Cyberattack Detection in Industrial Control Systems via Supervised Deep Neural Networks: A Case Study of the Secure Water Treatment Testbed

Amr Alfayoumy
Electronic and Computer Engineering Department
Nile University
Giza, Egypt
a.alfayoumy@nu.edu.eg

***Abstract*—This study investigates the application of supervised deep learning techniques to enhance cybersecurity in industrial control systems (ICS). Utilizing the Secure Water Treatment (SWaT) A1 & A2-Dec 2015 dataset, which comprises 11 days of operational data from a scaled-down water treatment plant, we developed and evaluated various deep learning models for both binary and multi-label classification of cyberattacks. For both the binary and multilabel classification tasks, we employed Long Short-Term Memory (LSTM) networks, Convolutional Neural Networks (CNNs), and hybrid CNN-LSTM models, to classify system states into normal operations and attack conditions. We further extended our analysis to identify specific attack points within the SWaT infrastructure through multi-label classification techniques also using LSTM, CNN, CNN-LSTM, as well as Multi-Layer Perceptrons (MLPs). Results demonstrate the high efficacy of these deep learning approaches in detecting and classifying cyberattacks. In binary classification, all models achieved macro F1 scores above 98.9%, with the CNN-LSTM hybrid model outperforming others, achieving a macro F1 score of 99.3%. For multi-label classification, individual models showed strong performance with macro F1 scores ranging from 98.0% to 99.5%. A majority voting ensemble method further improved performance, achieving a macro F1 score of 99.6%. This research contributes to the field by providing a comprehensive approach to cybersecurity in critical infrastructure systems. The high accuracy in both detecting attacks and identifying specific attack points offers promising avenues for enhancing the resilience and security of water treatment facilities.**

Keywords*—Industrial Control Systems (ICS), Secure Water Treatment (SWaT), Cyber-Physical Systems (CPS), cybersecurity, critical infrastructure protection, binary classification, multi-label classification, deep learning, LSTM, CNN, CNN-LSTM, MLP, ensemble methods.

I. Introduction

In recent years, the protection of critical industrial control systems (ICS), such as secure water treatment plants (SWaT), from cyber threats has become increasingly paramount. These systems play a vital role in ensuring public health and safety by providing clean and potable water to communities. However, with the advancement of technology, these systems have become susceptible to cyberattacks, posing significant risks to their operation and integrity.

The SWaT 2015 dataset [1] is a comprehensive repository of data collected from a secure water treatment plant, which is leveraged in this research to explore the effectiveness of multi-label classification in detecting and classifying cyberattacks. The dataset includes 51 features, such as sensor readings and control parameters, as well as 26 labeled attack points, providing valuable insights into the operation of the SWaT system and enabling the identification of anomalous behavior indicative of cyber threats.

Accurately detecting and classifying abnormal activities in SWaT systems is crucial for safeguarding against cyber threats. Binary classification, which distinguishes between normal operational behavior and cyberattacks, enables real-time identification of potential attacks, allowing operators to swiftly respond and mitigate the impact. Additionally, multi-label classification, a machine learning approach that can handle instances with multiple labels simultaneously, emerges as a promising solution for detecting and classifying various types of attacks on critical industrial control systems like secure water treatment plants.

The research follows a systematic approach to address the cybersecurity challenges in secure water treatment (SWaT) systems leveraging deep learning models, including Long Short-Term Memory (LSTM), Convolutional Neural Network (CNN), and their hybrid counterparts. Initially, the focus is on classifying records into normal operational states and abnormal states induced by cyberattacks. This binary classification task serves as the foundation for identifying instances of malicious activities within the SWaT systems. By accurately distinguishing between normal and attack states, the paper aims to establish a baseline understanding of system behavior and detect deviations indicative of potential security breaches. Following the binary classification, the research extends its scope to multilabel classification, where the goal is to classify the detected attacks into specific attack points or categories. This phase aims to identify and categorize various attack points within the SWaT systems. By employing multilabel classification techniques, the paper aims to provide a more granular and nuanced understanding of the cyber threats targeting SWaT infrastructure.

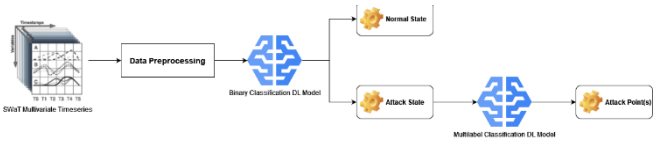

Fig. 1. Overview of the proposed solution.

The motivation behind this dual classification approach is twofold. Firstly, by first identifying whether an abnormality is present in the system, the approach ensures that resources are focused solely on instances requiring further investigation, thus optimizing detection efficiency. Secondly, the multilabel classification enables a comprehensive analysis of detected attacks, allowing for the identification of multiple attack points within a single instance. This holistic approach not only enhances the accuracy of attack detection but also facilitates a deeper understanding of the attack vectors and strategies employed by adversaries.

This research aims to develop robust and efficient binary classification techniques to detect cyberattacks on secure water treatment systems. Early attack detection can prevent disruptions, reduce contamination risks, and safeguard public health [2]. The study also explores multi-label classification to identify multiple attack points, addressing the need for effective cybersecurity against sophisticated threats targeting critical industrial control systems like SWaT. By leveraging deep learning models, the research seeks to accurately detect and distinguish cyberattacks from normal system behavior. The outcomes have implications for improving the resilience, security, and reliability of SWaT systems, protecting public health and safety. The study explores advanced techniques like multi-label classification to enhance the capability of SWaT testbed operators and cybersecurity professionals in detecting, mitigating, and responding to cyber threats effectively. The findings have broader implications for safeguarding critical infrastructure and public well-being.

This research fills a gap in the literature by applying supervised learning algorithms to the SWaT dataset to classify records into normal and attack states, and further identify the specific attack point(s) targeted by the detected attacks. While prior studies have explored anomaly detection to distinguish between normal and abnormal system behavior, this research takes a comprehensive approach by incorporating binary and multi-label classification to pinpoint the exact attack points within the SWaT infrastructure. By leveraging advanced deep learning models, including Long Short-Term Memory (LSTM) and Convolutional Neural Network (CNN), the paper aims to develop robust and efficient techniques that can accurately detect and classify cyberattacks on secure water treatment systems. This holistic approach not only enhances the accuracy of attack detection but also facilitates a deeper understanding of the attack vectors and strategies employed by adversaries, enabling more effective countermeasures and improved resilience of critical industrial control systems.

II. Literature Review

Industrial Control Systems (ICS) play a critical role in managing and monitoring various industrial processes, including manufacturing, power generation, water treatment, and more. As these systems become increasingly connected and digitized, they also become more vulnerable to cyber-attacks and anomalies that can disrupt operations and potentially cause significant damage. Anomaly detection in ICS has thus emerged as a crucial area of research, with numerous approaches being developed to identify and mitigate potential threats. This literature review aims to provide a comprehensive overview of recent advancements in anomaly detection for Industrial Control Systems. We will examine various methodologies, including traditional machine learning approaches, deep learning techniques, and hybrid models. Additionally, we will explore the challenges faced in implementing these solutions and discuss future research directions.

Industrial Control Systems are critical infrastructure components that manage and control industrial processes. Any disruption or anomaly in these systems can lead to severe consequences, including production losses, equipment damage, and even threats to public safety. Wen (2023) addresses the insufficient awareness among industrial practitioners regarding cyberattacks on industrial control systems (ICS) [3]. The authors apply this method to a continuous stirred tank reactor (CSTR) model to identify vulnerable locations and components within the ICS. Results indicate that physical system levels of ICS are highly vulnerable to cyberattacks, with controllers, workstations, and human-machine interfaces identified as critical components in both cyberattack scenarios and defensive strategies.

Detecting anomalies in ICS presents unique challenges due to the complex nature of these systems. Tushkanova et al. (2023) address the challenge of detecting cyberattacks early in cyber-physical systems (CPS), given the severe consequences such attacks can entail [4]. The authors emphasize the difficulty of detecting previously unknown attacks and propose using system behavior analysis methods along with unsupervised or semi-supervised machine learning techniques to tackle this issue. Methods discussed include reviewing existing approaches to attack and anomaly detection in CPS, focusing particularly on datasets and evaluation metrics used to assess the effectiveness of these approaches. Results from the analysis highlight that while attempts have been made to create datasets for training machine learning models, the completeness and validity of these datasets remain questionable due to security constraints on real-world CPS data. The paper identifies only a limited number of suitable datasets, primarily those generated using real test beds and containing both network and sensor data, as preferable for intrusion detection tasks.

Ahmed et al. (2020) highlight the increasing adoption of data-centric approaches in developing defense mechanisms for critical infrastructure like the electric power grid and water treatment plants [5]. These approaches often utilize established machine learning and system identification methods such as Multi-Layer Perceptrons, Convolutional Neural Networks, and Deep Auto Encoders to construct anomaly detectors for process monitoring. Methods discussed include the creation of anomaly detectors using machine learning and system identification techniques, typically evaluated using data either from operational plants or simulators. The authors note a scarcity of real-time assessments conducted on live plants. Results indicate that despite advancements in creating anomaly detectors and evaluating their performance, significant challenges remain. These challenges are particularly crucial to address before deploying such detectors confidently in large-scale settings like city-scale plants or expansive electric power grids.

Several studies have explored the use of traditional machine learning techniques for anomaly detection in ICS.

Inoue et al. (2017) proposed and evaluated the application of unsupervised machine learning techniques for anomaly detection in Cyber-Physical Systems (CPS) [6]. Specifically, the authors compare two methods: Deep Neural Networks (DNN) adapted to time series data from a CPS and one-class Support Vector Machines (SVM). Methods used involve training anomaly detectors using logs generated by the Secure Water Treatment (SWaT) testbed, which operates under normal conditions. The performance of these methods is then evaluated using logs generated by SWaT under 36 different attack scenarios. Results show that the DNN method produces fewer false positives compared to the one-class SVM, while the SVM detects slightly more anomalies. Overall, the DNN achieves a slightly better F measure than the SVM.

Yau et al. (2020) investigated the use of one-class support vector machines (OC-SVM) for detecting attacks on water treatment systems [7]. The aim of this paper is to improve attack detection and forensic investigations for critical infrastructure like power grids and water treatment plants using machine learning, specifically one-class Support Vector Machines (SVM). Methods involve using a water treatment testbed to collect data under normal and attack conditions. The water treatment process is divided into sub-processes, and individual one-class SVM models are trained for each sub-process to improve detection accuracy. Results show that sub-process models have better detection performance than a single model for the complete process, enhancing both detection accuracy and forensic investigation efficiency.

Deep learning has gained significant attention in recent years for its ability to automatically learn complex patterns from large datasets. Abdelaty et al. (2021) introduce DAICS, a deep learning framework for detecting cyber-attacks on Industrial Control Systems (ICSs) that adapts to evolving normal behaviors [8]. Methods include a 2-branch neural network that learns changes with minimal data and automatic tuning of detection thresholds, requiring no specialized human intervention. Results from evaluations using publicly available datasets indicate that DAICS achieves higher detection rates and accuracy compared to state-of-the-art approaches and demonstrates increased robustness to additive noise.

Feng et al. (2017) propose an anomaly detection method for industrial control systems (ICS) that integrates network package content analysis and time-series structure analysis [9]. Methods involve creating a baseline signature database of normal communication patterns between ICS field devices, stored using a Bloom filter for package content anomaly detection. Additionally, a stacked Long Short Term Memory (LSTM) network-based SoftMax classifier is used for time-series anomaly detection by predicting likely package signatures based on prior traffic. Results demonstrate that this combined approach achieves higher performance in anomaly detection compared to current state-of-the-art techniques, validated using a real dataset from a gas pipeline SCADA system.

Kravchik and Shabtai (2018) explored the use of Convolutional Neural Networks (CNNs) for detecting cyber attacks in ICS, specifically on the Secure Water Treatment (SWaT) testbed dataset. [10]. Methods involve anomaly detection by measuring the statistical deviation between predicted and observed values. Various deep neural network architectures, including convolutional and recurrent networks, were employed. Results show that the proposed method detected 31 out of 36 attacks with three false positives, improving upon previous research. The study finds that 1D convolutional networks are effective for anomaly detection and time series prediction in ICS, outperforming recurrent networks. Additionally, 1D convolutional networks are simpler, smaller, and faster than recurrent neural networks.

