JETIR.ORG ISSN: 2349-5162 | ESTD Year : 2014 | Monthly Issue
JOURNAL OF EMERGING TECHNOLOGIES AND INNOVATIVE RESEARCH (JETIR)
An International Scholarly Open Access, Peer-reviewed, Refereed Journal

A Study on detecting and preventing threats using AI in IOT-based Smart Elevators

Mohamed Afsal Mohamed Rasheed, M I Jawid Nazir
Student, Associate Professor
School of Engineering & IT, School of Engineering & IT
Manipal Academy of Higher Education, Dubai Campus, Dubai, U.A.E

Abstract : The Internet of Things (IoT) has revolutionized industries by connecting a number of devices, ranging from everyday household appliances to critical infrastructure components. Significant security concerns have been introduced along with new opportunities as a result of this integration. The integration of Internet of Things (IoT) technologies into smart elevator systems has revolutionized automation, operational efficiency, and user experience. However, this increased connectivity also brings heightened cybersecurity risks, exposing systems to potential threats. This case study explores the application of Artificial Intelligence (AI), particularly machine learning techniques, to detect and prevent such vulnerabilities in IoT-powered smart elevators. Using a telemetry dataset that simulates real-world elevator operations, the research implements LSTM Autoencoder and CNN LSTM model to identify anomalous behavior. The model demonstrates strong potential in enhancing real-time threat detection and initiating automated defense responses. Through simulations and scenario analysis, the study highlights the critical role of unsupervised learning, precise threshold tuning, and sequence modeling in building resilient and secure systems. Ultimately, the findings contribute to the growing body of research advocating for AI-driven cybersecurity frameworks within increasingly connected IoT ecosystems.

IndexTerms **- IoT, Smart Elevator, AI, ML Technique, Cybersecurity, Threat Detection, LSTM, Autoencoder, CNN**

I. INTRODUCTION

In recent years, smart elevators have become a key component of modern buildings, especially in smart cities and high-rise infrastructures. These elevators go beyond traditional mechanical systems by incorporating Internet of Things (IoT) technology. This integration allows elevators to offer advanced features such as automated floor selection, personalized user experiences, and real-time system monitoring. For example, a smart elevator can recognize a user through a mobile app or access card and automatically take them to their usual floor, or notify maintenance teams if a component is malfunctioning.

These capabilities are made possible through a network of sensors, microcontrollers, and communication modules embedded within the elevator system. These components collect and transmit data to cloud platforms or mobile applications, enabling remote access, diagnostics, and control. While this connectivity greatly improves operational efficiency and user convenience, it also introduces cybersecurity vulnerabilities. Because smart elevators are connected to the internet and other building systems, they can become targets for unauthorized access, data breaches, or even malicious cyberattacks that could disrupt operations or compromise user safety.

Traditional cybersecurity measures, such as firewalls or rule-based intrusion detection systems, often struggle to keep up with evolving threats—especially those that are new or previously unknown. This is where Artificial Intelligence (AI), particularly machine learning (ML), becomes essential. AI-based systems can learn from data patterns and detect unusual behaviour that may indicate a security threat or system failure.

In this context, anomaly detection models - especially those using unsupervised learning - play a critical role. These models analyse telemetry data (such as elevator movement, door activity, or sensor readings) to identify deviations from normal behaviour. For instance, if an elevator suddenly stops at unexpected floors or shows irregular usage patterns, the AI model can flag this as a potential issue. One powerful approach is the use of Long Short-Term Memory (LSTM) Autoencoders, which are capable of understanding sequences of data over time and detecting subtle anomalies. And using a hybrid approach CNN LSTM model where CNN extract spatial features and LSTM learns temporal dependencies. By combining AI with IoT, smart elevator systems can become not only more efficient but also more resilient and secure, capable of proactively identifying and responding to threats before they cause harm.

The integration of AI-based anomaly detection in smart elevator systems has broad applicability across various domains:

- Smart Cities: In urban environments where buildings, transportation, and utilities are interconnected, smart elevators contribute to efficient vertical mobility. AI-enhanced security ensures these systems remain reliable and safe for public use.

- Hospitals and Healthcare Facilities: Elevators in hospitals often transport patients, medical staff, and sensitive equipment. Ensuring their secure and uninterrupted operation is critical for patient safety and operational efficiency.

- Corporate and Government Buildings: High-security environments benefit from AI-driven monitoring to prevent unauthorized access and ensure compliance with safety protocols.

- Residential Complexes: In smart homes and apartment buildings, AI can personalize elevator usage while maintaining privacy and security.

- Airports and Transit Hubs: These high-traffic areas require robust systems that can detect and respond to anomalies in real time to avoid disruptions and ensure passenger safety.

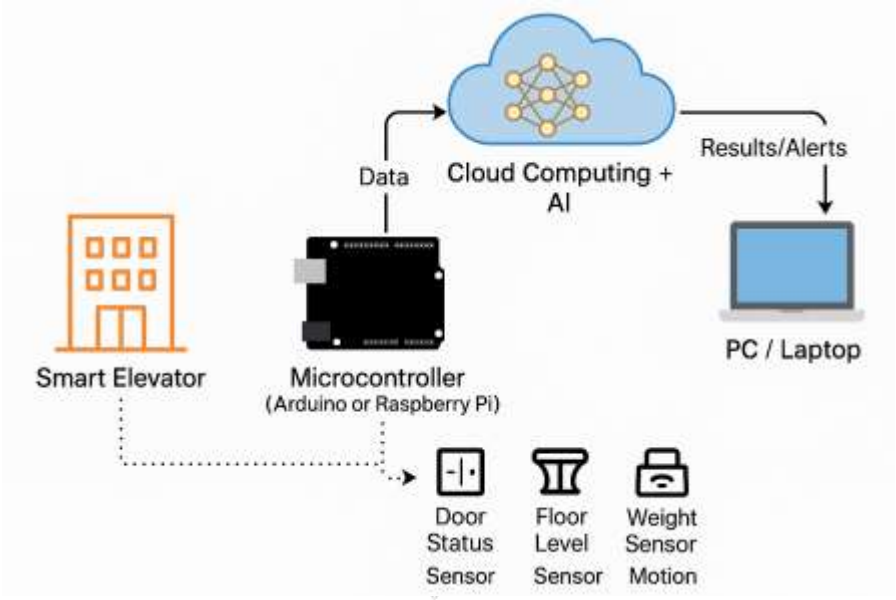


Fig1: Smart Elevator Function

II. LITERATURE REVIEW

The literature on AI-based cybersecurity systems in IoT environments has grown significantly in recent years. Smart elevator systems, being a part of smart infrastructure, are exposed to multiple vectors of attack, including remote hijacking, unauthorized access, DOS, spoofing, and malware injection.

Research by Alrawais et al. (2017) emphasized that IoT networks are particularly vulnerable due to their limited computational resources and heterogeneous nature. This makes them difficult to secure using conventional firewalls or antivirus mechanisms. As a result, machine learning (ML) and deep learning (DL) techniques have emerged as promising solutions for detecting unknown or zero-day threats by learning patterns of normal and anomalous behavior.

Studies focusing on anomaly detection in cyber-physical systems have shown that sequence-aware models like Long Short-Term Memory (LSTM) networks outperform traditional models, especially when analyzing time-series telemetry data. LSTM Autoencoders, in particular, have been explored in domains such as industrial control systems (ICS), smart homes, and autonomous vehicles. These models reconstruct input sequences and measure reconstruction error to infer anomalies.

The TON_IoT dataset developed by the UNSW Canberra Cyber group is a benchmark dataset designed to simulate telemetry data across multiple IoT devices including smart fridges, garage doors, motion sensors, and lights. Several recent studies have utilized this dataset for testing the performance of intrusion detection systems (IDS), both in supervised and unsupervised learning contexts.

