perform very well on the SWaT dataset.

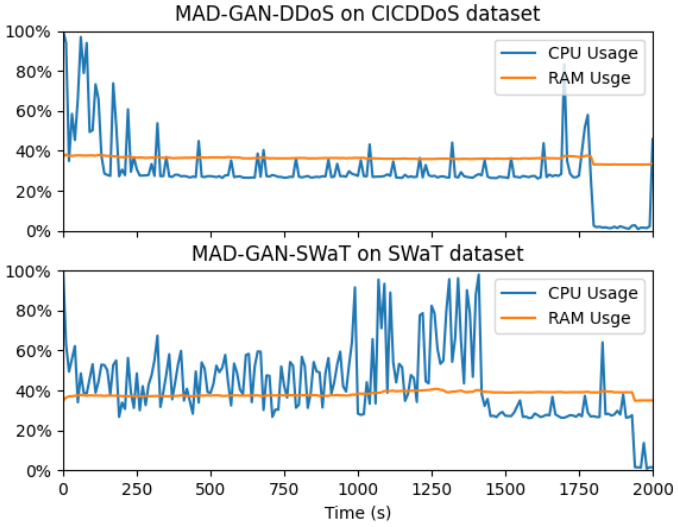


Fig. 4. MAD-GAN Models CPU & RAM Consumption

It is important to note that the MAD-GAN models take quite a long time to perform anomaly detection. As shown in Table VI, each model took at least 30 minutes to classify 10,000 samples. VI. Previous testing has shown that the more data that MAD-GAN models are given to detect at a time, the longer it will take the model to finish the classification task. This could be a concern when deploying MAD-GAN in a real-time detection environment.

We recorded the CPU and RAM usage for MAD-GAN-SWaT and MAD-GAN-DDoS when evaluated on the SWaT and CICDDoS datasets, in that order, respectively. Fig. 4 shows that that MAD-GAN-SWaT tend to reach high CPU usages at the start, which eventually reduces to approximately 40%. Meanwhile, MAD-GAN-DDoS has a higher CPU usage overall. The falls in CPU usage signifies that the two models have finished the classification task, where virtually no changes in RAM usage are observed. This suggested that MAD-GAN requires extensive CPU power to perform detection at a reasonable speed. However, MAD-GAN low RAM usage means that it is less likely to cause freezing or forceful shutdown of the machine it was deployed on. Nevertheless, MAD-GAN-DDoS high CPU usage shows that it could potentially interfere with crucial control components when deployed in a resource-constrained environment.

Overall, we have observed that the MAD-GAN model has potential for domain adaptation and transfer learning, considering that MAD-GAN-DDoS achieved comparable

TABLE VII
LUCID-DDOS TEST SAMPLES

| Samples | Packets | Time (s) |
|---|---|---|
| cicddos_benign_1 | 44998 | 605.00 |
| cicddos_benign_2 | 42500 | 647.65 |
| cicddos_benign_3 | 34215 | 605.72 |
| cicddos_benign_4 | 25132 | 378.67 |
| cicddos_attack_1 | 26389 | 371.32 |
| cicddos_attack_2 | 26535 | 369.33 |
| cicddos_attack_3 | 26093 | 361.80 |
| cicddos_attack_4 | 25906 | 359.36 |
| cicddos_attack_5 | 25943 | 355.65 |
| cicddos_attack_6 | 25922 | 364.65 |
| swat_benign_1 | 77746 | 1007.91 |
| swat_benign_2 | 78190 | 998.26 |
| swat_attack_1 | 78879 | 1,010.46 |
| swat_attack_2 | 77137 | 986.00 |

performance with MAD-GAN-SWaT when evaluated on the SWaT dataset. On another hand, MAD-GAN-DDoS still has better performance than MAD-GAN-SWaT when evaluated on the CICDDoS dataset. It should be noted that MAD-GAN scripts were designed to test the trained models on labelled data, which are usually not available in a real-time detection environment. Furthermore, MAD-GAN slow detection speed could be an obstacle when deploying it in a real-time detection environment, in addition to high false negative. The low detection performance given by the MAD-GAN-SWaT model could be a result of overfitting since the model was trained on duplicated data, as discussed in Sect. V. Future work is needed to improve MAD-GAN detection performance and capacity for live inference and domain adaptation.