Researchers have also investigated hybrid and ensemble models that combine multiple techniques to improve detection performance. Sapkota et al. (2020) introduced FALCON, a framework for anomaly detection in ICS that utilizes an ensemble of machine learning algorithms [11]. Methods include using dilated convolution and long short-term memory (LSTM) layers to capture temporal and long-term dependencies in sensor and actuator data. The data undergo a feature engineering process where wavelet transformation is applied to extract relevant features for the model. The study explores four variations of supervised deep learning models and an unsupervised support vector machine (SVM) model. Results demonstrate that the proposed framework, validated on the SWaT testbed, detects more attacks in a shorter period compared to previously published methods.

Lee et al. (2023) proposed an anomaly detection method using an ensemble of multi-point LSTMs tailored for diverse time-series domains like Cyber-Intrusion Detection, Fraud Detection, and Industrial Anomaly Detection [12]. Key features include automated model selection within a specified range, stacking ensemble techniques to combine outputs from multiple LSTMs, and final anomaly detection using the aggregated output vector. Experiments on three real-world datasets show that the proposed model achieves high accuracy, efficient execution times, and favorable F1 scores, despite the longer training time required for the LSTM ensemble.

As the complexity of anomaly detection models increases, there is a growing need for explainable AI techniques to interpret model decisions. Hoang et al. (2022) explored enhancing anomaly detection in Industrial Control Systems (ICS) using Explainable Artificial Intelligence (XAI) to improve the interpretability of AI-based methods, specifically focusing on an LSTM-based Autoencoder-OCSVM model [13]. Methods involve applying an LSTM-based Autoencoder combined with a One-Class Support Vector Machine (OCSVM) for anomaly detection, with XAI techniques to make the model's decisions more interpretable. The method is demonstrated using a well-known SCADA dataset. Results show that the proposed method not only achieves high detection performance but also provides more explainable and reliable results compared to traditional "black box" AI models.

Recent research has explored the use of graph-based techniques for anomaly detection in multivariate time series data. Zhang et al. (2022) proposed a novel Graph Relational Learning Network (GReLeN) to detect anomalies in multivariate time series data by learning the between-sensor dependence relationships [14]. The approach uses a Variational AutoEncoder (VAE) for feature extraction and system representation, combined with a Graph Neural Network (GNN) and a stochastic graph relational learning strategy to capture sensor dependencies. A composite anomaly metric is then established based on the learned dependence structure. Results from experiments on four real-world datasets demonstrate the superiority of the proposed method in terms of detection accuracy, anomaly diagnosis, and model interpretation.

Koutroulis et al. (2022) proposed a causality-inspired approach for anomaly detection in a water treatment testbed [15]. The proposed approach is an unsupervised anomaly detection method that leverages causal inference in two phases: Causal models are used to identify crucial interdependencies within the system, requiring minimal domain knowledge, and univariate models learn the normal behavior of individual system components. In the final phase, the extreme studentized deviate (ESD) is applied to the computed anomaly score to detect attacks and filter out irrelevant sensor signals. Validation on the Secure Water Treatment (SWaT) benchmark demonstrates that the approach achieves the highest F1 score with zero false alarms, highlighting its suitability for real-world deployment where accuracy and reliability are paramount.

Abid et al. (2023) investigated the use of distributed deep learning and transfer learning techniques for intrusion detection in ICS [16]. The approach aims to improve efficiency and response times by deploying various Machine Learning and Deep Learning algorithms, such as convolutional neural networks (CNN), tested using the SWaT industrial dataset for performance evaluation. Traditional protections are deemed insufficient against rapid, sophisticated attacks targeting production lines, highlighting the need for advanced security measures in modern manufacturing settings.

Several studies have focused on anomaly detection in water treatment systems, which are critical infrastructure components. Adepu and Mathur (2016) conducted an investigation to assess the impact of single-point cyber attacks on the Secure Water Treatment (SWaT) system [17]. These attacks targeted the SCADA server, which communicates with Programmable Logic Controllers (PLCs), sensors, and actuators. The study employed a novel attacker model to simulate various attack scenarios, focusing on two key findings: understanding how attacks spread across the system in terms of affected components, and observing the response of the water treatment process within SWaT to these attacks. Based on these observations, the study proposes attack detection mechanisms that leverage physical properties monitored during the treatment process, aiming to enhance the system's ability to detect and mitigate cyber threats effectively.

Boateng et al. (2022) propose an unsupervised learning approach for anomaly detection in ICS using neural networks [18]. This approach integrates a one-class objective function with a novel regularization term tailored to industrial requirements and performance metrics like precision or recall. Evaluation on the Secure Water Treatment (SWaT) dataset demonstrates the method's scalability and effectiveness in detecting attacks compared to existing approaches. Notably, the regularization-enhanced model achieves superior recall in 15 out of 36 attack scenarios, highlighting its robustness in real-world ICS environments. Qualitative analysis on SWaT's security showdown event data further validates the technique's proficiency in detecting real-time injected attacks.

Yang et al. (2024) proposed ADT, a time series anomaly detection method for cyber-physical systems using deep reinforcement learning [19]. ADT is an agent-based dynamic thresholding method using deep reinforcement learning (DQN) for anomaly detection. ADT optimizes thresholds based on anomaly scores from autoencoders and environmental cues, ensuring real-time adaptability in CPS time series data. Evaluated on SWaT, WADI, and HAI datasets, ADT achieves F1 scores between 0.995 and 0.999, outperforming benchmarks. It proves effective in noisy, delayed, or partial feedback scenarios and doubles as a dynamic thresholding controller for enhancing anomaly detection models in complex CPS environments.

Kumar et al. (2021) address the rising network security threats due to increased web traffic and emphasize the role of Network Intrusion Detection Systems (NIDS) [20]. It proposes a framework using supervised and unsupervised machine learning to detect various network attacks. Five algorithms—Random Forest, Decision Tree, Logistic Regression, K-Nearest Neighbors, and Artificial Neural Networks—are evaluated on the UNSW-NB15 dataset, achieving a top accuracy of 89.29% with Random Forest. Applying Synthetic Minority Oversampling Technique (SMOTE) improves accuracy to 95.1% when combined with Principal Component Analysis (PCA) for feature selection.

As ICSs become more complex and generate larger volumes of data, there is a growing need for scalable and real-time anomaly detection solutions. Future research should focus on developing efficient algorithms that can process high-dimensional data streams in real-time while maintaining high detection accuracy. Moreover, the dynamic nature of ICS and evolving attack patterns necessitate the development of adaptive and self-learning anomaly detection systems. Such approaches that can continuously adapt to changing system behaviors and emerging threats are promising avenues for future research. Additionally, incorporating domain-specific knowledge into anomaly detection models remains a challenge. Future research should explore methods to effectively combine data-driven approaches with expert knowledge to improve detection performance and reduce false alarms. As ICS become more interconnected, privacy concerns arise when sharing data for anomaly detection. Developing privacy-preserving techniques that allow collaborative anomaly detection without compromising sensitive information is an important area for future investigation.

III. Methodology

A. Dataset

The Secure Water Treatment (SWaT) testbed was developed by researchers and students at the Singapore University of Technology and Design in the context of Cyber-Physical systems security research [1]. It is an operational scaled-down water treatment plant with six process stages that can produce 5 gallons per minute of double-filtered water. The network architecture is a distributed control system where each stage of the process is under the control of programmable logic. Various datasets are available for the SWaT testbed. This study utilizes the SWaT A1 & A2-Dec 2015 dataset, which comprises recordings from network traffic and 51 field instruments over 11 days of continuous operation. The first seven days of data were collected under normal operation, while the remaining four days included 36 simulated attacks.

In this dataset, attackers hijacked data packets through the Level 1 communication link, manipulated the sensor data, and sent them to the PLCs [21]. While the physical properties of the field instruments were recorded periodically as process data in the Historian server, network data was captured from Level 1 at a much higher frequency. The network data is claimed to only include information valuable for intrusion

detection. All process and network data are timestamped, allowing the dataset provider to flag data occurring within an attack.

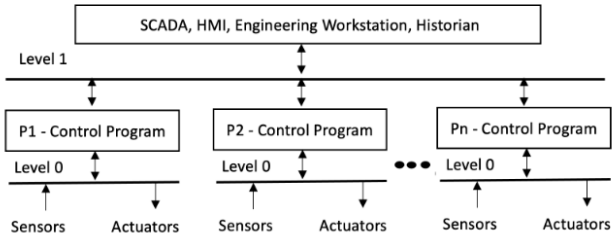

Fig. 2. Overview of SWaT Network Architecture, adapted from [21].

TABLE I. TYPE AND DESCRIPTION OF EACH SWAT SENSOR/ACTUATOR

No	Name	Type	Description
1	FIT-101	Sensor	Flow meter; Measures inflow into raw water tank.
2	LIT-101	Sensor	Level Transmitter; Raw water tank level.
3	MV-101	Actuator	Motorized valve; Controls water flow to the raw water tank.
4	P-101	Actuator	Pump; Pumps water from raw water tank to second stage.
5	P-102 (backup)	Actuator	Pump; Pumps water from raw water tank to second stage.
6	AIT-201	Sensor	Conductivity analyser; Measures NaCl level.
7	AIT-202	Sensor	pH analyser; Measures HCl level.
8	AIT-203	Sensor	ORP analyser; Measures NaOCl level.
9	FIT-201	Sensor	Flow Transmitter; Control dosing pumps.
10	MV-201	Actuator	Motorized valve; Controls water flow to the UF feed water tank.
11	P-201	Actuator	Dosing pump; NaCl dosing pump.
12	P-202 (backup)	Actuator	Dosing pump; NaCl dosing pump.
13	P-203	Actuator	Dosing pump; HCl dosing pump.
14	P-204 (backup)	Actuator	Dosing pump; HCl dosing pump.
15	P-205	Actuator	Dosing pump; NaOCl dosing pump.
16	P-206 (backup)	Actuator	Dosing pump; NaOCl dosing pump.
17	DPIT-301	Sensor	Differential pressure indicating transmitter; Controls the back-wash process.
18	FIT-301	Sensor	Flow meter; Measures the flow of water in the UF stage.
19	LIT-301	Sensor	Level Transmitter; UF feed water tank level.
20	MV-301	Actuator	Motorized Valve; Controls UF-Backwash process.
21	MV-302	Actuator	Motorized Valve; Controls water from UF process to De-Chlorination unit.
22	MV-303	Actuator	Motorized Valve; Controls UF-Backwash drain.
23	MV-304	Actuator	Motorized Valve; Controls UF drain.
24	P-301 (backup)	Actuator	UF feed Pump; Pumps water from UF feed water tank to RO feed water tank via UF filtration.
25	P-302	Actuator	UF feed Pump; Pumps water from UF feed water tank to RO feed water tank via UF filtration.
26	AIT-401	Sensor	RO hardness meter of water.
27	AIT-402	Sensor	ORP meter; Controls the $NaHSO_3$dosing(P203), NaOCl dosing (P205).
28	FIT-401	Sensor	Flow Transmitter ; Controls the UV dechlorinator.
29	LIT-401	Actuator	Level Transmitter; RO feed water tank level.
30	P-401 (backup)	Actuator	Pump; Pumps water from RO feed tank to UV dechlorinator.
31	P-402	Actuator	Pump; Pumps water from RO feed tank to UV dechlorinator.
32	P-403	Actuator	Sodium bi-sulphate pump.
33	P-404 (backup)	Actuator	Sodium bi-sulphate pump.
34	UV-401	Actuator	Dechlorinator; Removes chlorine from water.
35	AIT-501	Sensor	RO pH analyser; Measures HCl level.
36	AIT-502	Sensor	RO feed ORP analyser; Measures NaOCl level.
37	AIT-503	Sensor	RO feed conductivity analyser; Measures NaCl level.
38	AIT-504	Sensor	RO permeate conductivity analyser; Measures NaCl level.
39	FIT-501	Sensor	Flow meter; RO membrane inlet flow meter.
40	FIT-502	Sensor	Flow meter; RO Permeate flow meter.
41	FIT-503	Sensor	Flow meter; RO Reject flow meter.
42	FIT-504	Sensor	Flow meter; RO re-circulation flow meter.
43	P-501	Actuator	Pump; Pumps dechlorinated water to RO.
44	P-502 (backup)	Actuator	Pump; Pumps dechlorinated water to RO.
45	PIT-501	Sensor	Pressure meter; RO feed pressure.
46	PIT-502	Sensor	Pressure meter; RO permeate pressure.
47	PIT-503	Sensor	Pressure meter;RO reject pressure.
48	FIT-601	Sensor	Flow meter; UF Backwash flow meter.
49	P-601	Actuator	Pump; Pumps water from RO permeate tank to raw water tank (not used for data collection).
50	P-602	Actuator	Pump; Pumps water from UF back wash tank to UF filter to clean the membrane.
51	P-603	Actuator	Not implemented in SWaT yet.