The literature also suggests that combining models (e.g., CNN-LSTM hybrids or ensemble methods) can improve performance. Additionally, visualizing anomaly scores and incorporating real-time data streaming enhances model applicability in production environments.

In conclusion, the review of existing literature reinforces that AI, particularly deep learning-based unsupervised models, holds substantial potential for securing smart elevators and other IoT infrastructure. However, model tuning, interpretability, and real-time deployment remain ongoing challenges.

2.1 Review of Key Research Papers

1. LSTM-based Encoder-Decoder for Multi-sensor Anomaly Detection (Malhotra et al., 2016)

Overview:

Malhotra et al. introduced an innovative approach to anomaly detection in multivariate time-series data using a Long Short-Term Memory (LSTM) based Encoder-Decoder architecture, termed EncDec-AD. This model is particularly adept at handling data from mechanical systems equipped with multiple sensors, where external factors can introduce unpredictability in the time-series data .

Methodology:

Encoder-Decoder Architecture: The model comprises two primary components: an encoder LSTM that processes the input sequence and compresses it into a fixed-length context vector, and a decoder LSTM that reconstructs the sequence from this vector.

Training on Normal Data: The model is trained exclusively on normal operational data, learning to reconstruct sequences that represent typical system behavior.

Anomaly Detection: During inference, the reconstruction error (difference between the input and output sequences) is calculated. Sequences with errors exceeding a predefined threshold are flagged as anomalies.

Experimental Validation:

The model was evaluated on various datasets, including:

Power Demand Dataset: Representing periodic time-series data.

Space Shuttle Dataset: Characterized by quasi-periodic patterns.

ECG Dataset: Containing quasi-periodic sequences.

Engine Datasets: Comprising both predictable and unpredictable behaviors.

The EncDec-AD model demonstrated robustness across these datasets, effectively detecting anomalies in both predictable and unpredictable time-series data .

Reference :

P. Malhotra, A. Ramakrishnan, G. Anand, L. Vig, P. Agarwal, and G. R. Shroff, “LSTM-based Encoder-Decoder for Multi-sensor Anomaly Detection,” Jul. 2016, doi: https://doi.org/10.48550/arxiv.1607.00148.

Relevance to Current Study:

In the context of smart elevators, which generate multivariate time-series data from various sensors (e.g., door status, weight sensors, motion detectors), the EncDec-AD model's ability to learn normal operational patterns and detect deviations makes it a suitable choice for unsupervised anomaly detection.

2. A Deep Learning Approach for Intrusion Detection Using Recurrent Neural Networks (Yin et al., 2017)

Overview:

Yin et al. proposed a deep learning framework for intrusion detection systems (IDS) utilizing Recurrent Neural Networks (RNNs), specifically focusing on Long Short-Term Memory (LSTM) networks. The study aimed to enhance the detection accuracy of various network attacks by capturing temporal dependencies in network traffic data .

Methodology:

RNN-Based IDS: The model employs LSTM networks to process sequences of network traffic data, capturing temporal patterns that are indicative of normal or malicious activities.

Binary and Multi-class Classification: The study explores both binary classification (normal vs. attack) and multi-class classification (identifying specific types of attacks).

Performance Evaluation: The model's performance was assessed using metrics such as accuracy, precision, recall, and F1-score.

Experimental Results:

The RNN-based IDS demonstrated superior performance compared to traditional machine learning algorithms, achieving higher accuracy and better generalization in detecting various types of network intrusions .

Reference:

C. Yin, Y. Zhu, J. Fei, and X. He, "A Deep Learning Approach for Intrusion Detection Using Recurrent Neural Networks," *IEEE Access*, vol. 5, pp. 21954–21961, 2017, doi: https://doi.org/10.1109/access.2017.2762418.

Relevance to Current Study:

For smart elevator systems connected to networks, ensuring cybersecurity is paramount. The application of LSTM networks for intrusion detection aligns with the need to monitor and analyze network traffic for potential threats, making this approach relevant for supervised multi-class classification of known cyber threats in smart elevator environments.

2.2 **Extant System Vs Prospective System**

Table 2.1: Extant System Vs Proposed System

Feature	Extant System	Proposed System
Detection Method	Rule-based or signature-based	AI-based (LSTM Autoencoder, CNN-LSTM)
Adaptability to New Threats	Low	High
Anomaly Detection	Limited to known patterns	Capable of detecting unseen behavioral deviations
Multi-class Threat Classification	Not supported	Supports classification of 8+ cyber threat types

Response Time	Manual or delayed	Real-time detection and response
Application Scope	Mostly commercial and residential buildings	Broader scope including smart cities, hospitals, airports, and high-security facilities
Technology Base	Traditional IoT-enabled elevators with basic automation and monitoring	AI-integrated smart elevators with advanced anomaly detection and adaptive learning
Data Handling	Basic telemetry logging; limited real-time analysis	Real-time telemetry analysis with feedback loops for continuous learning

2.3 **How AI can predict upcoming threats in IOT based smart elevator**

1. **Study and Analyze Existing Threats**

- Unauthorized access attempts.
- Network attacks like DDoS or man-in-the-middle.
- Sensor malfunctions or unusual behavior.
- Software bugs or abnormal commands.

2. **Classify Threats**

- Hardware-level threats (e.g., tampering with sensors)
- Software threats (e.g., malware in elevator control apps)
- Network threats (e.g., Wi-Fi hacking)
- User behavior anomalies (e.g., unauthorized person accessing admin settings)

3. **Prediction of Future Threats**

- Machine Learning Models (e.g., Decision Trees, SVM, Neural Networks)
- Anomaly Detection Algorithms
- Behavioral Pattern Analysis

4. **Real-time Monitoring and Notifications**

- Monitor the elevator system in real-time
- Detect suspicious activity instantly
- Send automatic alerts/notifications to admin or maintenance teams via: Mobile apps, Email, Dashboard alerts.

2.4 **Types of Threats**

Table 2.2: Types of Threats

Cybersecurity Threat	Real-World Example	Severity Level	Impact
Backdoor	Malware installs a hidden backdoor on an elevator control system to allow remote access	High	Unauthorized access; attacker may operate or sabotage elevator functions silently
DDoS	Attackers overwhelm the building's access control servers, denying entry to employees	Critical	Service disruption; prevents legitimate access, halts operations temporarily
DoS	A malicious command floods the elevator's control software, causing system reboot	High	Temporary system downtime; impacts service reliability and user safety
Injection	Hacker exploits a flaw in a maintenance web app to inject commands into elevator logic	Critical	Full control hijack; attacker could cause elevator to malfunction dangerously
MITM (Man-in-the-Middle)	Attacker intercepts data between elevator control unit and monitoring station	High	Data integrity breach; attacker could alter commands or leak sensitive data
Password Attack	Brute force used to crack admin credentials for the elevator's central system	Medium to High	Loss of control; enables unauthorized reconfiguration or shutdown
Ransomware	System displays a ransom note; elevators disabled until payment is made	Critical	Complete operational lockout; financial loss, potential panic in high-rises
XSS (Cross-Site Scripting)	Malicious script entered into elevator panel display via touchscreen interface	Medium	Unauthorized content display; possible phishing or social engineering risk