### *B. LUCID-DDoS*

To determine LUCID-DDoS accuracy on unlabelled data, we evaluated LUCID-DDoS with fully benign and fully malicious data from both the SWaT and CICDDoS datasets. For each sample given to LUCID-DDoS, it will then classify the given data in multiple instances. There are 12 test samples used in total, with packets count and detection time for each shown in Table. VII. Overall, there are 146845 packets in CICDDoS benign sample, 156788 packets in CICDDoS attack sample, 155936 packets in SWaT benign sample, and 156016 packets in the SWaT attack sample.

LUCID-DDoS predicted DDoS rate on the CICDDoS datasets is shown in Table VIII. In this case, we assume that the actual DDoS rate of benign samples in the CICDDoS dataset is close to 0%, and the actual DDoS rate of attack samples in the same dataset is close to 100%. The results show that LUCID-DDoS detected a low DDoS rate in benign samples, mostly below 15% in each instance. There are a few exceptions, but all of these detection instances reported DDoS rates below 30%. Meanwhile, LUCID-DDoS reported near 100% DDoS rate in CICDDoS attack samples. This huge difference suggested that LUCID-DDoS can effectively discern between benign and malicious traffic in the CICDDoS dataset. As a result, we expect LUCID-DDoS to perform similarly in similar DDoS attacks when deployed in a real system.

TABLE VIII
LUCID-DDOS PREDICTED DDOS % ON CICDDOS DATASET

| CICDDoS (Benign) | DDoS% | CICDDoS (Attack) | DDoS% |
|---|---|---|---|
| cicddos_benign_1_ins_1 | 2.7% | cicddos_attack_1_ins_1 | 98.6% |
| cicddos_benign_1_ins_2 | 3.3% | cicddos_attack_1_ins_2 | 98.7% |
| cicddos_benign_1_ins_3 | 0.3% | cicddos_attack_1_ins_3 | 99.6% |
| cicddos_benign_1_ins_4 | 1.1% | cicddos_attack_2_ins_1 | 99.9% |
| cicddos_benign_1_ins_5 | 2.8% | cicddos_attack_2_ins_2 | 98.2% |
| cicddos_benign_2_ins_1 | 4.7% | cicddos_attack_2_ins_3 | 99.2% |
| cicddos_benign_2_ins_2 | 6.0% | cicddos_attack_3_ins_1 | 99.9% |
| cicddos_benign_2_ins_3 | 15.8% | cicddos_attack_3_ins_2 | 99.9% |
| cicddos_benign_2_ins_4 | 14.8% | cicddos_attack_3_ins_3 | 100.0% |
| cicddos_benign_2_ins_5 | 3.5% | cicddos_attack_4_ins_1 | 99.9% |
| cicddos_benign_2_ins_6 | 1.6% | cicddos_attack_4_ins_2 | 99.8% |
| cicddos_benign_3_ins_1 | 2.0% | cicddos_attack_4_ins_3 | 99.8% |
| cicddos_benign_3_ins_2 | 4.0% | cicddos_attack_5_ins_1 | 100.0% |
| cicddos_benign_3_ins_3 | 29.2% | cicddos_attack_5_ins_2 | 100.0% |
| cicddos_benign_3_ins_4 | 18.8% | cicddos_attack_5_ins_3 | 100.0% |
| cicddos_benign_3_ins_5 | 4.5% | cicddos_attack_6_ins_1 | 100.0% |
| cicddos_benign_4_ins_1 | 11.4% | cicddos_attack_6_ins_2 | 100.0% |
| cicddos_benign_4_ins_2 | 18.8% | cicddos_attack_6_ins_3 | 99.9% |
| cicddos_benign_4_ins_3 | 7.7% | | |

The LUCID DDoS percentage for two benign samples and two attack samples from the SWaT dataset, each classified into nine instances, is shown in the Table IX. Although we expected the real DDoS rate for benign samples to be close to 0%, we observed a predicted DDoS rate ranging between 18% to 33%, with two other instances approximating 90% DDoS rate. Likewise, we expected the DDoS rate for most instances in the two SWaT attack samples to be close to 100%, but LUCID-DDoS predictions returned DDoS rates ranging between 11% and 43% for most detection instances. Comparing the results for benign samples and attack samples also shows no particular difference, considering that the two ranges are similar. Therefore, it is very likely that LUCID-DDoS was not able to distinguish between benign and attack samples in the SWaT dataset.