The SWaT process begins with raw water intake and storage in a tank. The water then undergoes pre-treatment, where its quality is assessed, and chemical dosing is performed if necessary. The water then reaches the third process stage, P3, where undesirable materials are removed using fine filtration membranes. After the residuals are filtered through the Ultrafiltration system, any remaining chlorine is destroyed in the Dechlorination process using Ultraviolet lamps. Subsequently, the water from P4 is pumped into the Reverse Osmosis system to reduce inorganic impurities. In the final process, P6, the water from the Reverse Osmosis system is stored and ready for distribution in a water distribution system. In the case of SWaT, the treated water can be transferred back to the raw tank for reprocessing, but for the purpose of data collection, the water from P6 is disposed of to mimic water distribution [22].

Characteristics of the dataset (SWaT.A1_Dec 2015):

- 11 days of continuous operation: 7 under normal operation and 4 days with attack scenarios.
- Collected network traffic & all the values obtained from all the 51 sensors and actuators.
- Data labeled according to normal and abnormal behaviors.
- Attack Scenarios: Derived through the attack models developed by our research team. The attack model considers the intent space of a CPS as an attack model. 41 attacks were launched.

B. Data Preprocessing

In this section, we use a Python script to read and preprocess the multiple Excel files containing the SWaT 2015 data. Three Excel files are read, containing data on normal and attack instances in the SWaT system. The data is concatenated into a single dataframe, making it easier to manipulate and analyze.

The dataset contains detailed information about the SWaT system, including 946,719 timestamped observations of sensor readings and control signals, starting at 2015-12-22 16:00:00, and ending at 2016-01-02 14:59:59, and it includes both normal operation and attack instances. The data is stored in a DataFrame with 53 columns, comprising 25 float64 columns, and 26 int64 columns, in addition to the Timestamp and Normal/Attack columns. The entire dataset occupies approximately 390 MB of memory.

The analysis of the number of unique values in the dataset provides insights into the diversity of the dataset. Columns with few unique values likely represent categorical or binary features, while those with many unique values indicate continuous or granular measurements. The 'Normal/Attack' column has three unique values: 'Normal', 'Attack', and 'A ttack'. The latter likely represents a typo, which is corrected to 'Attack' for consistency. Certain sensors (FIT101, LIT101, etc.) exhibit a wide range of unique values, reflecting variability in measurements, while binary columns (P101, P102, etc.) have a limited number of unique values, reflecting discrete operational states.

The Normal/Attack column, representing the status of the water treatment process (normal or attack), is examined and corrected for consistent labeling.

- Normal: The majority of instances are labeled as 'Normal'. There are 892,098 (94.23%) instances labeled as 'Normal', indicating normal operational periods.
- Attack: There are fewer instances labeled as 'Attack', with around 54,621(5.76%) occurrences.

The dataset is highly skewed towards normal instances, with attack instances being relatively rare, indicating class

imbalance. A histogram is used to visualize the frequency of 'Normal' and 'Attack' labels, highlighting the occurrence of normal and attack states in the dataset. The plot confirms the imbalance between normal and attack instances, which is essential to address during model training.

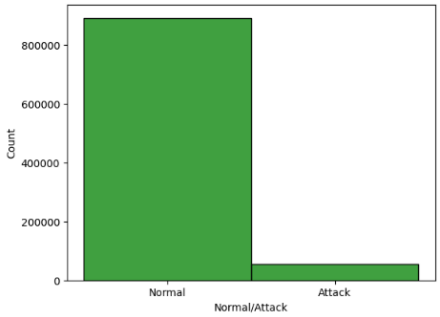

Fig. 3. Count of Normal and Attack instances in the SWaT.A1_Dec 2015 dataset

Attack information is read from an Excel file ('List_of_attacks_Final.xlsx') into a DataFrame df_attack_readings. This DataFrame contains details about various attacks, including attack points, start and end times, and their impacts. We created a new DataFrame by filtering the original DataFrame to include only rows where the 'Timestamp' column matches any value in the datetimes list that includes all the timestamps where the Target value was a recorded attack found in the Excel file. This operation results in a DataFrame of all the instances of the 36 recorded attacks, with 53,900 rows and 53 columns. We created another DataFrame by filtering the original DataFrame to include only rows where the 'Target' column value is 'Attack'. The resulting DataFrame has 54,621 rows and 53 columns. This reveals that there are 721 attack instances that were not recorded in the Excel file ('List_of_attacks_Final.xlsx'). Further analysis reveals that the unrecorded attack starts at 2015-12-29 06:30:00 and ends at 2015-12-29 06:42:00.

Then, we used the Excel file ('List_of_attacks_Final.xlsx') to update the 'Target' column in the original DataFrame with attack point labels. The attack points in the 'Target' column in df_attack_readings were originally separated sometimes by semicolons and other times by commas. In order to fix that, we replace semicolons (;) with commas (,), and then use commas to split the string values into lists of attack points. Moreover, the attack point 'DIT-301' is found in the Excel file ('List_of_attacks_Final.xlsx'), which is a typo of 'DPIT-301' The erroneous value is therefore replaced with 'DPIT-301' for consistency. Next, we counted the occurrences of each unique attack point in the Target column of the resulting DataFrame. The following graph demonstrates that the most frequent attack target is 'P-302', followed by 'LIT-301', with the top attack type 'P-302' constituting over 55% of the attack instances. We save the preprocessed training dataset to a CSV file. The DataFrame is exported to a CSV file named 'training_2015_attack.csv' for future use.

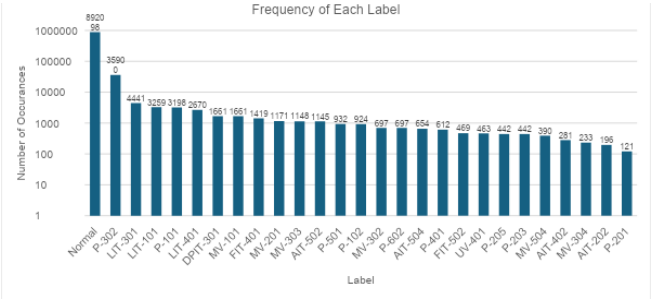

Fig. 4. Histogram showing the number of normal instances and the number of instances that belong to each attack point. (*Plotted in Log scale for clarity*)

For the binary classification task, we one-hot-encoded the "Target" column to a binary classification of "Normal" or "Attack". The OneHotEncoder is used to transform categorical variables into a binary format, where each category is represented by a binary vector. This simplifies the classification task by converting multiclass labels into binary labels and preparing the dataset for training deep learning models designed for binary classification [23].

As for the multilabel classification task, the Multilabel Binarizer utility is used. It is specifically designed to handle scenarios where each sample in a dataset can belong to multiple classes simultaneously. Given a dataset with samples that can belong to multiple classes, the Multilabel Binarizer encodes these class labels into a binary format. For each sample, it creates a binary array where each element represents the presence or absence of a particular class [24]. If a sample belongs to a particular attack point out of the 26 attack points present in the data, the corresponding element in its binary array is set to 1; otherwise, it's set to 0. This binary encoding allows multi-label classification algorithms to process the data effectively. In contrast to traditional classification tasks where each sample belongs to only one class, multi-label classification tasks allow for samples to have multiple class labels simultaneously. The Multilabel Binarizer facilitates the handling of such scenarios by transforming the class labels into a suitable binary representation.

The dataset is then split into training and testing sets using an 80-20 ratio. The input features are standardized using the StandardScaler, which removes the mean and scales them to unit variance (Z-score normalization). This ensures equal feature contribution and stable optimization for algorithms like neural networks. The resulting feature matrices and target vectors are represented as X_train, X_test, y_train, and y_test.

C. Evaluation Metrics

Accuracy, precision, recall, and F1 Score are metrics commonly used to evaluate the performance of classification models.

1) Accuracy:

Accuracy is a fundamental metric used to evaluate the overall performance of a classification model. It measures the proportion of correctly classified instances out of the total instances in the dataset. The formula for accuracy is defined as:

Accuracy = Number of correctly classified instances / Total number of instances ×100%

where:

- Number of correctly classified instances: The total number of instances correctly predicted by the model (TP + TN).
- Total number of instances: The total number of instances in the dataset, including both normal and abnormal operation states.

High accuracy values indicate that the model is effectively distinguishing between different operation states. However, accuracy alone may not provide a comprehensive assessment of model performance, especially in scenarios with imbalanced class distributions [25]. Therefore, it is essential to complement accuracy with other metrics such as precision, recall, and F1-score to gain a more nuanced understanding of the model's capabilities.

2) Precision:

Precision measures the ratio of correctly predicted positive observations to the total predicted positives [26]. It is calculated as:

$$\text{Precision} = TP / (TP + FP)$$

where:

- TP (True Positives) is the number of correctly predicted positive instances.
- FP (False Positives) is the number of incorrectly predicted positive instances.

Precision focuses on the accuracy of positive predictions, indicating how many of the predicted positive instances are actually positive.

3) Recall:

Recall, also known as sensitivity or true positive rate, measures the ratio of correctly predicted positive observations to the total actual positives [26]. It is calculated as:

$$\text{Recall} = TP / (TP + FN)$$

where:

- TP (True Positives) is the number of correctly predicted positive instances.
- FN (False Negatives) is the number of positive instances incorrectly predicted as negative.

Recall emphasizes the ability of the model to capture all positive instances, indicating how many of the actual positive instances were correctly predicted.

4) F1 Score:

F1 Score is the harmonic mean of precision and recall, providing a single metric that balances both precision and recall. It is calculated as:

$$F1 = (2 \times \text{Precision} \times \text{Recall}) / (\text{Precision} + \text{Recall})$$

F1 Score provides a holistic measure of a model's performance, considering both false positives and false negatives. It is particularly useful when dealing with imbalanced datasets, as it accounts for the trade-off between precision and recall [27].

5) Cohen's Kappa Score:

Cohen's kappa score measures the agreement between two raters who each classify items into mutually exclusive categories. It considers the possibility of the agreement occurring by chance. The formula for Cohen's kappa is:

$$\text{Cohen's Kappa} = (\text{Observed Agreement} - \text{Expected Agreement}) / (1 - \text{Expected Agreement})$$

Cohen's kappa provides a measure of inter-rater reliability, particularly useful when assessing the performance of classification models where there are two or more raters.

In the context of classification models, scikit-learn provides a method to calculate Cohen's kappa score for multiclass classification models using the cohen_kappa_score function. This function computes Cohen's kappa statistic for classification tasks with more than two classes. It takes the true labels and predicted labels as inputs and returns Cohen's kappa score [28].

6) Average Scores:

a) Micro Average:

Micro average calculated metrics globally by counting the total true positives, false positives, and false negatives across all classes. It is suitable for evaluating performance in multiclass classification problems, as it treats each instance equally regardless of class.

b) Macro Average:

Macro average calculates metrics independently for each class and then takes the unweighted average. It gives equal weight to each class, making it suitable for evaluating performance across all classes, especially in imbalanced datasets.

c) Weighted Average:

Weighted average calculates metrics for each class and then takes the average weighted by the number of true instances in each class. It is useful when there is a class imbalance, as it considers the importance of each class based on its representation in the dataset.

Using macro average F1 Score in imbalanced data helps to address the issue of class imbalance by giving equal importance to each class. It provides a balanced evaluation of the model's performance across all classes, regardless of their prevalence in the dataset. This ensures that the evaluation metric is not biased towards the majority class and provides insights into how well the model performs for each class independently.

D. Activation Functions

1) Sigmoid Activation Function

The sigmoid activation function, also known as the logistic function, is commonly used in the output layer of neural networks for binary classification tasks [29]. It squashes the output of the neural network to a range between 0 and 1, which can be interpreted as probabilities. Mathematically, the sigmoid function is defined as:

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

where:

- x is the input to the function.
- e is the base of the natural logarithm (Euler's number).

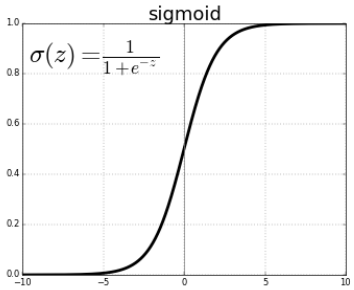

Fig. 5. Visualization of the sigmoid function.