III. CASE STUDY

1. KONE Smart Elevators with AWS IoT

Table 3.1: KONE Case study

Aspect	Details
Data Collected	Real-time telemetry from 1.6M+ elevators: speed, vibration, door cycles, fault codes, and usage frequency.
Verification	Data is validated through AWS IoT Core with 99.9% provisioning success. Faults are cross-verified with technician reports and customer feedback.
Method Used	Predictive analytics using AWS cloud services. While not explicitly LSTM-based, the architecture supports integration with deep learning models.
Accuracy Calculation	Performance measured by reduction in incidents: 40% fewer entrapments, 70% proactive fault detection.
Graph Interpretations	AWS dashboards visualize fault trends, usage spikes, and predictive maintenance triggers.
Problem & Fix	Legacy systems lacked real-time insights. Solution: migrated to AWS IoT Core, enabling scalable, secure, and proactive maintenance via KONE 24/7 Connected Services.
Reference	KONE Unlocks New Efficiencies Using AWS IoT | KONE Case Study | AWS

Fig3.1 Architecture

2. Elevator Fault Prediction Using LSTM-AE + Attention

Table 3.2: Elevator Fault Prediction Case study

Aspect	Details
Data Collected	Elevator operation parameters: current, voltage, speed, and vibration signals.
Verification	Attribute Correlation Density Ranking (ACDR) for feature selection; TSO-VMD denoising to remove noise.
Method Used	VMD-BiLSTM-AEAM: Variational Mode Decomposition + Bidirectional LSTM + Autoencoder with Attention Mechanism.
Accuracy Calculation	Achieved 91.9% True Positive Rate (TPR), 0.090 False Positive Rate (FPR), and AUC of 0.919 with 95% confidence intervals.
Graph Interpretations	ROC curves, TPR/FPR plots, and denoised signal graphs illustrate model performance.
Problem & Fix	Noise and redundancy in sensor data reduced prediction accuracy. Solution: hybrid model with denoising and attention layers to enhance signal clarity and temporal learning.
Reference	https://doi.org/10.1371/journal.pone.0320566

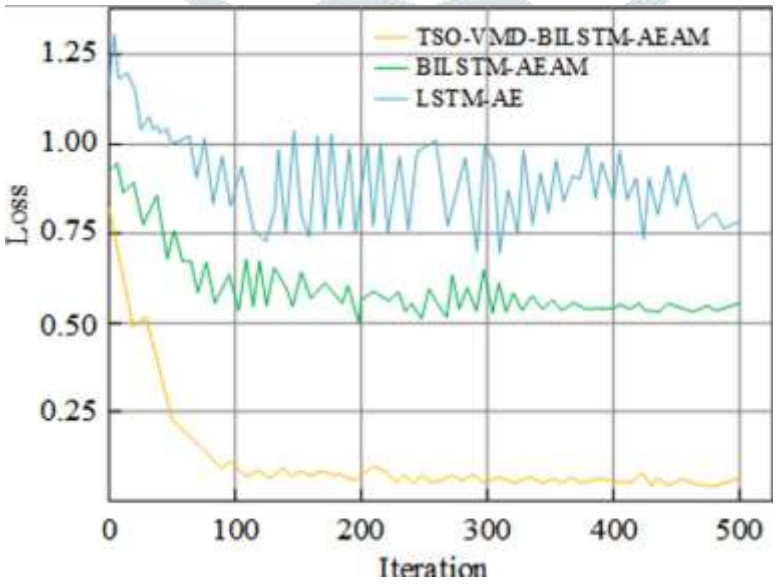
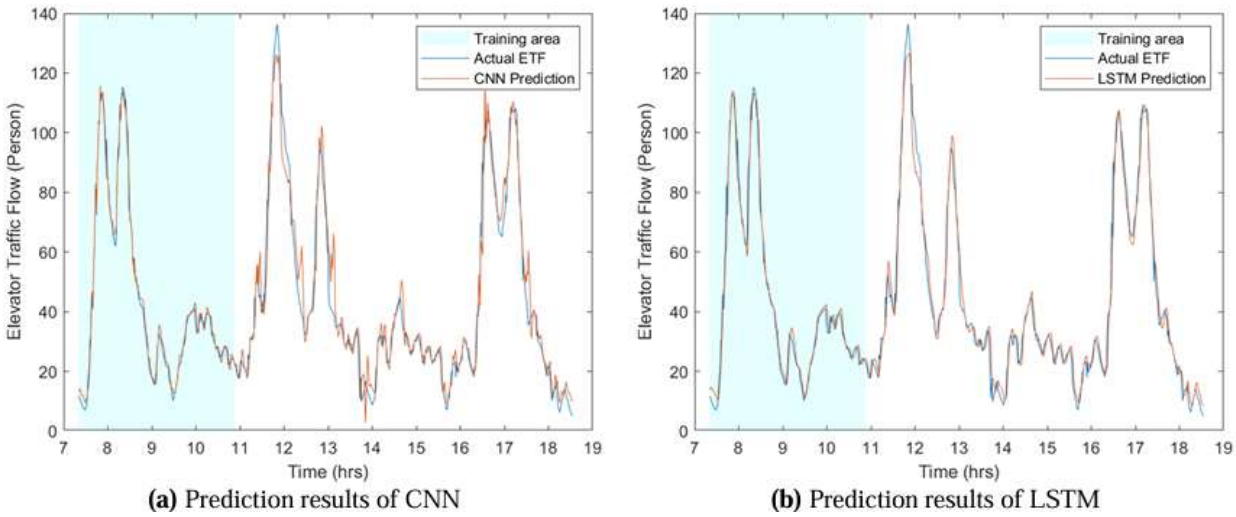


Fig 3.2 Comparison before and after improvement of LSTM Autoencoder.

3. **CNN vs. LSTM for Elevator Traffic Flow Prediction**

Table 3.3: Elevator Traffic Flow Prediction Case study

Aspect	**Details**
Data Collected	655 real-world elevator traffic data points from a commercial building.
Verification	Data preprocessed with normalization and time-series validation.
Method Used	Comparative study using CNN and LSTM models for traffic flow prediction.
Accuracy Calculation	LSTM outperformed CNN in temporal accuracy; metrics included RMSE and MAE across time intervals.
Graph Interpretations	Time-series prediction vs. actual traffic flow graphs; error distribution plots.
Problem & Fix	Static scheduling led to inefficiencies. Solution: dynamic prediction using LSTM improved dispatch timing and reduced wait times.
Reference	ojs.acad-pub.com/index.php/ICSE/article/download/1871/1104/



(a) Prediction results of CNN **(b)** Prediction results of LSTM

Fig 3.3 Prediction results

IV. PROBLEM STATEMENT

As smart elevators become increasingly integrated into modern infrastructure, their reliance on IoT connectivity introduces significant cybersecurity challenges. These systems, while efficient and user-friendly, are vulnerable to a range of threats including unauthorized access, data tampering, and service disruptions. Traditional security mechanisms, which often rely on static rules or signature-based detection, are inadequate for identifying novel or evolving threats in real time. This gap in security poses a risk not only to the functionality of the elevator systems but also to the safety and privacy of users. Therefore, there is a pressing need for intelligent, adaptive security solutions capable of detecting anomalies and potential cyber intrusions in smart elevator environments.

V. PROJECT BACKGROUND

The rapid advancement of Internet of Things (IoT) technologies has led to the transformation of traditional infrastructure into intelligent, interconnected systems. Among these, smart elevators have emerged as a key innovation in modern buildings, offering enhanced functionality such as automated floor access, predictive maintenance, and real-time monitoring. These systems rely on a network of embedded sensors, actuators, and communication modules that continuously collect and transmit operational data to centralized platforms, often via cloud services.

As urban environments evolve into smart cities, the demand for intelligent vertical transportation systems has grown significantly. Smart elevators not only improve user experience and operational efficiency but also contribute to broader goals such as energy optimization, accessibility, and integrated building management. However, the increased connectivity and reliance on digital infrastructure also introduce new cybersecurity risks. These include unauthorized access, data manipulation, and system disruptions, which can compromise both safety and privacy.