In other words, LUCID-DDoS has limited applicability in detecting anomalies in vastly different attack types than those on which the model was trained.As a matter of fact, the LUCID model was tightly implemented around the TCP/IP protocol, as seen in the 11 features extracted by LUCID in [20]. These features are vastly irrelevant to application-layer protocols, such as the Common Industrial Protocol used in the SWaT testbed. Naturally, the packet patterns that LUCID learned would not be very relevant for anomaly detection in any other industrial protocols that are unfamiliar to LUCID, including those present in the SWaT dataset. Given that LUCID was not able to train on the SWaT dataset as iterated in Sect. V, further improvements on LUCID are needed to adapt it to a wider range of attacks and protocols.

Fig. 5 shows the CPU and RAM consumption in four samples from four different test cases on the Raspberry Pi. It can be seen that LUCID has a high RAM usage demand for all four test cases, while its CPU usage remains mostly stable at 30% except for some initial spikes. From Table VII, we also notice that samples with higher number of packets tend to have a higher RAM consumption peak. This suggests that while LUCID-DDoS has a very low CPU usage, it should only perform anomaly detection on a low number of packets

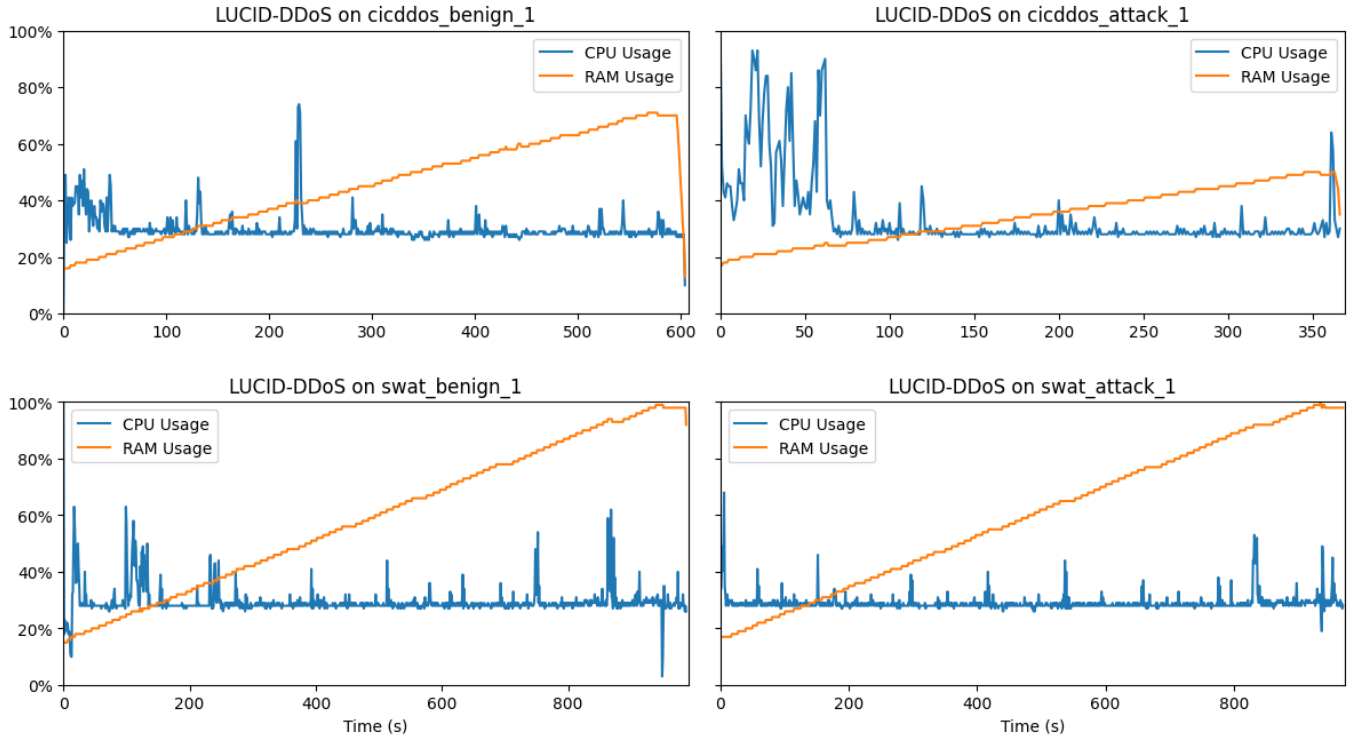


Fig. 5. LUCID-DDoS CPU & RAM Consumption

TABLE IX
LUCID-DDoS PREDICTED DDoS % ON SWAT DATASET

| SWaT (Benign) | DDoS% | SWaT (Attack) | DDoS% |
|---|---|---|---|
| swat_benign_1_ins_1 | 29.4% | swat_attack_1_ins_1 | 79.2% |
| swat_benign_1_ins_2 | 89.3% | swat_attack_1_ins_2 | 78.2% |
| swat_benign_1_ins_3 | 29.9% | swat_attack_1_ins_3 | 15.6% |
| swat_benign_1_ins_4 | 23.6% | swat_attack_1_ins_4 | 21.2% |
| swat_benign_1_ins_5 | 27.8% | swat_attack_1_ins_5 | 27.3% |
| swat_benign_1_ins_6 | 28.5% | swat_attack_1_ins_6 | 13.2% |
| swat_benign_1_ins_7 | 18.0% | swat_attack_1_ins_7 | 43.4% |
| swat_benign_1_ins_8 | 31.3% | swat_attack_1_ins_8 | 11.6% |
| swat_benign_1_ins_9 | 21.2% | swat_attack_1_ins_9 | 14.4% |
| swat_benign_2_ins_1 | 34.6% | swat_attack_2_ins_1 | 19.3% |
| swat_benign_2_ins_2 | 29.5% | swat_attack_2_ins_2 | 18.9% |
| swat_benign_2_ins_3 | 23.4% | swat_attack_2_ins_3 | 38.1% |
| swat_benign_2_ins_4 | 18.1% | swat_attack_2_ins_4 | 12.9% |
| swat_benign_2_ins_5 | 31.4% | swat_attack_2_ins_5 | 29.3% |