The output of the sigmoid function represents the probability that the input belongs to the positive class (class 1). Since the sigmoid function outputs values between 0 and 1, it is commonly used to model binary outcomes, where values closer to 1 indicate a higher probability of belonging to the positive class, and values closer to 0 indicate a higher probability of belonging to the negative class [29]. In binary classification tasks, a decision threshold (typically 0.5) is used to classify the inputs. Outputs greater than or equal to the threshold are classified as belonging to the positive class, while outputs less than the threshold are classified as belonging to the negative class.

The sigmoid function is smooth and differentiable everywhere, making it suitable for gradient-based optimization algorithms like gradient descent. This allows for efficient training of neural networks using techniques such as backpropagation.

2) ReLU Activation Function

ReLU, which stands for Rectified Linear Unit, is a popular activation function used in neural networks. It introduces non-linearity to the model, allowing it to learn complex patterns and relationships in the data. The ReLU function is defined as: $f(x) = max(0, x)$. In other words, it outputs the input x if it is positive, and zero otherwise [30]. Visually, ReLU looks like a linear function with a "bend" at x=0, which results in a piecewise-linear function.

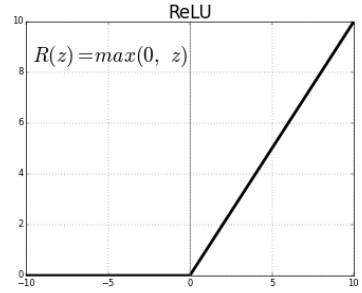

Fig. 6. Visualization of the ReLU function.

There are several main advantages of ReLU. First, ReLU produces sparse activation, meaning that only a subset of neurons are activated while others remain inactive. This sparsity can help with reducing computational complexity and overfitting. ReLU is also computationally efficient and allows for faster training of deep neural networks compared to traditional activation functions like sigmoid or tanh, which suffer from the vanishing gradient problem. Additionally, ReLU does not saturate for positive values of x, which means it does not suffer from the vanishing gradient problem for positive inputs. This property helps with training deeper networks.

However, ReLU also has some limitations. Neurons in the ReLU activation function can become "dead" if they always output zero for any input. This can happen during training if the weights are updated in such a way that a neuron always receives negative inputs. ReLU also does not have an upper bound, which means that it can produce very large values for positive inputs. This can lead to numerical instability during training, especially in deep networks. Despite these limitations, ReLU remains one of the most commonly used activation functions in deep learning due to its simplicity, effectiveness, and computational efficiency.

E. Binary Cross-Entropy Loss Function

Binary cross-entropy loss, also known as log loss or logistic loss, is a common loss function used in binary classification tasks. It measures the difference between the true binary labels and the predicted probabilities output by the model. Mathematically, the binary cross-entropy loss function is defined as [31]:

$$L_{BCE} = -\frac{1}{n}\sum_{i=1}^{n}\left(Y_i \cdot \log\ \hat{Y}_i + (1 - Y_i) \cdot \log\left(1 - \hat{Y}_i\right)\right)$$

For each sample in the dataset, the model predicts a probability p that the sample belongs to the positive class (class 1) and 1−p probability that it belongs to the negative class (class 0). If the true label for the sample is 1, the binary cross-entropy loss for that sample is calculated as −log(p), penalizing lower probabilities of the positive class. If the true label for the sample is 0, the binary cross-entropy loss for that sample is calculated as −log(1−p), penalizing lower probabilities of the negative class. The average binary cross-entropy loss across all samples in the dataset is then computed to obtain the final loss value for the model.

The binary cross-entropy loss function encourages the model to output high probabilities for the true labels and low probabilities for the false labels. It is widely used in binary classification tasks because it provides a smooth and differentiable measure of how well the model is performing, making it suitable for optimization using gradient-based methods like stochastic gradient descent.

F. Adam Optimizer

Adam optimizer, short for Adaptive Moment Estimation, is an optimization algorithm used for training artificial neural networks. It combines ideas from two other popular optimization methods: RMSprop (Root Mean Square Propagation) and Momentum.

Adam maintains two moving averages of the gradients: m (the first moment) and v (the second moment). These moving averages are initialized as vectors of zeros. During each iteration of training, the gradients of the loss function with respect to the model parameters are computed using backpropagation. Because m and v are initialized as vectors of zeros, they are biased towards zero, especially during the early iterations. Adam performs bias correction to counteract this bias. It calculates bias-corrected estimates of m and v by scaling them with correction terms [32].

Adam updates the model parameters using a learning rate α scaled by the moving averages of the gradients. It also includes an additional hyperparameter ϵ to prevent division by zero.

$$\theta_{t+1} = \theta_t - \frac{\alpha}{\sqrt{\hat{v}_t} + \epsilon} \cdot \hat{m}_t$$

where:

- θ_t is the parameter at iteration t. $\hat{m}_t$ and $\hat{v}_t$ are the bias-corrected estimates of the first and second moments of the gradients, respectively.
- α is the learning rate. ϵ is a small constant (typically around 10^{-8}) to prevent division by zero.

Adam optimizer offers several advantages. It adapts the learning rate for each parameter individually, which can be beneficial for training models with sparse gradients or noisy data. It also incorporates momentum, allowing it to navigate the loss landscape more efficiently by accumulating gradients from past iterations. Additionally, it performs well in practice across a wide range of deep learning tasks and architectures.

G. Classification Models

1) LSTM

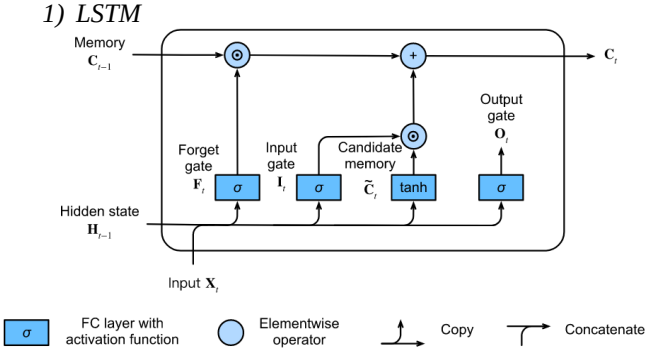

Fig. 7. LSTM cell structure.

Long Short-Term Memory (LSTM) is a type of recurrent neural network (RNN) architecture designed to address the vanishing gradient problem in traditional RNNs. It is particularly effective in capturing and modeling sequential data with long-range dependencies. LSTM networks contain memory cells that can maintain information over long periods of time. These memory cells are equipped with a gating mechanism that allows them to selectively add or remove information from the cell state [33]. LSTM networks have three main gates:

1. Forget Gate: Decides what information to discard from the cell state.
2. Input Gate: Determines what new information to store in the cell state.
3. Output Gate: Controls what information to output based on the current input and the cell state.

The cell state serves as a conveyor belt that runs through the entire sequence, allowing information to flow across time steps while being carefully regulated by the gates. The gates in LSTM are controlled by activation functions, typically sigmoid (to regulate the flow of information) and hyperbolic tangent (to regulate the values of the cell state and the output). The LSTM architecture enables the network to learn and remember information over long sequences, making it well-suited for tasks involving time-series data, natural language processing (NLP), speech recognition, and more.

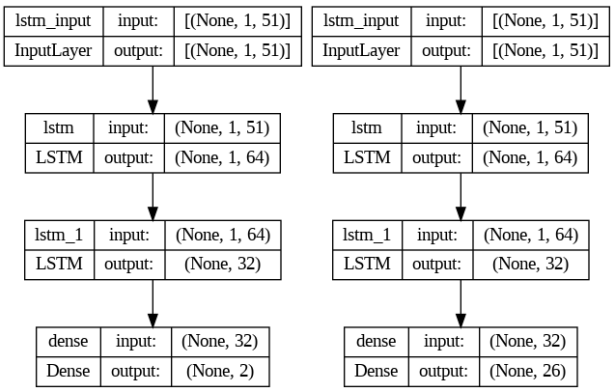

Fig. 8. LSTM model architectures used in binary classification (left) and multilabel classification (right).

The model architecture includes two LSTM layers, with the first layer having 64 units and the second layer having 32 units. To prevent overfitting and improve the model's generalization, 20% dropout regularization is applied to the second LSTM layer. Dropout regularization is a technique used in neural networks, including LSTMs, that randomly drops a fraction of input units during training, forcing the network to learn more robust features. This helps prevent overfitting and improves the model's ability to generalize to unseen data. Finally, the model includes a dense output layer with sigmoid activation. The sigmoid activation function is commonly used in the output layer of neural networks for classification tasks. It maps the output to a range between 0 and 1, representing the probability that the input belongs to a certain class. The sigmoid function is smooth and differentiable, making it suitable for gradient-based optimization and efficient training of neural networks.

The model is compiled with the binary cross-entropy loss function and the Adam optimizer with a 0.001 learning rate. Additionally, the F1 score metric is used to evaluate the model during training using the validation data during training to monitor model performance. Macro Average F1 Score is chosen as the evaluation metric as it is suitable for imbalanced datasets. The model is trained for 10 epochs with a batch size of 32. The model aims to minimize the loss function and improve the F1 Score across epochs.

To classify the observations into the Attack or Normal classes, the LSTM model uses the entire dataset of 946,719 observations split using an 80-20 ratio into train and test subsets, and the model has a total of 42,178 parameters, all trainable. Whereas, to classify the observations into their corresponding Attack Point(s), the LSTM model uses the 53,900 recorded attack observations, also split into train and test subsets using an 80-20 ratio, and the model has a total of 42,970 parameters, all trainable as well.

2) 1-D CNN

A 1D Convolutional Neural Network (CNN) is a type of neural network architecture commonly used for processing sequential data, such as time series, signals, text, and audio. Unlike traditional CNNs that operate on two-dimensional grids of data (e.g., images), 1D CNNs operate on one-dimensional sequences. The key components of a 1D CNN include [34]:

1. Convolutional Layers: These layers apply convolution operations to the input sequences using learnable filters. The filters slide across the input

sequence, extracting local patterns and features. Convolutional layers in 1D CNNs have one-dimensional kernels that move along the sequence, capturing patterns in the temporal domain. Non-linear activation functions like ReLU (Rectified Linear Unit) are typically applied after convolution operations to introduce non-linearity into the model, allowing it to learn complex relationships in the data.

2. Pooling Layers: Pooling layers downsample the feature maps produced by convolutional layers, reducing the spatial dimensionality of the data. Common pooling operations include max pooling, which extracts the maximum value from each sub-region, and average pooling, which computes the average value.

3. Flatten Layer: Before passing the output of convolutional and pooling layers to fully connected layers, a Flatten layer is typically added. This layer transforms the multi-dimensional feature maps into a one-dimensional vector, effectively "flattening" the data. By reshaping the output from the convolutional and pooling layers into a vector format, the Flatten layer enables seamless connectivity with the subsequent fully connected layers. This step is crucial for maintaining the sequential flow of information and facilitating the integration of extracted features for subsequent classification tasks.

4. Fully Connected Layers: After the convolutional and pooling layers, one or more fully connected layers are often used to integrate the extracted features and make predictions. These layers connect every neuron in one layer to every neuron in the next layer, enabling the model to learn complex relationships.

5. Output Layer: The output layer of a 1D CNN produces the final predictions. Depending on the task, the output layer may consist of one or more neurons with appropriate activation functions.

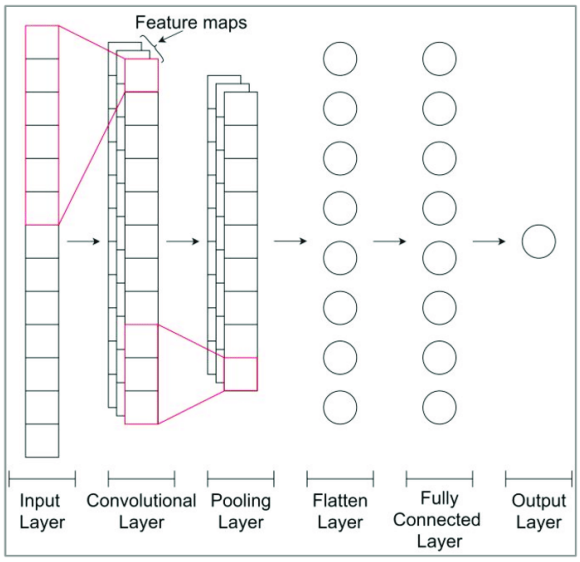

Fig. 9. Layers in the CNN model.

1D CNNs can capture local patterns and dependencies in sequential data, making them suitable for a wide range of applications where the order of elements matters.