Traditional security approaches, which often depend on predefined rules or static configurations, are insufficient in addressing the dynamic and complex nature of modern cyber threats. This has led to a growing interest in Artificial Intelligence (AI) and machine learning (ML) as tools for enhancing the security and resilience of IoT-enabled systems. In particular, anomaly detection models have shown promise in identifying unusual patterns in telemetry data that may indicate faults or malicious activity.

This project aims to explore the application of unsupervised machine learning techniques, specifically LSTM Autoencoders, for detecting anomalies in smart elevator systems. By analyzing simulated telemetry data, the study seeks to develop a model capable of identifying potential threats in real time, thereby contributing to the development of more secure and intelligent urban infrastructure.

VI. GOALS / OBJECTIVE

The primary objective of this research is to develop and evaluate an AI-based anomaly detection model for IoT-enabled smart elevator systems. The specific goals include:

- To investigate the cybersecurity vulnerabilities associated with smart elevator systems in IoT environments.
- To collect and simulate telemetry data that reflects real-world elevator operations and potential threat scenarios.
- To design and implement an LSTM Autoencoder model capable of detecting anomalies in time-series data without prior labeling.
- To evaluate the model's performance in terms of accuracy, precision, recall, and its ability to detect both known and unknown threats.
- To explore the practical applications of the proposed model in various smart infrastructure settings, such as smart cities, hospitals, and commercial buildings.
- To identify limitations and propose future improvements for enhancing real-time threat detection and system resilience.

VII. RESEARCH METHODOLOGY

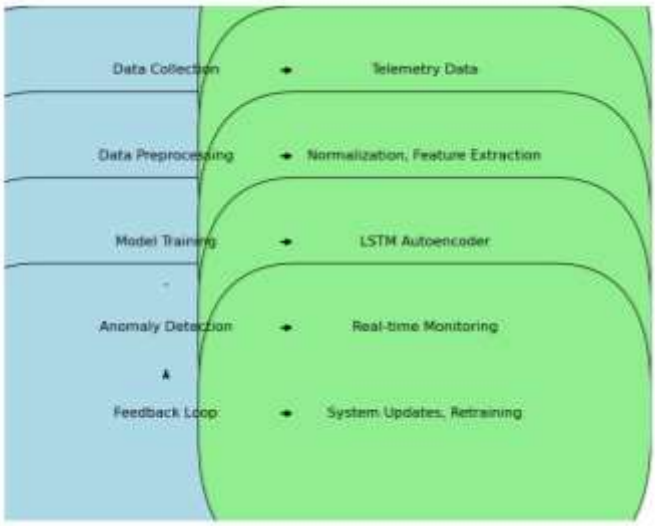


Fig 7.1 Flowchart 1

This research incorporates two powerful and complementary deep learning techniques—LSTM Autoencoder and CNN-LSTM to address the detection and classification of cybersecurity threats in IoT-based smart elevator systems. These models were selected due to their proven performance in time-series analysis and cybersecurity applications. Both models were trained and validated using telemetry data from the TON_IoT dataset, which simulates realistic IoT device behavior and cyber threats.

1. LSTM Autoencoder: Unsupervised Anomaly Detection

Objective:

- To detect unseen or abnormal behaviors in smart elevator telemetry data without 818nalyses threat categories.

How it works:

- LSTM (Long Short-Term Memory) networks are designed to remember sequences of data and detect temporal dependencies.
- An autoencoder is a neural network that attempts to compress input data into a smaller representation (encoder) and then reconstruct the original data from this representation (decoder).
- By combining both, the LSTM Autoencoder learns patterns of normal telemetry behavior—like typical elevator door states, signal fluctuations, or usage patterns.

Key Process:

- The model is trained exclusively on normal (non-malicious) data.
- During inference, it attempts to reconstruct incoming telemetry data.
- If the reconstruction error exceeds a certain threshold, the event is flagged as anomalous.
- This method is ideal for zero-day threats or unseen attack patterns, as it doesn't rely on pre-labeled threat classes.

Application in Smart Elevators:

- Detects anomalies like unexpected door behavior, abnormal power or network signal drops, or unusual usage patterns—potential indicators of cyber intrusion or hardware malfunction.

Strengths:

- Does not require labeled attack data.
- Excellent for early detection and unknown threats.
- Lightweight and efficient for IoT environments.

2. CNN-LSTM: Supervised Multi-Class Threat Classification

Objective:

- To classify telemetry events into specific types of threats (e.g., DoS, DdoS, MITM) by learning both spatial patterns and temporal sequences.

How it works:

- CNN (Convolutional Neural Networks) are effective in extracting spatial features from structured input data.
- LSTM is used to model temporal dependencies and trends over time in the telemetry.
- The CNN-LSTM hybrid model first uses convolutional layers to learn spatial patterns (like frequency or value variations across sensors), then passes this encoded information through LSTM layers to understand the time-dependent context.

Key Process:

- Data is labeled with known attack types using the TON_IoT dataset.
- CNN extracts complex features from each telemetry snapshot.
- LSTM analyses sequences of these features to identify consistent patterns over time.
- The final dense layer classifies input into one of 8 attack categories (e.g., DoS, DdoS, Backdoor, MITM, etc.).

Application in Smart Elevators:

- Helps facility IT teams identify exactly what type of attack is happening in real-time.
- Supports proactive mitigation strategies based on specific threat intelligence.

Strengths:

- High classification accuracy (~99%).
- Capable of differentiating between multiple types of attacks.
- Useful for creating detailed alerts and automated responses.

Procedure:

- Dataset Collection:
  - Dataset Used: TON_IoT Telemetry Dataset (Garage Door Device)
  - Relevance: Garage door data mimics elevator door behavior (open/close patterns), useful for anomaly detection. The dataset includes sensor readings like door state, signal strength, and timestamps, which were adapted to fit a smart elevator use case.

- Data Preprocessing:
  - Removed non-numeric and redundant columns: date, time, sphone_signal, type.
  - Encoded door_state: Converted ‘open’ to 1 and ‘closed’ to 0 using safe mapping.
  - Handled missing values: Dropped rows with null values.
  - Normalized data: Scaled using MinMaxScaler to bring all values between 0 and 1.
  - Created time-based sequences: 10-time step windows were created for feeding into the LSTM model to preserve temporal context.

- Model Development and Training:
  - Model Type: LSTM Autoencoder & CNN LSTM
  - Framework: TensorFlow/Keras

- Architecture:
- Input Layer → LSTM → Repeat Vector → LSTM → TimeDistributed Dense
- Trained on normal data only to capture baseline behavior.

- Evaluation Metrics:
  - Precision: Accuracy of threat predictions
  - Recall: Percentage of actual threats correctly detected
  - F1-score: Harmonic mean of precision and recall
  - Confusion Matrix: Comparison of true vs. predicted labels

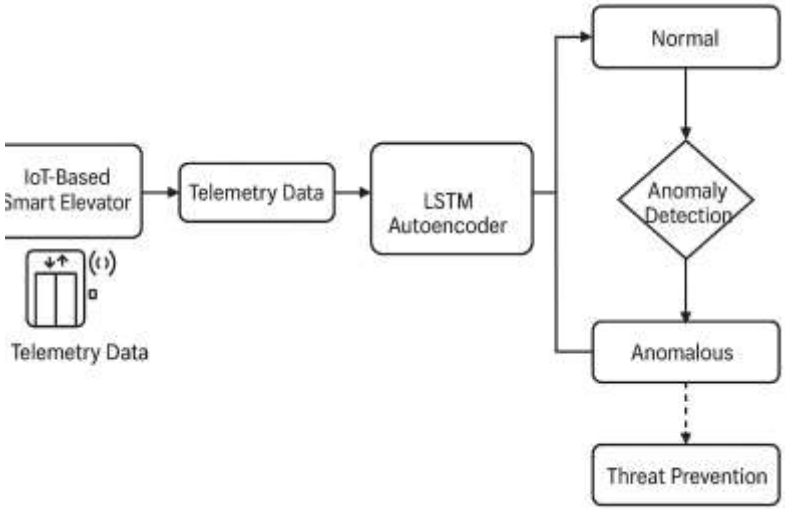


Fig 7.2 Flowchart 2

- Make sure you have:
  - Python installed.
  - Jupyter Notebook or any IDE (like VS Code, PyCharm).
  - Refer The TON_IoT Datasets | UNSW Research for dataset.