| swat_benign_2_ins_6 | 29.6% | swat_attack_2_ins_6 | 25.3% |
| swat_benign_2_ins_7 | 88.7% | swat_attack_2_ins_7 | 84.2% |
| swat_benign_2_ins_8 | 32.6% | swat_attack_2_ins_8 | 29.7% |
| swat_benign_2_ins_9 | 21.2% | swat_attack_2_ins_9 | 19.7% |

at a time to avoid maxing out the available RAM. In such case, the machine it was deployed on could freeze and force a shutdown to resume operations. In ICS and CPS where availability requirements are high, such situations are not acceptable. Further work is needed to address LUCID-DDoS high RAM requirements.

To summarize, LUCID-DDoS is effective at detecting attacks that are strictly based on the TCP/IP protocol. While it was not able to classify attacks on the application layers, which include most industrial protocols as seen with the SWaT dataset, LUCID-DDoS has possible applications for detecting (D)DoS attacks in the IoT domain.

### C. Discussion & Threats to Validity

Our experimental results have proven the proposed hypothesis, as shown by the fact that we were able to perform domain adaptation on MAD-GAN, which satisfied the following two conditions: 1) The model was able to extract relevant features for training, and 2) the model provides sufficient generalisation that allows flexibility to adapt MAD-GAN to a different detection domain. As for LUCID, it was not able to extract the relevant features from the SWaT dataset and thus was not able to train on that dataset. This was a problem of architectural design rather than a technical difficulty.

LUCID-DDoS is shown to have a significantly better performance than both MAD-GAN-SWaT and MAD-GAN-DDoS when evaluated on the CICDDoS dataset. While the MAD-GAN-SWaT did not achieve a good detection performance on the SWaT dataset, we still consider it to be more effective than the LUCID-DDoS performance on the same dataset. This is because LUCID-DDoS was not able to distinguish between benign and malicious traffic in the SWaT dataset at all with the data patterns it learned from training on the CICDDoS dataset, which means that there is no basis to compare LUCID-DDoS against MAD-GAN-SWaT. There are no observable changes to the three detection models' computational overhead when used to detect attacks in a different dataset. Despite using the same number of samples, we can see that MAD-GAN-SWaT has higher CPU usage than MAD-GAN-DDoS.