The trained data is prepared by reshaping the data for CNN input, which involves adding an extra dimension to represent the channel (in this case, 1 channel). The CNN model architecture is then defined using the Keras Sequential API.

The model consists of a convolutional layer with 64 filters and a kernel size of 3, followed by a max-pooling layer with a pool size of 2. Then, a Flatten layer is added to convert the 2D output into a 1D array. Two dense layers follow, a dense layer with 32 units and a ReLU activation function, and the final dense layer with a sigmoid activation function.

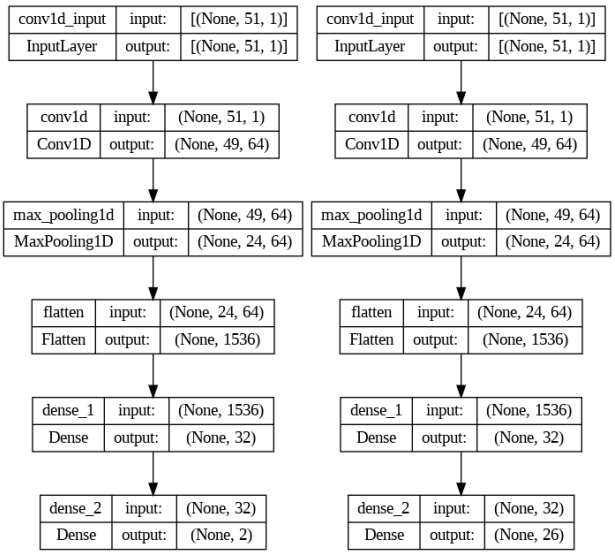

Fig. 10. CNN model architectures used in binary classification (left) and multilabel classification (right).

The model is compiled with the binary cross-entropy loss function and the Adam optimizer with a 0.001 learning rate. Additionally, the F1 score metric is used to evaluate the model during training. The model is trained for 10 epochs. Macro Average F1 Score is chosen as the evaluation metric as it is suitable for imbalanced datasets.

To classify the observations into the Attack or Normal classes, the 1-D CNN model uses the entire dataset of 946,719 observations split using an 80-20 ratio into train and test subsets, and the model has a total of 49,506 parameters, all trainable. Whereas to classify the observations into their corresponding Attack Point(s), the 1-D CNN model uses the 53,900 recorded attack observations, also split into train and test subsets using an 80-20 ratio, and the model has a total of 50,298 parameters, all trainable as well.

3) 1-D CNN-LSTM

A 1D CNN-LSTM is a neural network architecture that combines the strengths of both convolutional neural networks (CNNs) and long short-term memory (LSTM) networks to process and learn from sequential data [35]. The 1D CNN part of the architecture is used to extract relevant features from the input sequential data. In the context of time-series data, such as the one you provided about the cyberattacks on a secure water treatment plant (SWaT), the 1D CNN filters can capture patterns and local dependencies in the data. The LSTM part of the architecture is designed to model temporal dependencies and long-range dependencies in sequential data. LSTMs are well-suited for capturing patterns that span across multiple time steps and for learning from sequences of varying lengths.

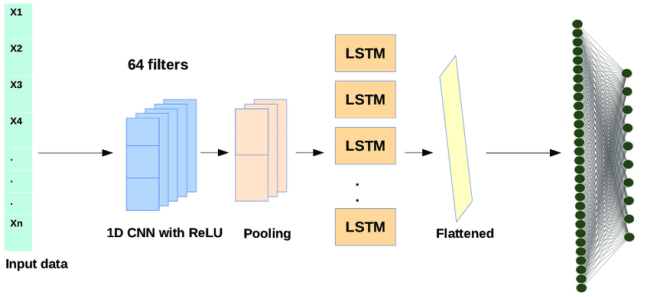

Fig. 11. Layers in the CNN-LSTM model.

In a 1D CNN-LSTM architecture, the output of the CNN layers is fed into the LSTM layers. This allows the model to learn hierarchical representations of the input data, with the CNN layers capturing local features and the LSTM layers capturing temporal dependencies. The combination of CNN and LSTM layers enables the model to effectively learn from sequential data, such as time-series data, while also leveraging the ability of CNNs to automatically extract relevant features from the input [35].

In the context of cyberattacks on a secure water treatment plant (SWaT), a 1D CNN-LSTM model could be trained on the time-series sensor data collected from the plant to detect and classify different types of attacks or anomalous behavior. By combining CNNs for feature extraction and LSTMs for sequence modeling, the model can learn to identify complex patterns indicative of cyberattacks while accounting for the temporal nature of the data.

The CNN-LSTM model starts by reshaping the data for CNN input, adding an extra dimension to represent the channel (in this case, 1 channel). The CNN-LSTM model architecture is then defined using the Keras Sequential API. The model architecture starts with a convolutional layer with 64 filters and a kernel size of 3, followed by a max-pooling layer with a pool size of 2. Next, two LSTM layers are added with 64 and 32 units respectively, where the first LSTM layer returns sequences. A flatten layer is introduced to convert the 3D output into a 2D array, followed by one dense layer with 32 units and a ReLU activation function, and the final dense layer with a sigmoid activation function.

The model is compiled with the binary cross-entropy loss function and the Adam optimizer with a 0.001 learning rate. Additionally, the F1 score metric is used to evaluate the model during training. The model is trained for 10 epochs.

To classify the observations into the Attack or Normal classes, the LSTM model uses the entire dataset of 946,719 observations split using an 80-20 ratio into train and test subsets, and the model has a total of 46,818 parameters, all trainable. Whereas, to classify the observations into their corresponding Attack Point(s), the LSTM model uses the 53,900 recorded attack observations, also split into train and test subsets using an 80-20 ratio, and the model has a total of 47,610 parameters, all trainable as well.

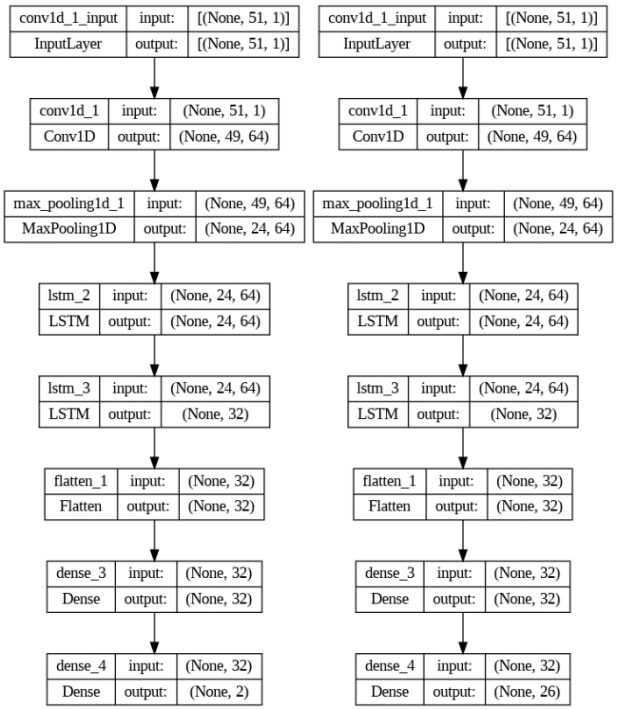

Fig. 12. CNN-LSTM model architectures used in binary classification (left) and multilabel classification (right).

4) MLP

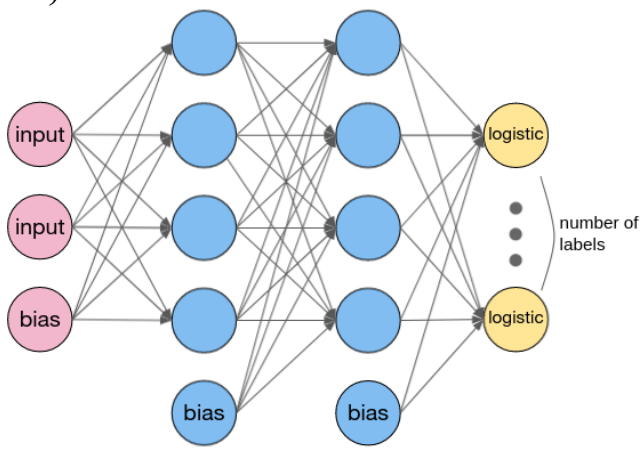

Fig. 13. Layers in the MLP model.

MLP stands for Multi-Layer Perceptron. It's a type of artificial neural network that consists of multiple layers of nodes, each connected to the nodes in the adjacent layers. The MLP is a feedforward neural network, meaning the information flows in one direction, from the input layer through the hidden layers to the output layer. MLP consists of [36]:

1. Input Layer: This layer consists of nodes representing input features. Each node corresponds to a feature in the input data.
2. Hidden Layers: These are intermediate layers between the input and output layers. Each hidden layer consists of nodes (neurons) that perform computations on the input data using weights and activation functions.
3. Output Layer: The final layer in the network produces the output. In classification tasks, each node in the

output layer typically represents a class label, and the output values are interpreted as probabilities or class scores.

Other key features of MLP include non-linear activation functions (such as ReLU, sigmoid, or tanh) are applied to the output of each node in the hidden layers to introduce non-linearity into the model, enabling it to learn complex patterns in the data. MLPs are also trained using algorithms like backpropagation, which adjusts the weights of connections between nodes to minimize the difference between the predicted outputs and the actual targets in the training data.

MLPs are widely used in various machine learning tasks, including classification, regression, and pattern recognition. They are known for their flexibility and ability to approximate complex nonlinear functions, making them a popular choice for many applications in artificial intelligence and data science.

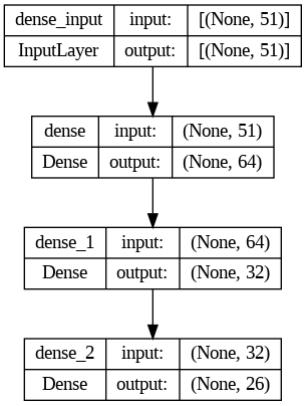

Fig. 14. MLP model architectures used in multilabel classification.

For the multilabel classification task, the Multilayer Perceptron (MLP) model is built with three dense layers. The input layer has 64 units with ReLU activation, followed by a hidden layer of 32 units with ReLU activation, and a final output layer with sigmoid activation. The model is compiled with binary cross-entropy loss and the Adam optimizer, with 6,266 trainable parameters. The F1 score metric is used for evaluation. The MLP model is trained for 10 epochs with a batch size of 32, and performance metrics are monitored.

IV. Results

A. Binary Classification

1) LSTM

The trained LSTM model is evaluated on the test set, yielding an accuracy of 0.998, precision of 0.990, recall of 0.991, F1 score of 0.991, and a kappa score of 0.981. The model achieves high performance across various metrics, indicating its effectiveness in classifying between normal and attack instances. The confusion matrix visually represents the model's ability to correctly classify instances as either normal or attack, showing high true positive and true negative rates.

The classification report summarizes the precision, recall, and F1 score for each class (normal and attack). It provides a comprehensive assessment of the model's classification performance for both classes. For the "Normal" class, precision, recall, and F1-score are all approximately 1. For the "Attack" class, precision, recall, and F1-score are all approximately 0.98, indicating strong performance in both classes.

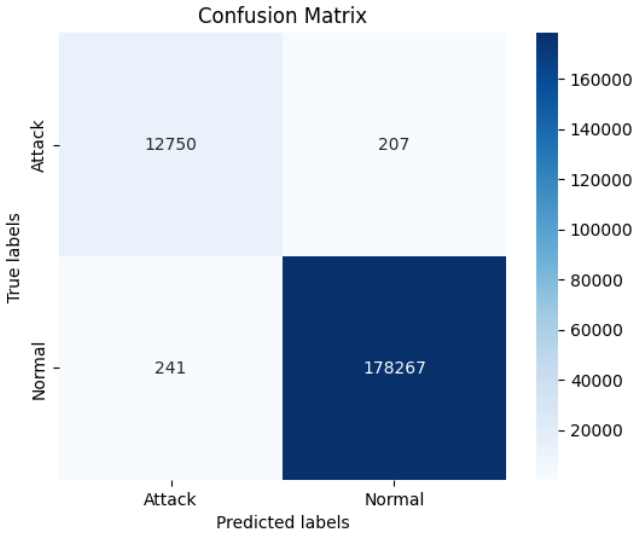

Fig. 15. Confusion matrix of the LSTM model used in the binary classification task.