- Model 1: LSTM Autoencoder

LSTM Autoencoder Model for Anomaly Detection in IoT-based Smart Elevator

This notebook uses a LSTM Autoencoder hybrid model to detect threats using the TON_IoT Telemetry dataset.

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.preprocessing import MinMaxScaler, LabelEncoder
from sklearn.model_selection import train_test_split
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense, Dropout
from sklearn.metrics import classification_report, confusion_matrix, ConfusionMatrixDisplay
```

In [2]:
```
# Load your dataset
df = pd.read_csv("Train_Test_IoT_Garage_Door.csv")
df.head()
df.dropna(inplace=True)

# Encode object columns
for col in df.select_dtypes(include='object').columns:
    df[col] = LabelEncoder().fit_transform(df[col])

# Separate features and label
X = df.drop(columns=['type'])
y = df['type']

# Normalize features
scaler = MinMaxScaler()
X_scaled = scaler.fit_transform(X)

# Encode class labels
le = LabelEncoder()
y_encoded = le.fit_transform(y)
y_categorical = to_categorical(y_encoded)
```

In [3]:
```
timesteps = 10  # try different values as needed
X_seq, y_seq = [], []

for i in range(len(X_scaled) - timesteps):
    X_seq.append(X_scaled[i:i+timesteps])
    y_seq.append(y_categorical[i+timesteps])

X_seq = np.array(X_seq)
y_seq = np.array(y_seq)

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X_seq, y_seq, test_size=0.3, random_state=42)
```

In [4]:
```
model = Sequential()
model.add(LSTM(64, input_shape=(X_train.shape[1], X_train.shape[2]), return_sequences=False))
model.add(Dropout(0.3))
model.add(Dense(64, activation='relu'))
model.add(Dense(y_train.shape[1], activation='softmax'))  # Multi-class output

model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
model.summary()

# Train the model
model.fit(X_train, y_train, epochs=10, batch_size=64, validation_data=(X_test, y_test))
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm (LSTM)                 (None, 64)                17920

 dropout (Dropout)           (None, 64)                0

 dense (Dense)               (None, 64)                4160

 dense_1 (Dense)             (None, 8)                 520

=================================================================
Total params: 22,600
Trainable params: 22,600
Non-trainable params: 0
_________________________________________________________________
Epoch 1/10
433/433 [==============================] - 8s 9ms/step - loss: 0.3015 - accuracy: 0.9135 - val_loss: 0.0383 - val_accuracy: 0.9
943
Epoch 2/10
433/433 [==============================] - 3s 7ms/step - loss: 0.0597 - accuracy: 0.9856 - val_loss: 0.0300 - val_accuracy: 0.9
957
Epoch 3/10
433/433 [==============================] - 3s 7ms/step - loss: 0.0553 - accuracy: 0.9875 - val_loss: 0.0353 - val_accuracy: 0.9
915
Epoch 4/10
433/433 [==============================] - 3s 7ms/step - loss: 0.0470 - accuracy: 0.9888 - val_loss: 0.0255 - val_accuracy: 0.9
953
Epoch 5/10
433/433 [==============================] - 3s 7ms/step - loss: 0.0419 - accuracy: 0.9905 - val_loss: 0.0361 - val_accuracy: 0.9
894
Epoch 6/10
433/433 [==============================] - 3s 7ms/step - loss: 0.0396 - accuracy: 0.9905 - val_loss: 0.0284 - val_accuracy: 0.9
916
Epoch 7/10
433/433 [==============================] - 3s 7ms/step - loss: 0.0370 - accuracy: 0.9917 - val_loss: 0.0315 - val_accuracy: 0.9
909
Epoch 8/10
433/433 [==============================] - 3s 7ms/step - loss: 0.0395 - accuracy: 0.9903 - val_loss: 0.0262 - val_accuracy: 0.9
906
Epoch 9/10
433/433 [==============================] - 3s 7ms/step - loss: 0.0374 - accuracy: 0.9905 - val_loss: 0.0373 - val_accuracy: 0.9
914
Epoch 10/10
433/433 [==============================] - 3s 7ms/step - loss: 0.0348 - accuracy: 0.9910 - val_loss: 0.0240 - val_accuracy: 0.9
933
```

Out[4]: <keras.callbacks.History at 0x2a7704f3d30>

In [6]:
```
y_pred_probs = model.predict(X_test)
y_pred = y_pred_probs.argmax(axis=1)
y_true = y_test.argmax(axis=1)

# Fix: Convert class labels to strings
print(classification_report(y_true, y_pred, target_names=[str(label) for label in le.classes_]))

# Confusion Matrix
cm = confusion_matrix(y_true, y_pred)
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=[str(label) for label in le.classes_])
disp.plot(cmap='Blues', xticks_rotation=45)
plt.title("Confusion Matrix - LSTM Multi-Class")
plt.show()
```

```
372/372 [==============================] - 1s 3ms/step
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      1525
           1       1.00      0.97      0.98      1523
           2       0.97      1.00      0.98      1498
           3       1.00      1.00      1.00      4464
           4       1.00      1.00      1.00      1472
           5       0.99      0.99      0.99       897
           6       1.00      0.99      0.99       161
           7       0.97      0.99      0.98       334

    accuracy                           0.99     11874
   macro avg       0.99      0.99      0.99     11874
weighted avg       0.99      0.99      0.99     11874
```

- Model 2: CNN-LSTM

CNN-LSTM Model for Anomaly Detection in IoT-based Smart Elevator

This notebook uses a CNN-LSTM hybrid model to detect threats using the TON_IoT Telemetry dataset.

In [1]:
```
# Import Libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.preprocessing import MinMaxScaler, LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv1D, MaxPooling1D, LSTM, Dense, Flatten, Dropout
from tensorflow.keras.utils import to_categorical
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
from sklearn.preprocessing import LabelEncoder
```

In [2]:
```
# Load your telemetry dataset (modify path as needed)
df = pd.read_csv('Train_Test_IoT_Garage_Door.csv')
df.head()
df = df.dropna()

# Encode categorical features
for col in df.select_dtypes(include='object').columns:
    df[col] = LabelEncoder().fit_transform(df[col])

# Split features and labels
X = df.drop(columns=['type'])
y = df['type']

# Scale features
scaler = MinMaxScaler()
X_scaled = scaler.fit_transform(X)

# One-hot encode labels
y_encoded = to_categorical(LabelEncoder().fit_transform(y))

# Reshape for CNN-LSTM input: (samples, time steps, features)
timesteps = 10
n_features = X_scaled.shape[1]
X_cnn_lstm = []
y_cnn_lstm = []
for i in range(len(X_scaled) - timesteps):
    X_cnn_lstm.append(X_scaled[i:i+timesteps])
    y_cnn_lstm.append(y_encoded[i+timesteps])