Furthermore, our experiments have confirmed the fact that a model does not necessarily perform well on other datasets

aside from the one it was trained on. Particularly, we highlight the following points of consideration with regard to domain adaptation and transfer learning in cybersecurity for CPS and IoT, as well as existing limitations:

- **Ability to Learn from Relevant Features in Training Dataset**. Our experiments have shown that a model's design plays an important role in what data it can learn from. For example, the architectural design of LUCID only allows it to learn characteristics pertaining to TCP/IP packets, and as a result, it was not able to learn any other features in different communication protocols. Furthermore, LUCID's lack of consideration for packet payloads is an obstacle for attack detection on application layer protocols. Meanwhile, MAD-GAN's architecture allows it to learn from any given characteristic in any protocol and utilise that for anomaly detection. In other words, the model's design must enable it to learn transferrable knowledge from the provided data in order to detect DDoS attacks in different domains. Especially in the case of LUCID-SWaT, the model failed mainly because it would not be able to learn meaningful patterns from the SWaT dataset, and as such, any detection results it might give from these patterns would be of no use.
- **Consideration for Real-time Detection**.According to [21], the LUCID model was designed for real-time detection, allowing it to collect packets as.pcap files while also performing detection on them. Meanwhile, the MAD-GAN model can only detect on existing labelled data in the .csv formats. While this feature allows us to accurately assess MAD-GAN performance in an experimental setting, it is not suited for real-time detection where MAD-GAN is forced to perform detection on unlabelled data. Furthermore, MAD-GAN takes a very long time to perform anomaly detection on a small number of samples. Therefore, the MAD-GAN model will need an improvement in detection speed before being deployed in a real-time, resource constrained detection environment where a stronger processor might not be available.
- **Computational Overhead in Resource-Constrained Environment.** It is important to maintain a low computational overhead when performing anomaly detection in resource-constrained environment to avoid interruptions to concurrently running processes. The LUCID model has a very high RAM consumption when performing the classification task on pre-recorded .pcap files, which makes it impractical for deployment in such an environment. On another hand, MAD-GAN model does not have a very high RAM requirement, although it does have high CPU usage at times and can delay certain concurrently running tasks. In other worlds, MAD-GAN has low computational overhead that makes it suit for deployment in resource-constrained environment. However, MAD-GAN slow detection speed is still in need of improvements, as iterated previously.
- **Detection Accuracies**. We have observed that LUCID-DDoS performs very well on the attacks it was trained on but is otherwise unable to discern between benign and malicious network packets from the SWaT dataset. However, both MAD-GAN-DDoS and MAD-GAN-SWaT models achieved poor results on both CICDDoS and SWaT datasets in all four evaluation instances. This means that the knowledge that LUCID and MAD-GAN learned cannot be sufficiently transferred to a different detection domain to achieve an acceptable detection performance. However, it is possible that MAD-GAN-SWaT was overfit due to training on duplicated data from the SWaT 2019 dataset. Although in such cases, we would expect a higher detection performance when testing MAD-GAN-SWaT on the SWaT dataset.

There are also various limitations, or threats to the validity of the experiment, that must be addressed. Future work to address these limitations is mentioned briefly in Sect. VIII.

- **Differences in Training Data in LUCID and MAD-GAN models.** When training LUCID and MAD-GAN, we intended to do so based on what each model was originally designed for. As such, there are some difference in the training data that we must also mention. As a matter of fact, we were only able to attempt training LUCID on the SWaT dataset with network packages recorded during the attacks, as that is what LUCID was designed for. Meanwhile, the MAD-GAN model was trained on the physical characteristics of the SWaT testbed during normal operations rather than network packages in .pcap format. In fact, [22] used this training method in their paper. Likewise, we used the same training method used by the authors to accurately assess each model's potential at the time of publication, with the only difference being we use newer versions of the respective datasets. Data availability in .pcap and .csv formats also mean that each model was trained on a slightly different amount of data.
- **Lack of Testing Comparison with State-of-the-art.** In our experiment, we have only evaluated LUCID and MAD-GAN performance and not other state-of-the-art models. This is largely due to the fact that we were not able to access the source code of these models.

## VIII. Conclusion & Future Works

In conclusion, our experiments have confirmed our hypothesis on domain adaptation for (D)DoS attack detection models for the IoT and CPS. In domain adaptation, it is important that the model in question is able to extract and learn from relevant features in given datasets to train on, in addition to having sufficient flexibility in its code. We are also able to draw the following conclusions regarding domain adaptation within the context of detection in a resource-constrained environment, particularly:

- Domain adaptation is heavily reliant on the model's ability to learn from relevant features and whether the patterns learned are transferrable to a different detection domain. This factor plays a crucial role in the model's detection performance.
- Detecting attacks from an entirely different detection domain tends to not return high detection results, as the

learned data patterns may not transfer very well into the aforementioned domain. However, there are possible common points in (D)DoS attacks in the two domains of IoT and CPS that detection models could train on and use for domain adaptation.
- We have observed that domain adaptation on detection models does not have a large effect on a model's computational overhead, and that each model's computing resource consumption seems to largely depend on the model architecture itself. Nevertheless, these factors should also be given due consideration when deployed in a resource-constrained environment.