TABLE II. CLASSIFICATION REPORT FOR THE LSTM MODEL USED IN THE BINARY CLASSIFICATION TASK

	Precision	Recall	F1-Score	Support
Attack	0.98	0.98	0.98	12957
Normal	1.00	1.00	1.00	178508
Accuracy			1.00	191465
Macro Avg	0.99	0.99	0.99	191465
Weighted Avg	1.00	1.00	1.00	191465

2) 1-D CNN

The trained CNN model is evaluated on the test set, yielding an accuracy of 0.997, precision of 0.990, recall of 0.987, F1 score of 0.989, and a kappa score of 0.977. The model achieves high performance across various metrics, indicating its effectiveness in classifying between normal and attack instances. The confusion matrix visually represents the model's ability to correctly classify instances as either normal or attack, showing high true positive and true negative rates.

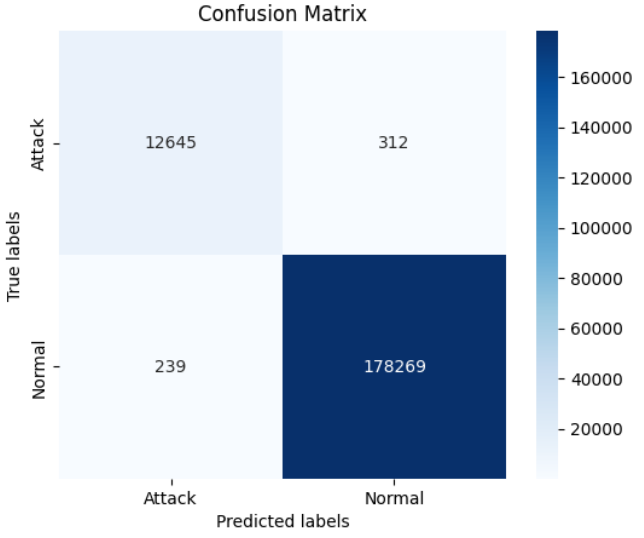

Fig. 16. Confusion matrix of the 1-D CNN model used in the binary classification task.

The classification report is printed, revealing precision, recall, F1-score, and support for each class (normal and attack). For the normal class, precision, recall, and F1-score are all approximately 1. For the attack class, precision, recall, and F1-score are all approximately 0.98, indicating strong performance in both classes.

TABLE III. CLASSIFICATION REPORT FOR THE 1-D CNN MODEL USED IN THE BINARY CLASSIFICATION TASK

	Precision	Recall	F1-Score	Support
Attack	0.98	0.98	0.98	12957
Normal	1.00	1.00	1.00	178508
Accuracy			1.00	191465
Macro Avg	0.99	0.99	0.99	191465
Weighted Avg	1.00	1.00	1.00	191465

3) CNN-LSTM

The trained model is evaluated on the test set, yielding an accuracy of 0.997, precision of 0.994, recall of 0.992, F1 score of 0.993, and a kappa score of 0.986. The model demonstrates high performance across various metrics, indicating its effectiveness in classifying between normal and attack instances. The confusion matrix visually represents the model's ability to correctly classify instances as either normal or attack, showing high true positive and true negative rates.

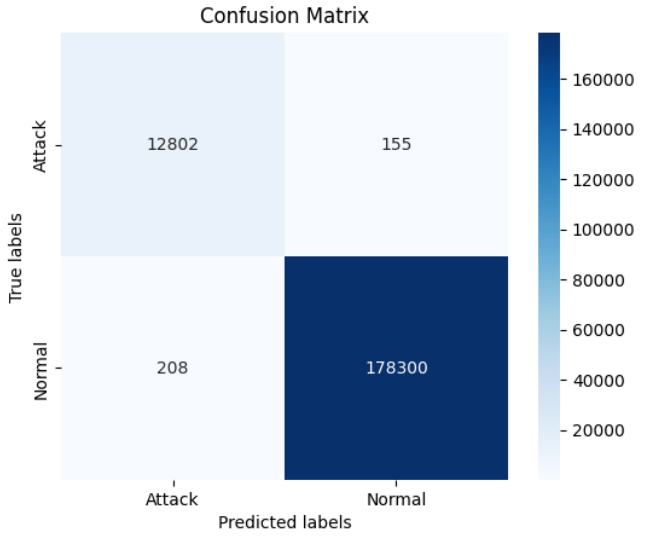

Fig. 17. Confusion matrix of the CNN-LSTM model used in the binary classification task.

Fig. 18. CLASSIFICATION REPORT FOR THE CNN-LSTM MODEL USED IN THE BINARY CLASSIFICATION TASK

	Precision	Recall	F1-Score	Support
Attack	0.98	0.99	0.99	12957
Normal	1.00	1.00	1.00	178508
Accuracy			1.00	191465
Macro Avg	0.99	0.99	0.99	191465
Weighted Avg	1.00	1.00	1.00	191465

The classification report summarizes the precision, recall, and F1 score for each class (normal and attack). It provides a comprehensive assessment of the model's classification performance for both classes. For the "Normal" class, precision, recall, and F1-score are all approximately 1. For the "Attack" class, precision is 0.98, while recall and F1-score are all approximately 0.99, indicating strong performance in both classes.

B. Multilabel Classification

1) LSTM

The LSTM model begins training with poor performance (high loss and low F1 score). However, the improvement over 10 epochs suggests that the model improves significantly during training.

Fig. 19. F1-score/loss curve of the LSTM model used in the multilabel classification task.

Training F1 score rising closely with validation F1 score indicates that the model is performing well not only on the data it was trained on but also on unseen validation data. The validation loss and training loss declining together while being close to each other indicates that the model is minimizing its error effectively during training, and the performance on the validation set is consistent with the training set. This suggests that the model is not overfitting to the training data, as it's not achieving significantly lower losses on the training set compared to the validation set.

The model attains a precision of 0.996 and a recall of 0.996 with a threshold of 0.382 calculated using the Precision-Recall Curve. The macro average F1 score on the test set is 0.98, indicating strong performance across all classes.

TABLE IV. CLASSIFICATION REPORT FOR THE LSTM MODEL USED IN THE MULTILABEL CLASSIFICATION TASK

	Precision	Recall	F1-Score	Support
AIT-202	1.00	0.90	0.95	42
AIT-402	1.00	1.00	1.00	54
AIT-502	1.00	0.96	0.98	225
AIT-504	1.00	0.97	0.98	143
DPIT-301	1.00	1.00	1.00	336
FIT-401	1.00	1.00	1.00	300
FIT-502	1.00	1.00	1.00	91
LIT-101	1.00	1.00	1.00	642
LIT-301	0.99	1.00	0.99	873
LIT-401	1.00	1.00	1.00	539
MV-101	1.00	0.99	1.00	365
MV-201	1.00	1.00	1.00	216
MV-302	1.00	1.00	1.00	133
MV-303	0.92	1.00	0.96	231
MV-304	0.88	0.61	0.72	36
MV-504	0.84	1.00	0.92	76

P-101	1.00	0.98	0.99	620
P-102	1.00	0.98	0.99	175
P-201	1.00	1.00	1.00	20
P-203	1.00	1.00	1.00	89
P-205	1.00	1.00	1.00	89
P-302	1.00	1.00	1.00	7167
P-401	1.00	1.00	1.00	125
P-501	1.00	1.00	1.00	189
P-602	1.00	1.00	1.00	133
UV-401	1.00	1.00	1.00	98
Micro Avg	1.00	1.00	1.00	13007
Macro Avg	0.99	0.98	0.98	13007
Weighted Avg	1.00	1.00	1.00	13007
Samples Avg	1.00	1.00	1.00	13007

2) 1-D CNN

The 1-D CNN model starts training with strong performance right from the beginning. Despite this advantage, the curves indicate that this model has less room for improvement compared to the other models. However, the slow improvement during training still signifies that the model refines its performance further, albeit from the high starting point.

When both F1 curves start high and rise slightly over 10 epochs, while both loss curves start low and decline slightly, it suggests the model initially performs well and continues to improve without significant overfitting. High initial F1 scores indicate good performance, while gradual improvements suggest ongoing learning. Low initial losses and slight decreases indicate effective error minimization without over-optimization. Overall, the model maintains a good balance between fitting the training data and generalizing to unseen data.

The model attains a precision of 0.999 and recall of 0.999 with a threshold of 0.540 calculated using the Precision-Recall Curve. The macro average F1 score on the test set is 0.99, indicating strong performance across all classes.

Fig. 20. F1-score/loss curve of the 1-D CNN model used in the multilabel classification task.

TABLE V. CLASSIFICATION REPORT FOR THE 1-D CNN MODEL USED IN THE MULTILABEL CLASSIFICATION TASK

	Precision	**Recall**	**F1-Score**	**Support**
AIT-202	1.00	0.90	0.95	42
AIT-402	1.00	1.00	1.00	54
AIT-502	1.00	1.00	1.00	225
AIT-504	1.00	0.97	0.99	143
DPIT-301	1.00	1.00	1.00	336
FIT-401	1.00	1.00	1.00	300
FIT-502	1.00	1.00	1.00	91
LIT-101	1.00	1.00	1.00	642
LIT-301	0.98	1.00	0.99	873
LIT-401	1.00	1.00	1.00	539
MV-101	1.00	1.00	1.00	365
MV-201	1.00	1.00	1.00	216
MV-302	1.00	1.00	1.00	133
MV-303	1.00	1.00	1.00	231
MV-304	0.97	1.00	0.99	36
MV-504	0.99	1.00	0.99	76
P-101	1.00	1.00	1.00	620
P-102	1.00	0.98	0.99	175
P-201	1.00	1.00	1.00	20
P-203	1.00	0.99	0.99	89
P-205	1.00	0.99	0.99	89
P-302	1.00	1.00	1.00	7167
P-401	1.00	1.00	1.00	125
P-501	1.00	1.00	1.00	189
P-602	1.00	1.00	1.00	133
UV-401	1.00	1.00	1.00	98
Micro Avg	1.00	1.00	1.00	13007
Macro Avg	1.00	0.99	1.00	13007
Weighted Avg	1.00	1.00	1.00	13007
Samples Avg	1.00	1.00	1.00	13007

3) CNN-LSTM

The model begins with moderate performance. Despite this moderate start, the curves suggest that the model continues to learn and slowly improve over time. This implies that the model effectively leverages the data and training process to enhance its performance. Overall, the model maintains a good balance between fitting the training data and generalizing to unseen data.

Fig. 21. F1-score/loss curve of the CNN-LSTM model used in the multilabel classification task.

TABLE VI. CLASSIFICATION REPORT FOR THE CNN-LSTM MODEL USED IN THE MULTILABEL CLASSIFICATION TASK

	Precision	**Recall**	**F1-Score**	**Support**
AIT-202	1.00	0.90	0.95	42
AIT-402	1.00	1.00	1.00	54
AIT-502	0.97	1.00	0.98	225
AIT-504	1.00	0.98	0.99	143
DPIT-301	0.97	1.00	0.98	336
FIT-401	1.00	0.99	0.99	300
FIT-502	0.99	1.00	0.99	91

LIT-101	1.00	0.97	0.98	642
LIT-301	0.96	0.99	0.98	873
LIT-401	0.97	0.99	0.98	539
MV-101	0.99	0.99	0.99	365
MV-201	1.00	1.00	1.00	216
MV-302	0.99	0.99	0.99	133
MV-303	1.00	0.98	0.99	231
MV-304	0.92	1.00	0.96	36
MV-504	0.95	1.00	0.97	76
P-101	1.00	0.94	0.97	620
P-102	1.00	0.97	0.99	175
P-201	1.00	1.00	1.00	20
P-203	0.96	1.00	0.98	89
P-205	0.96	1.00	0.98	89
P-302	1.00	1.00	1.00	7167
P-401	1.00	1.00	1.00	125
P-501	1.00	0.99	0.99	189
P-602	0.99	0.99	0.99	133
UV-401	1.00	1.00	1.00	98
Micro Avg	0.99	0.99	0.99	13007
Macro Avg	0.98	0.99	0.99	13007
Weighted Avg	0.99	0.99	0.99	13007
Samples Avg	0.99	0.99	0.99	13007

The model attains a precision of 0.993 and recall of 0.993 with a threshold of 0.446 calculated using the Precision-Recall Curve. The macro average F1 score on the test set is 0.99, indicating strong performance across all classes.

4) MLP

Fig. 22. F1-score/loss curve of the MLP model used in the multilabel classification task.

The model begins with moderate performance. Despite this moderate start, the curves suggest that the model continues to learn and slowly improve over time. This implies that the model effectively leverages the data and training process to enhance its performance. Overall, the model maintains a good balance between fitting the training data and generalizing to unseen data.