X_cnn_lstm = np.array(X_cnn_lstm)
y_cnn_lstm = np.array(y_cnn_lstm)

X_train, X_test, y_train, y_test = train_test_split(X_cnn_lstm, y_cnn_lstm, test_size=0.3, random_state=42)
```

In [3]:
```
# Build CNN-LSTM model
model = Sequential()
model.add(Conv1D(filters=64, kernel_size=3, activation='relu', input_shape=(timesteps, n_features)))
model.add(MaxPooling1D(pool_size=2))
model.add(LSTM(50))
model.add(Dense(50, activation='relu'))
model.add(Dropout(0.3))
model.add(Dense(y_train.shape[1], activation='softmax'))

model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv1d (Conv1D)             (None, 8, 64)             1024

 max_pooling1d (MaxPooling1D  (None, 4, 64)            0
 )

 lstm (LSTM)                 (None, 50)                23000

 dense (Dense)               (None, 50)                2550

 dropout (Dropout)           (None, 50)                0

 dense_1 (Dense)             (None, 8)                 408

=================================================================
Total params: 26,982
Trainable params: 26,982
Non-trainable params: 0
_________________________________________________________________
```

In [4]:
```
# Train the model
history = model.fit(X_train, y_train, epochs=10, batch_size=64, validation_data=(X_test, y_test))
```

```
Epoch 1/10
433/433 [==============================] - 8s 8ms/step - loss: 0.3358 - accuracy: 0.8989 - val_loss: 0.1188 - val_accuracy: 0.9
639
Epoch 2/10
433/433 [==============================] - 3s 7ms/step - loss: 0.0635 - accuracy: 0.9865 - val_loss: 0.0543 - val_accuracy: 0.9
807
Epoch 3/10
433/433 [==============================] - 3s 7ms/step - loss: 0.0540 - accuracy: 0.9882 - val_loss: 0.0237 - val_accuracy: 0.9
956
Epoch 4/10
433/433 [==============================] - 3s 7ms/step - loss: 0.0440 - accuracy: 0.9906 - val_loss: 0.0172 - val_accuracy: 0.9
967
Epoch 5/10
433/433 [==============================] - 4s 10ms/step - loss: 0.0469 - accuracy: 0.9887 - val_loss: 0.0236 - val_accuracy: 0.
9912
Epoch 6/10
433/433 [==============================] - 3s 7ms/step - loss: 0.0354 - accuracy: 0.9918 - val_loss: 0.0145 - val_accuracy: 0.9
969
Epoch 7/10
433/433 [==============================] - 3s 7ms/step - loss: 0.0359 - accuracy: 0.9911 - val_loss: 0.0387 - val_accuracy: 0.9
857
Epoch 8/10
433/433 [==============================] - 3s 7ms/step - loss: 0.0319 - accuracy: 0.9923 - val_loss: 0.0294 - val_accuracy: 0.9
896
Epoch 9/10
433/433 [==============================] - 4s 10ms/step - loss: 0.0285 - accuracy: 0.9930 - val_loss: 0.0049 - val_accuracy: 0.
9742
Epoch 10/10
433/433 [==============================] - 3s 8ms/step - loss: 0.0311 - accuracy: 0.9918 - val_loss: 0.0166 - val_accuracy: 0.9
941
```

In [5]:
```
# Evaluate the model
y_pred_probs = model.predict(X_test)
y_pred = np.argmax(y_pred_probs, axis=1)
y_true = np.argmax(y_test, axis=1)

print(classification_report(y_true, y_pred))
```

```
372/372 [==============================] - 2s 3ms/step
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      1525
           1       0.96      1.00      0.98      1523
           2       1.00      0.96      0.98      1498
           3       1.00      1.00      1.00      4464
           4       1.00      1.00      1.00      1472
           5       1.00      1.00      1.00       897
           6       1.00      0.99      0.99       161
           7       0.99      0.99      0.99       334

    accuracy                           0.99     11874
   macro avg       0.99      0.99      0.99     11874
weighted avg       0.99      0.99      0.99     11874
```

```
In [6]: # Detect class names
le = LabelEncoder()
le.fit(df['type'])  # Original 'type' column before encoding
class_names = le.classes_