### A. Future Works

There are various limitations in our experiments that could be improved upon in our future works, including but not limited to:

- Utilize a realistic, resource-constrained testbed to evaluate MAD-GAN and LUCID. We will then conduct attacks on the system and evaluate both models' performance and computational overhead in real-time detection settings instead of using public datasets.
- Implement real-time detection on unlabelled data modules for the MAD-GAN model. We also consider how to address the non-converging problem that MAD-GAN encountered when using both a discriminator and a generator to detect anomalies, as discussed in Sect. V. It is possible that by using both the discriminator and generator, we will be able to obtain better detection results. MAD-GAN slow detection speed is also in need of improvement for detection efficiency.
- Improve LUCID's capacities for feature extraction, including the model's consideration for packet payloads, such that it is applicable for detecting attacks in different protocols. This will improve its capacity for domain adaptation.
- Address the high computational overhead in LUCID for detection in pre-recorded .pcap files and real-time detection. The latter is only possible if we are able to develop a realistic resource-constrained testbed. We will seek to observe the effect that LUCID's computational requirements have on the testbed and evaluate whether it is suitable for deployment in such an environment.
- Additional evaluation of different detection models on other datasets to confirm the validity of our hypothesis.

## REFERENCES


[1] E. A. Lee and S. A. Seshia, "Introduction to embedded systems: A cyber-physical systems approach," Dec 2016. [Online]. Available: https://mitpress.mit.edu/9780262533812/introduction-to-embedded-systems/

[2] S. K. Khaitan and J. D. McCalley, "Design techniques and applications of cyberphysical systems: A survey," *IEEE Systems Journal*, vol. 9, no. 2, pp. 350–365, 2015.

[3] D. Kiran, "Chapter 35 - internet of things," in *Production Planning and Control*, D. Kiran, Ed. Butterworth-Heinemann, 2019, pp. 495–513. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780128183649000354

[4] R. F. Ibrahim, Q. Abu Al-Haija, and A. Ahmad, "Ddos attack prevention for internet of thing devices using ethereum blockchain technology," *Sensors*, vol. 22, no. 18, p. 6806, 2022.

[5] F. Zahid, G. Funchal, V. Melo, M. M. Kuo, P. Leitao, and R. Sinha, "DDoS Attacks on Smart Manufacturing Systems: A Cross-Domain Taxonomy and Attack Vectors," in *2022 IEEE 20th International Conference on Industrial Informatics, INDIN'22*. IEEE, July 2022, pp. 1–6.

[6] V. Ngo, F. Zahid, M. Mohaghegh, and R. Sinha, "A systematic mapping of datasets and machine learning models for (d)dos detection in industrial control systems," 2022.

[7] C. Mujeeb Ahmed, M. A. Umer, B. S. S. Binte Liyakkathali, M. T. Jilani, and J. Zhou, *Machine Learning for CPS Security: Applications, Challenges and Recommendations*. Cham: Springer International Publishing, 2021, pp. 397–421. [Online]. Available: https://doi.org/10.1007/978-3-030-57024-8_18

[8] B. A. Tama, S. Y. Lee, and S. Lee, "A systematic mapping study and empirical comparison of data-driven intrusion detection techniques in industrial control networks," *Archives of Computational Methods in Engineering*, 5 2022.

[9] M. M. Salim, S. Rathore, and J. H. Park, "Distributed denial of service attacks and its defenses in iot: a survey," *Journal of Supercomputing*, vol. 76, pp. 5320–5363, 2020. [Online]. Available: https://doi.org/10.1007/s11227-019-02945-z

[10] M. R. Babu and K. N. Veena, "A survey on attack detection methods for iot using machine learning and deep learning," in *2021 3rd International Conference on Signal Processing and Communication (ICPSC)*, 2021, pp. 625–630.