TABLE VII. CLASSIFICATION REPORT FOR THE MLP MODEL USED IN THE MULTILABEL CLASSIFICATION TASK

	Precision	**Recall**	**F1-Score**	**Support**
AIT-202	1.00	0.90	0.95	42
AIT-402	1.00	1.00	1.00	54
AIT-502	1.00	1.00	1.00	225
AIT-504	1.00	0.97	0.98	143
DPIT-301	1.00	1.00	1.00	336
FIT-401	1.00	1.00	1.00	300
FIT-502	1.00	1.00	1.00	91
LIT-101	1.00	1.00	1.00	642
LIT-301	0.99	1.00	0.99	873
LIT-401	1.00	1.00	1.00	539
MV-101	1.00	1.00	1.00	365
MV-201	1.00	1.00	1.00	216
MV-302	1.00	1.00	1.00	133
MV-303	0.99	1.00	1.00	231
MV-304	0.92	1.00	0.96	36
MV-504	1.00	1.00	1.00	76
P-101	1.00	1.00	1.00	620
P-102	1.00	0.98	0.99	175
P-201	1.00	1.00	1.00	20
P-203	1.00	0.99	0.99	89
P-205	1.00	0.99	0.99	89
P-302	1.00	1.00	1.00	7167
P-401	1.00	1.00	1.00	125
P-501	1.00	1.00	1.00	189
P-602	1.00	1.00	1.00	133
UV-401	1.00	1.00	1.00	98
Micro Avg	1.00	1.00	1.00	13007
Macro Avg	1.00	0.99	0.99	13007
Weighted Avg	1.00	1.00	1.00	13007
Samples Avg	1.00	1.00	1.00	13007

The model attains a precision of 0.999 and a recall of 0.999 with a threshold of 0.375 calculated using the Precision-Recall Curve. The macro average F1 score on the test set is 0.99, indicating strong performance across all classes.

V. DISCUSSION

A. Binary Classification

This section provides a comparative analysis of the different deep learning models in terms of their complexity, parameter count, model size, training/inference time, and evaluation metrics.

The LSTM (Long Short-Term Memory) model is a type of recurrent neural network designed for sequence modeling. It is relatively simpler in architecture compared to CNNs (Convolutional Neural Networks) but is effective in handling sequential data. On the other hand, CNNs are well-suited for spatial data like images due to their ability to capture local patterns through convolutional and pooling layers. However, CNNs are more complex than LSTMs when applied to sequential data. The CNN-LSTM model combines the strengths of both CNNs and LSTMs, integrating both spatial and sequential processing. As a result, it has a more complex architecture than either LSTM or CNN alone.

Regarding parameter count, the LSTM model has 42,178 parameters, the CNN model has 49,506 parameters, and the CNN-LSTM model has 46,818 parameters. In terms of model size, the LSTM model is the smallest at 164.76 KB, followed by the CNN-LSTM model at 182.88 KB and the CNN model at 193.38 KB.

When it comes to training time, the CNN-LSTM model has longer training times compared to LSTM and CNN because of the combination of both architectures. Specifically, the CNN model has the shortest training time per epoch at approximately 85 seconds, the LSTM model has a slightly longer training time at around 125 seconds, and the CNN-LSTM model has the longest training time per epoch at approximately 175 seconds. It is worth noting that the Training Time is calculated as an average per epoch, and each model was trained for 10 epochs, whereas Inference Time is calculated for the entire test subset of the data, which is 20%

of the dataset. For example, the LSTM model takes 12 seconds to produce classification probabilities for the entire test subset of the dataset.

TABLE VIII. COMPARISON OF THE BINARY CLASSIFICATION MODELS

Model	Parameter Count	Size (KB)	Training Time* (sec/epoch)	Inference Time** (sec)
LSTM	42,178	164.76	125	12
CNN	49,506	193.38	85	8
CNN-LSTM	46,818	182.88	175	10

As for the evaluation metrics, the LSTM model achieves an accuracy of 0.998, indicating that it correctly classifies 99.8% of instances. With a precision of 0.990, the model makes very few false positive predictions, ensuring that when it predicts an instance as an attack, it is highly likely to be correct. The recall score of 0.991 indicates that the model captures 99.1% of actual attack instances, minimizing false negative predictions. The F1 score of 0.991, which is the harmonic mean of precision and recall, confirms the overall effectiveness of the model in balancing both metrics.

The CNN model achieves a slightly lower accuracy of 0.997 compared to the LSTM model but still demonstrates excellent performance. Its precision score of 0.990 indicates a high proportion of true positive predictions relative to all positive predictions made by the model. With a recall score of 0.987, the model effectively identifies the majority of actual attack instances, minimizing false negative predictions. The F1 score of 0.989 suggests a strong balance between precision and recall, showcasing the model's robustness in classification.

Similar to the CNN model, the CNN-LSTM model achieves an accuracy of 0.997, indicating its ability to classify instances accurately. Its precision score of 0.994 surpasses both the LSTM and CNN models, demonstrating a superior ability to make correct positive predictions. With a recall score of 0.992, the model effectively captures the majority of actual attack instances, minimizing false negatives. The F1 score of 0.993 confirms the model's exceptional balance between precision and recall, highlighting its overall effectiveness in classification.

All three models exhibit outstanding performance in distinguishing between normal and attack instances, with the CNN-LSTM model slightly outperforming the LSTM and CNN models in terms of precision and F1 score. These results indicate the efficacy of deep learning techniques in cybersecurity applications, particularly in detecting and mitigating cyber threats.

TABLE IX. COMPARISON OF THE BINARY CLASSIFICATION MODELS' EVALUATION METRICS

Model	Accuracy	Macro Recall	Macro Precision	Macro F1-score	Cohen's Kappa
LSTM	99.8%	99.1%	99.0%	99.1%	0.981
CNN	99.7%	98.7%	99.0%	98.9%	0.977
CNN-LSTM	99.7%	99.2%	99.4%	99.3%	0.986

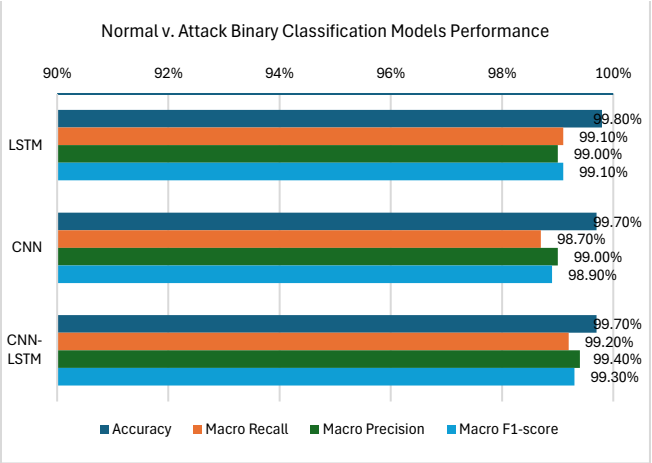

Fig. 23. Plot of the binary classification models' evaluation metrics.

B. Multilabel Classification

When considering all three multilabel classification models (LSTM, CNN, and CNN-LSTM), they demonstrate high performance across the evaluated metrics. Based on the classification reports provided for the LSTM, CNN, and CNN-LSTM, several insights can be drawn. All four models achieved high precision, recall, and F1 scores across the micro, macro, weighted, and sample averages. This consistency suggests robust performance across different evaluation metrics. While micro-averaged metrics consider each instance equally, macro-averaged metrics treat all classes equally. In most cases, micro-averaged scores were slightly higher, indicating a balanced performance across all classes.

The CNN model achieved slightly higher scores in the macro-average precision and F1-score compared to the other models. This suggests that the CNN model performs particularly well when considering class-wise performance equally important. The CNN-LSTM model demonstrated balanced performance across precision, recall, and F1-score, with slightly lower scores compared to LSTM and CNN models. This indicates that the combined architecture of CNN-LSTM effectively captures both spatial and temporal dependencies in the data. The MLP model's performance was comparable to the LSTM, CNN, and CNN-LSTM models across all evaluated metrics. MLPs are known for their simplicity and flexibility, making them a viable option for various classification tasks. We notice that in this case, the simpler the model is, the better it performs, which can be attributed to the fact that complex models like LSTMs and CNNs may have a higher risk of overfitting in the presence of limited data as they have a large number of parameters that can capture noise in the training data.

The classification models discussed—LSTM, CNN, CNN-LSTM, and MLP—may face limitations in generalization due to several factors:

1. With a low number of samples, the models may struggle to capture the full complexity and variability of the underlying data distribution. This limitation can lead to overfitting, where the model performs well on the training data but fails to generalize to unseen data.
2. The imbalance in the number of samples across classes can bias the models towards the majority classes, leading to suboptimal performance for minority classes.

3. In the presence of limited data, complex models like LSTMs and CNNs may have a higher risk of overfitting, as they have a large number of parameters that can capture noise in the training data. Overfitting can hinder the model's ability to generalize to new, unseen examples.

4. With a low number of samples, the models may struggle to learn robust representations of the data, which can affect the models' ability to extract meaningful features and patterns, especially in complex datasets with high-dimensional inputs.

5. The scarcity of training samples can impact the models' generalization performance, making them less effective at recognizing patterns in unseen data, which is particularly concerning in real-world applications where the model needs to perform well on data that was not present during training.

To mitigate these limitations, techniques such as regularization and ensemble methods (Voting Classifier) are employed. However, collecting more labeled data, especially for underrepresented classes, is the best technique to improve the models' ability to generalize and make more accurate predictions.

Therefore, a majority voting classifier is created to combine the predictions of four models: LSTM, CNN, MLP, and CNN-LSTM. The voting classifier iterates through each element of the predictions made by the LSTM model, CNN model, MLP model, and CNN-LSTM model. For each element in the predictions, a majority voting scheme is employed. The classifier sums up the binarized predictions from all four models for that particular element and checks if the sum is greater than or equal to 3. If it is, the majority vote for this class is (1); otherwise, it is (0).

Combining predictions from multiple models can help mitigate the weaknesses of individual models and lead to more robust overall predictions. Additionally, ensemble methods like majority voting often lead to better generalization performance by reducing overfitting and capturing more diverse patterns in the data. By aggregating predictions from multiple models, the variance in predictions can be reduced, leading to more stable and reliable outcomes. In many cases, ensemble methods such as majority voting can outperform individual models by leveraging the collective intelligence of multiple models. While a voting classifier can improve model performance and robustness, it comes with drawbacks. Notably, it increases both training and inference times due to the need to process predictions from individual models. This extra computation can strain resources and is especially undesirable in real-time applications.

TABLE X. COMPARISON OF THE BINARY CLASSIFICATION MODELS' EVALUATION METRICS

Model	Accuracy	Macro Recall	Macro Precision	Macro F1-score
LSTM	99.5%	97.7%	98.6%	98.0%
CNN	99.7%	99.3%	99.8%	99.5%
CNN-LSTM	99.0%	98.8%	98.5%	98.6%
MLP	99.8%	99.3%	99.6%	99.5%
Majority Voting	99.7%	99.2%	99.9%	99.6%

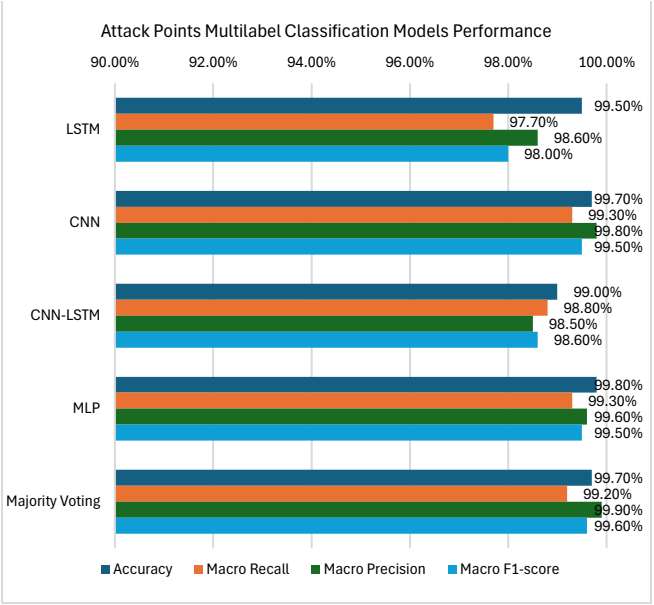

Fig. 24. Plot of the multilabel classification models' evaluation metrics.