# Generate and display confusion matrix
cm = confusion_matrix(y_true, y_pred)
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=class_names)
disp.plot(cmap=plt.cm.Blues, xticks_rotation=45)
plt.title("Confusion Matrix - CNN-LSTM")
plt.show()
```

VIII. RESULTS AND DISCUSSION

8.1 Results of Model 1: LSTM Autoencoder

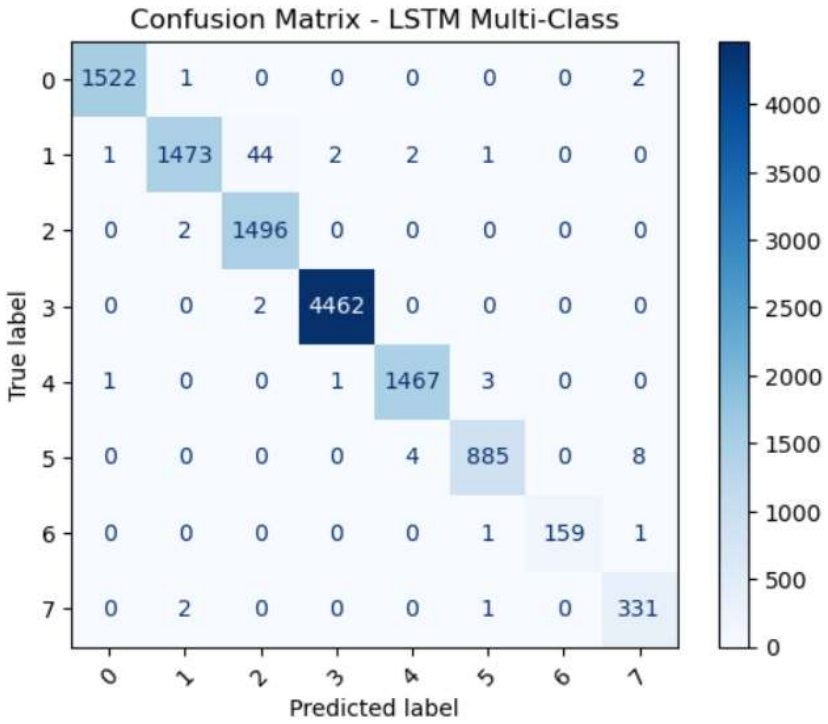


Fig 8.1 Confusion matrix

Table 8.1: Classification Report – LSTM Autoencoder

Class	Precision	Recall	F1-score	Summary
0 - Backdoor	**1.00**	**1.00**	**1.00**	**Perfect performance**
1 - DDoS	**1.00**	**0.97**	**0.98**	**Slight drop in recall — some DDoS were missed**
2 - DoS	**0.97**	**1.00**	**0.98**	**Slight drop in precision — a few misclassifications**
3 - Injection	**1.00**	**1.00**	**1.00**	**Excellent**

4 - MITM	**1.00**	**1.00**	**1.00**	**Excellent**
5 - Password	**0.99**	**0.99**	**0.99**	**Excellent**
6 - Ransomware	**1.00**	**0.99**	**0.99**	**Slight miss in recall**
7 - XSS	**0.97**	**0.99**	**0.98**	**Very good**

8.2 Results of Model 2: CNN LSTM

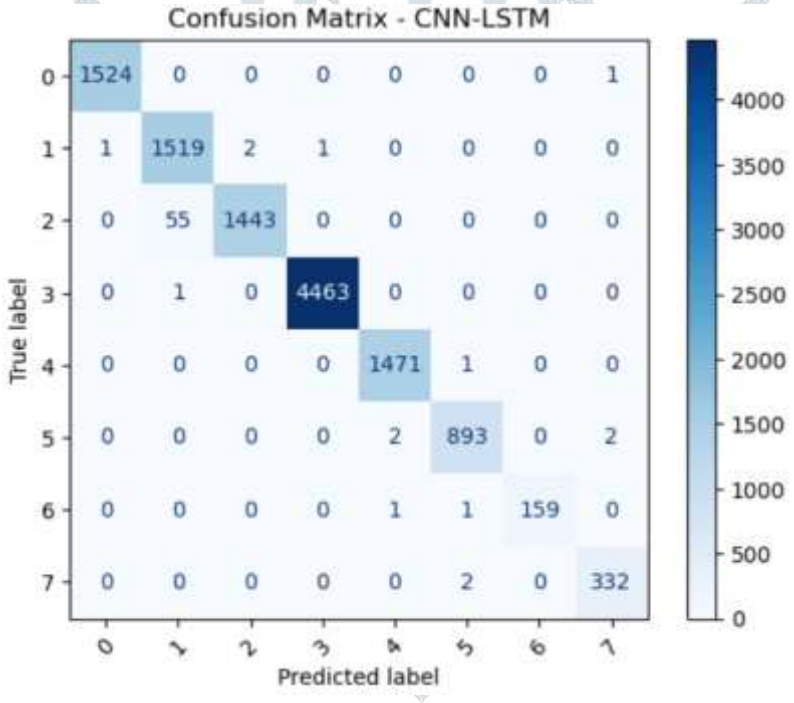


Fig 8.2 Confusion matrix

Table 8.2: Classification Report – CNN LSTM

Class	Precision	Recall	F1-Score	Summary
0 - Backdoor	**1.00**	**1.00**	**1.00**	**Perfect detection**
1 - DDoS	**0.96**	**1.00**	**0.98**	**Very high, slight miss in precision**
2 - DoS	**1.00**	**0.96**	**0.98**	**A few DoS attacks misclassified**
3 - Injection	**1.00**	**1.00**	**1.00**	**Perfect**

4 - MITM	**1.00**	**1.00**	**1.00**	**Perfect**
5 - Password	**1.00**	**1.00**	**1.00**	**Perfect**
6 - Ransomware	**1.00**	**0.99**	**0.99**	**1 sample missed**
7 - XSS	**0.99**	**0.99**	**0.99**	**Slight drop in precision and recall**

IX. FINDINGS

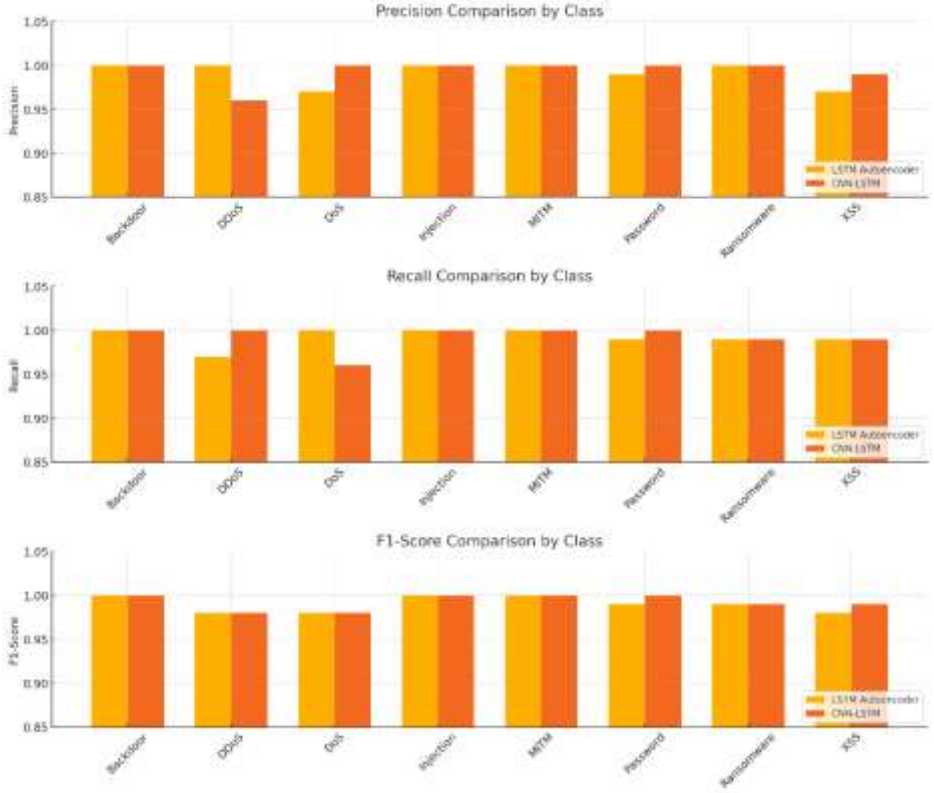


Fig 9.1 Graph Metrics Vs Class

- Deep Learning Enhances Threat Detection
  - Traditional rule-based systems fail to detect unknown or evolving cyber threats.
  - AI models like LSTM Autoencoder and CNN-LSTM provide dynamic learning capabilities that improve detection of both known and novel threats.
- LSTM Autoencoder is Effective for Anomaly Detection
  - The LSTM Autoencoder model achieved ~99% accuracy in detecting deviations from normal behavior using reconstruction error.
  - Ideal for identifying unusual patterns such as abnormal elevator door activity or sudden signal drops.
- CNN-LSTM Enables Multi-Class Classification
  - The CNN-LSTM model accurately classified 8 distinct types of cyber threats (e.g., DoS, DDoS, Backdoor, MITM).

- Delivered high performance with macro average F1-score of 0.99, making it suitable for real-time classification of diverse threats.

- TON_IoT Dataset is Practical for Real-World Simulation
  - The use of the TON_IoT Telemetry dataset provided a realistic simulation of IoT sensor data from smart systems, ensuring the models were trained and validated with authentic scenarios.
- Combination of AI Models Provides Full-Spectrum Protection
  - LSTM Autoencoder is best for real-time anomaly detection.
  - CNN-LSTM is best for classifying and understanding the nature of the threats.
  - Together, they form a hybrid security framework that is both proactive and responsive.
- Cloud Integration is Key for Scalable Deployment
  - Smart elevator systems can leverage cloud-based AI models, ensuring scalability and real-time processing without needing heavy edge devices like Raspberry Pi or Arduino.
- Schematic Architecture Enables Practical Deployment
  - A clearly defined architecture shows how sensors, microcontrollers, cloud services, and AI modules interact.
  - Helps in visualizing where AI models are applied in the data pipeline.

X. CONCLUSION

This research explored the integration of Artificial Intelligence (AI) techniques—specifically LSTM Autoencoders and CNN-LSTM networks—for detecting and classifying cybersecurity threats in IoT-based smart elevator systems. The study utilized the TON_IoT Telemetry dataset to train and evaluate both models, providing practical insights into the models' effectiveness.

The LSTM Autoencoder proved to be a powerful tool for unsupervised anomaly detection, capable of learning the normal operational patterns of elevator telemetry data and flagging deviations that may indicate potential threats. On the other hand, the CNN-LSTM hybrid model offered high accuracy in supervised multi-class classification, efficiently identifying and categorizing multiple types of cyber threats such as DoS, DDoS, Backdoor, and MITM attacks.

With both models achieving near-perfect accuracy, this dual-model approach demonstrates a highly robust, intelligent, and scalable framework for enhancing the security of smart elevators. This system addresses a critical need in the IoT domain, where conventional rule-based systems fall short in detecting dynamic or previously unknown threats.

XI. FUTURE SCOPE

While the current study has achieved strong results, there are several promising directions for future research and real-world implementation:

1. Edge Deployment and Real-Time Processing

- Future work can focus on deploying trained models onto edge devices (e.g., NVIDIA Jetson, Coral TPU) to enable real-time processing within the elevator system, reducing reliance on cloud latency.

2. Integration with Elevator Control Systems

- A seamless integration between the AI model and actual elevator control software can allow the system to take automated protective actions such as isolating network traffic, locking doors, or sending alerts.

3. Adaptive Learning and Model Updates

- Introduce continuous learning pipelines to ensure the models evolve with new threats, reducing the risk of model drift over time.

4. User Behavior Analytics