[11] R. Ahmad and I. Alsmadi, "Machine learning approaches to iot security: A systematic literature review," *Internet of Things*, vol. 14, p. 100365, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2542660521000093

[12] A. Gangopadhyay, I. Odebode, and Y. Yesha, "A domain adaptation technique for deep learning in cybersecurity," vol. 11878 LNCS. Springer, 2020, pp. 221–228.

[13] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 10, pp. 1345–1359, 2010.

[14] S. De, M. Bermudez-Edo, H. Xu, and Z. Cai, "Deep generative models in the industrial internet of things: A survey," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 9, pp. 5728–5737, 2022.

[15] M. I. Tariq, N. A. Memon, S. Ahmed, S. Tayyaba, M. T. Mushtaq, N. A. Mian, M. Imran, and M. W. Ashraf, "A review of deep learning security and privacy defensive techniques," *Mobile Information Systems*, vol. 2020, p. 1–18, 2020.

[16] C. R. Kothari, *Research Methodology: Methods and Techniques*. New Age International, 2004.

[17] S. Easterbrook, J. Singer, M.-A. Storey, and D. Damian, "Selecting empirical methods for software engineering research," 2008.

[18] A. R. Hevner, S. T. March, J. Park, and S. Ram, "Design science in information systems research," pp. 75–105, 2004. [Online]. Available: https://www.jstor.org/stable/25148625

[19] V. Ngo, M. Mohaghegh, and R. Sinha, "Domain-Adaptation-of-MAD-GAN-and-LUCID-for-Cyberattack-Detection," 11 2022. [Online]. Available: https://github.com/vickyngo-code/Domain-Adaptation-of-MAD-GAN-and-LUCID-for-Cyberattack-Detection

[20] R. Doriguzzi-Corin, S. Millar, S. Scott-Hayward, J. Martinez-Del-Rincon, and D. Siracusa, "Lucid: A practical, lightweight deep learning solution for ddos attack detection," *IEEE Transactions on Network and Service Management*, vol. 17, pp. 876–889, 2020, context is edge computing, is it applicable for other domains?¡br/¿¡br/¿¡br/¿¡br/¿¡br/¿Look into DeepDefense LSTM (4), TR-IDS (36), E3ML (47).

[21] R. Doriguzzi-Corin, S. Millar, S. Scott-Hayward, J. Martínez-del Rincón, and D. Siracusa, "Lucid: A practical, lightweight deep learning solution for ddos attack detection," Jun 2022. [Online]. Available: https://github.com/doriguzzi/lucid-ddos

[22] D. Li, D. Chen, B. Jin, L. Shi, J. Goh, and S.-K. Ng, "Mad-gan: Multivariate anomaly detection for time series data with generative adversarial networks," in *Artificial Neural Networks and Machine Learning – ICANN 2019: Text and Time Series: 28th International Conference on Artificial Neural Networks, Munich, Germany, September 17–19, 2019, Proceedings, Part IV*. Berlin, Heidelberg: Springer-Verlag, 2019, p. 703–716. [Online]. Available: https://doi.org/10.1007/978-3-030-30490-4_56

[23] D. Li, S.-K. Ng., D. Chen, and J. Goh, "Multivariate anomaly detection for time series data with gans," Jan 2019. [Online]. Available: https://github.com/LiDan456/MAD-GANs

[24] "Ddos 2019 — datasets — research — canadian institute for cybersecurity — unb." [Online]. Available: https://www.unb.ca/cic/datasets/ddos-2019.html

[25] I. Sharafaldin, A. H. Lashkari, S. Hakak, and A. A. Ghorbani, "Developing realistic distributed denial of service (ddos) attack dataset and taxonomy," in *2019 International Carnahan Conference on Security Technology (ICCST)*, 2019, pp. 1–8.
[26] "itrust labs dataset info," May 2022. [Online]. Available: https://itrust.sutd.edu.sg/itrust-labs_datasets/dataset_info/
[27] J. Goh, S. Adepu, K. N. Junejo, and A. Mathur, "A dataset to support research in the design of secure water treatment systems," in *Critical Information Infrastructures Security*, G. Havarneanu, R. Setola, H. Nassopoulos, and S. Wolthusen, Eds. Cham: Springer International Publishing, 2017, pp. 88–99.
[28] M. A. Ferrag, L. Shu, H. Djallel, and K.-K. R. Choo, "Deep learning-based intrusion detection for distributed denial of service attack in agriculture 4.0," *Electronics*, vol. 10, no. 11, 2021. [Online]. Available: https://www.mdpi.com/2079-9292/10/11/1257