The results showcase the impressive performance of each individual model, with accuracies ranging from 99.0% to 99.8%. Notably, the CNN and MLP models demonstrate the highest Macro F1 score at 99.5%. These high accuracies reflect the models' ability to correctly classify instances across various classes.

In terms of recall, precision, and F1 score, all models exhibit strong performance, with scores consistently above 97.70%, 98.50%, and 98.00%, respectively. This indicates that the models effectively balance between minimizing false negatives (recall) and false positives (precision), resulting in robust classification capabilities.

Considering model efficiency, the MLP model stands out with the lowest parameter count and size, implying a more compact and potentially faster model. However, the CNN model, despite having a larger parameter count and size, achieves comparable performance while maintaining a relatively low training time of 8 seconds per epoch.

The majority voting approach further boosts overall performance, achieving a Macro F1-score of 99.60% and outperforming individual models. However, it comes at the cost of increased training time, with each epoch requiring 39 seconds due to the need to aggregate predictions from multiple models.

In summary, while each model demonstrates exceptional performance individually, the majority voting ensemble enhances overall accuracy and reliability. However, practitioners must consider the trade-off between improved performance and increased computational overhead (especially training time and inference time) when employing ensemble methods like majority voting.

VI. CONCLUSION

This study aims to advance the field of cybersecurity for Secure Water Treatment (SWaT) systems and, by extension, critical Industrial Control Systems (ICS). Through the application of deep learning techniques through binary and multi-label classification approaches, we have demonstrated the efficacy of these methods in detecting and classifying cyberattacks on SWaT systems. Our research leveraged the

SWaT 2015 dataset to develop and evaluate a range of deep learning models, including Long Short-Term Memory (LSTM), Convolutional Neural Network (CNN), hybrid CNN-LSTM, and Multilayer Perceptron (MLP) architectures. We employed a two-phase approach, beginning with binary classification to distinguish between normal operational states and cyberattacks, followed by multi-label classification to identify specific attack points within the system. The results of our study are highly promising:

1. In the binary classification task, our models achieved remarkable performance, with macro F1-scores ranging from 98.9% to 99.3% across different architectures. The hybrid CNN-LSTM model demonstrated particularly strong results, highlighting its potential for robust cyber threat detection.
2. The multi-label classification phase further enhanced our understanding of cyber threats by accurately identifying attack points. Our models achieved macro F1-scores between 98.0% and 99.5%, with a majority voting ensemble technique enhancing it to 99.6%.

These findings underscore the transformative potential of deep learning approaches in safeguarding critical ICS against cyber threats. By accurately detecting and classifying abnormal states induced by cyberattacks, we contribute to the enhanced resilience and security of SWaT systems, ultimately ensuring the continued provision of safe and reliable water services to communities worldwide. The significance of this research extends beyond SWaT systems, offering valuable insights for critical infrastructure protection. Our work demonstrates the potential of deep learning techniques to reinforce the cybersecurity of various ICS sectors, thereby safeguarding public health and safety.

While our study has yielded promising results, we acknowledge certain limitations and areas for future exploration. Data scarcity, class imbalance, and the need for real-world validation in SWaT environments pose significant challenges to model generalization and performance. Moving forward, we recommend several directions for future research: exploring model enhancements such as integrating anomaly detection techniques and adapting to evolving threats; investigating scalability and applicability to other critical ICS sectors; addressing data-related challenges through innovative augmentation or transfer learning approaches; and conducting extensive real-world testing. Additionally, the potential of ensemble methods, as demonstrated by our majority voting approach, presents opportunities for further performance improvements and increased reliability in operational settings. By addressing these areas, we can continue to advance the field of cybersecurity for SWaT systems and critical infrastructure protection.

In conclusion, this research paper represents a contribution to the field of cybersecurity for critical ICS, particularly in SWaT systems. By leveraging advanced deep learning techniques and following a two-step classification process, we have demonstrated the potential to strengthen the security measures of vital infrastructure against emerging cyber threats.

REFERENCES

[1] A. P. Mathur and N. O. Tippenhauer, "SWaT: a water treatment testbed for research and training on ICS security," in *2016 International Workshop on Cyber-physical Systems for Smart Water Networks (CySWater)*, Apr. 2016, pp. 31–36. doi: 10.1109/CySWater.2016.7469060.

[2] B. Brentan, P. Rezende, D. Barros, G. Meirelles, E. Luvizotto, and J. Izquierdo, "Cyber-Attack Detection in Water Distribution Systems Based on Blind Sources Separation Technique," *Water*, vol. 13, no. 6, p. 795, Mar. 2021, doi: 10.3390/w13060795.

[3] H. Wen, "Vulnerability Assessment of Industrial Control System with an Improved CVSS," *arXiv preprint arXiv:2306.08631*, 2023.

[4] O. Tushkanova, D. Levshun, A. Branitskiy, E. Fedorchenko, E. Novikova, and I. Kotenko, "Detection of cyberattacks and anomalies in cyber-physical systems: Approaches, data sources, evaluation," *Algorithms*, vol. 16, no. 2, p. 85, 2023.

[5] C. M. Ahmed, G. R. MR, and A. P. Mathur, "Challenges in machine learning based approaches for real-time anomaly detection in industrial control systems," in *Proceedings of the 6th ACM on cyber-physical system security workshop*, 2020, pp. 23–29.

[6] J. Inoue, Y. Yamagata, Y. Chen, C. M. Poskitt, and J. Sun, "Anomaly detection for a water treatment system using unsupervised machine learning," in *2017 IEEE international conference on data mining workshops (ICDMW)*, IEEE, 2017, pp. 1058–1065.

[7] F. Xue, W. Yan, T. Wang, H. Huang, and B. Feng, "Detecting attacks on a water treatment system using oneclass support vector machines," in *Advances in Digital Forensics XVI: 16th IFIP WG 11.9 International Conference, New Delhi, India, January 6–8, 2020, Revised Selected Papers 16*, Springer, 2020, pp. 95–108.

[8] M. Abdelaty, R. Doriguzzi-Corin, and D. Siracusa, "DAICS: A deep learning solution for anomaly detection in industrial control systems," *IEEE Transactions on Emerging Topics in Computing*, vol. 10, no. 2, pp. 1117–1129, 2021.

[9] C. Feng, T. Li, and D. Chana, "Multi-level anomaly detection in industrial control systems via package signatures and LSTM networks," in *2017 47th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, IEEE, 2017, pp. 261–272.

[10] M. Kravchik and A. Shabtai, "Detecting cyber attacks in industrial control systems using convolutional neural networks," in *Proceedings of the 2018 workshop on cyber-physical systems security and privacy*, 2018, pp. 72–83.

[11] S. Sapkota, A. N. Mehdy, S. Reese, and H. Mehrpouyan, "Falcon: Framework for anomaly detection in industrial control systems," *Electronics*, vol. 9, no. 8, p. 1192, 2020.

[12] G. Lee, Y. Yoon, and K. Lee, "Anomaly Detection Using an Ensemble of Multi-Point LSTMs," *Entropy*, vol. 25, no. 11, p. 1480, 2023.

[13] N. X. Hoang, N. V. Hoang, N. H. Du, T. T. Huong, K. P. Tran, and others, "Explainable anomaly detection for industrial control system cybersecurity," *IFAC-PapersOnLine*, vol. 55, no. 10, pp. 1183–1188, 2022.

[14] W. Zhang, C. Zhang, and F. Tsung, "GRELEN: Multivariate Time Series Anomaly Detection from the Perspective of Graph Relational Learning.," in *IJCAI*, 2022, pp. 2390–2397.

[15] G. Koutroulis, B. Mutlu, and R. Kern, "A causality-inspired approach for anomaly detection in a water treatment testbed," *Sensors*, vol. 23, no. 1, p. 257, 2022.

[16] A. Abid, F. Jemili, and O. Korbaa, "Distributed deep learning approach for intrusion detection system in industrial control systems based on big data technique and transfer learning," *Journal of Information and Telecommunication*, vol. 7, no. 4, pp. 513–541, 2023.

[17] S. Adepu and A. Mathur, "Distributed attack detection in a water treatment plant: Method and case study," *IEEE Transactions on Dependable and Secure Computing*, vol. 18, no. 1, pp. 86–99, 2018.

[18] E. A. Boateng, J. Bruce, and D. A. Talbert, "Anomaly detection for a water treatment system based on one-class neural network," *IEEE access*, vol. 10, pp. 115179–115191, 2022.

[19] X. Yang, E. Howley, and M. Schukat, "ADT: Time series anomaly detection for cyber-physical systems via deep reinforcement learning," *Computers & Security*, vol. 141, p. 103825, 2024.

[20] A. Kumar, N. Saxena, S. Jung, and B. Choi, "Improving Detection of False Data Injection Attacks Using Machine Learning with Feature Selection and Oversampling. Energies 2022, 15, 212," *Machine Learning and Data Mining Applications in Power Systems*, p. 283, 2021.

[21] M. Pordelkhaki, S. Fouad, and M. Josephs, "Intrusion Detection for Industrial Control Systems by Machine Learning using Privileged Information," in *2021 IEEE International Conference on*

Intelligence and Security Informatics (ISI), Nov. 2021, pp. 1–6. doi: 10.1109/ISI53945.2021.9624757.

[22] 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. doi: 10.1007/978-3-319-71368-7_8.

[23] K. Potdar, T. S., and C. D., "A Comparative Study of Categorical Variable Encoding Techniques for Neural Network Classifiers," *IJCA*, vol. 175, no. 4, pp. 7–9, Oct. 2017, doi: 10.5120/ijca2017915495.

[24] A. N. Tarekegn, M. Ullah, and F. A. Cheikh, "Deep Learning for Multi-Label Learning: A Comprehensive Survey." arXiv, Jun. 25, 2024. doi: 10.48550/arXiv.2401.16549.

[25] Ž. Đ. Vujovic, "Classification Model Evaluation Metrics," *International Journal of Advanced Computer Science and Applications (IJACSA)*, vol. 12, no. 6, Art. no. 6, 30 2021, doi: 10.14569/IJACSA.2021.0120670.

[26] D. M. W. Powers, "Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation." arXiv, Oct. 10, 2020. doi: 10.48550/arXiv.2010.16061.

[27] S. Raschka, "An Overview of General Performance Metrics of Binary Classifier Systems." arXiv, Oct. 16, 2014. doi: 10.48550/arXiv.1410.5330.

[28] M. Fatourechi, R. K. Ward, S. G. Mason, J. Huggins, A. Schlögl, and G. E. Birch, "Comparison of Evaluation Metrics in Classification Applications with Imbalanced Datasets: International Conference on Machine Learning and Applications," *International Conference on Machine Learning and Applications*, 2008, Accessed: Jul. 08, 2024. [Online]. Available: http://www.icmla-conference.org/icmla08/

[29] C. Nwankpa, W. Ijomah, A. Gachagan, and S. Marshall, "Activation Functions: Comparison of trends in Practice and Research for Deep Learning." arXiv, Nov. 08, 2018. doi: 10.48550/arXiv.1811.03378.

[30] A. F. Agarap, "Deep Learning using Rectified Linear Units (ReLU)." arXiv, Feb. 07, 2019. doi: 10.48550/arXiv.1803.08375.

[31] J. Terven, D. M. Cordova-Esparza, A. Ramirez-Pedraza, and E. A. Chavez-Urbiola, "Loss Functions and Metrics in Deep Learning." arXiv, Sep. 06, 2023. doi: 10.48550/arXiv.2307.02694.

[32] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization." arXiv, Jan. 29, 2017. doi: 10.48550/arXiv.1412.6980.

[33] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997, doi: 10.1162/neco.1997.9.8.1735.

[34] S. Kiranyaz, O. Avci, O. Abdeljaber, T. Ince, M. Gabbouj, and D. J. Inman, "1D Convolutional Neural Networks and Applications: A Survey." arXiv, May 09, 2019. doi: 10.48550/arXiv.1905.03554.

[35] "A Hybrid Lightweight 1D CNN-LSTM Architecture for Automated ECG Beat-Wise Classification | IIETA." Accessed: Jul. 08, 2024. [Online]. Available: https://www.iieta.org/journals/ts/paper/10.18280/ts.380503

[36] H. Taud and J. F. Mas, "Multilayer Perceptron (MLP)," in *Geomatic Approaches for Modeling Land Change Scenarios*, M. T. Camacho Olmedo, M. Paegelow, J.-F. Mas, and F. Escobar, Eds., Cham: Springer International Publishing, 2018, pp. 451–455. doi: 10.1007/978-3-319-60801-3_27.