- Incorporate user access patterns and biometric data to enhance behavioral anomaly detection, such as unauthorized access or prolonged dwell time at specific floors.

5. Federated Learning for Data Privacy

- Employ federated learning to train AI models across multiple buildings or elevators without transferring sensitive raw data to a central server.

6. Comprehensive Threat Intelligence Dashboard

- Develop a visual dashboard for real-time monitoring, threat classification, anomaly scores, and system health, which can be useful for facility managers or IT teams.

REFERENCES

[1] A. Halbouni, T. S. Gunawan, M. H. Habaebi, M. Halbouni, M. Kartiwi, and R. Ahmad, "CNN-LSTM: Hybrid Deep Neural Network for Network Intrusion Detection System," *IEEE Access*, pp. 1–1, 2022, doi: https://doi.org/10.1109/access.2022.3206425.

[2] T. Markovic, Alireza Dehlaghi-Ghadim, M. Leon, A. Balador, and Sasikumar Punnekkat, "Time-series Anomaly Detection and Classification with Long Short-Term Memory Network on Industrial Manufacturing Systems," *Annals of Computer Science and Information Systems*, Sep. 2023, doi: https://doi.org/10.15439/2023f5263.

[3] F. Khanmohammadi and R. Azmi, "Time-Series Anomaly Detection in Automated Vehicles Using D-CNN-LSTM Autoencoder," *IEEE Transactions on Intelligent Transportation Systems*, vol. 25, no. 8, pp. 9296–9307, Aug. 2024, doi: https://doi.org/10.1109/tits.2024.3380263.

[4] Y. Fu, Y. Du, Z. Cao, Q. Li, and W. Xiang, "A Deep Learning Model for Network Intrusion Detection with Imbalanced Data," *Electronics*, vol. 11, no. 6, p. 898, Mar. 2022, doi: https://doi.org/10.3390/electronics11060898.

[5] S. Maleki, S. Maleki, and N. R. Jennings, "Unsupervised anomaly detection with LSTM autoencoders using statistical data-filtering," *Applied Soft Computing*, vol. 108, p. 107443, Sep. 2021, doi: https://doi.org/10.1016/j.asoc.2021.107443.

[6] M. Elsayed, Nhien-An Le-Khac, S. Dev, and Anca Delia Jurcut, "Network Anomaly Detection Using LSTM Based Autoencoder," *Proceedings of the 16th ACM Symposium on QoS and Security for Wireless and Mobile Networks*, Nov. 2020, doi: https://doi.org/10.1145/3416013.3426457.

[7] Hajar Homayouni, S. Ghosh, I. Ray, Shlok Gondalia, J. Duggan, and M. G. Kahn, "An Autocorrelation-based LSTM-Autoencoder for Anomaly Detection on Time-Series Data," Dec. 2020, doi: https://doi.org/10.1109/bigdata50022.2020.9378192.

[8] A. Sagheer and M. Kotb, "Unsupervised Pre-training of a Deep LSTM-based Stacked Autoencoder for Multivariate Time Series Forecasting Problems," *Scientific Reports*, vol. 9, no. 1, Dec. 2019, doi: https://doi.org/10.1038/s41598-019-55320-6.

[9] G. Hangli, T. Hamada, T. Sumitomo, and N. Koshizuka, "Intellevator: an Intelligent Elevator System Proactive in Traffic Control for Time-Efficiency Improvement," *IEEE Access*, vol. 8, pp. 35535–35545, 2020, doi: https://doi.org/10.1109/ACCESS.2020.2975020.

[10]B. W. Masduki, K. Ramli, F. A. Saputra, and D. Sugiarto, "Study on implementation of machine learning methods combination for improving attacks detection accuracy on Intrusion Detection System (IDS)," *IEEE Xplore*, Aug. 01, 2015. https://ieeexplore.ieee.org/document/7374895 (accessed Oct. 19, 2020).

[11] L. Huang, Y. Chen, and K. Zhao, "Intelligent elevator system based on the Internet of Things: improving elevator safety and management efficiency," *International Conference on Smart Transportation and City Engineering (STCE 2022)*, pp. 236–236, Apr. 2025, doi: https://doi.org/10.1117/12.3062341.

[12] N. Rane, S. Choudhary, and J. Rane, "Artificial Intelligence (AI) and Internet of Things (IoT) - based sensors for monitoring and controlling in architecture, engineering, and construction: applications, challenges, and opportunities," *Social Science Research Network*, Jan. 2023, doi: https://doi.org/10.2139/ssrn.4642197.

[13] A. Alrawais, A. Alhothaily, C. Hu, and X. Cheng, "Fog Computing for the Internet of Things: Security and Privacy Issues," *IEEE Internet Computing*, vol. 21, no. 2, pp. 34–42, Mar. 2017, doi: https://doi.org/10.1109/mic.2017.37.

[14] I. Alrashdi, A. Alqazzaz, E. Aloufi, R. Alharthi, M. Zohdy, and H. Ming, "AD-IoT: Anomaly Detection of IoT Cyberattacks in Smart City Using Machine Learning," *IEEE Xplore*, Jan. 01, 2019. https://ieeexplore.ieee.org/document/8666450 (accessed Dec. 08, 2021).

[15] M. Ali, Aamir Raza, Malik Arslan Akram, Haroon Arif, and Aamir Ali, "Enhancing IOT Security: A review of Machine Learning-Driven Approaches to Cyber Threat Detection," *Journal of Informatics and Interactive Technology*, vol. 2, no. 1, pp. 316–324, Apr. 2025, doi: https://doi.org/10.63547/jiite.v2i1.64.

[16] M. Bhavsar, K. Roy, J. Kelly, and Odeyomi Olusola, "Anomaly-based intrusion detection system for IoT application," *Discover Internet of things*, vol. 3, no. 1, May 2023, doi: https://doi.org/10.1007/s43926-023-00034-5.

[17] A. Khacha, Rafika Saadouni, Y. Harbi, and Zibouda Aliouat, "Hybrid Deep Learning-based Intrusion Detection System for Industrial Internet of Things," Nov. 2022, doi: https://doi.org/10.1109/isia55826.2022.9993487.

[18] Noor Wali Khan *et al.*, "A hybrid deep learning-based intrusion detection system for IoT networks," *Mathematical Biosciences and Engineering*, vol. 20, no. 8, pp. 13491–13520, Jan. 2023, doi: https://doi.org/10.3934/mbe.2023602.

[19] M. Abdullahi *et al.*, "Detecting Cybersecurity Attacks in Internet of Things Using Artificial Intelligence Methods: A Systematic Literature Review," *Electronics*, vol. 11, no. 2, p. 198, Jan. 2022, doi: https://doi.org/10.3390/electronics11020198.

[20] Kanchan Naithani, "AI-based Intrusion Detection System for Internet of Things (IoT) Networks," *Türk bilgisayar ve matematik eğitimi dergisi*, vol. 10, no. 2, pp. 1095–1100, Sep. 2019, doi: https://doi.org/10.17762/turcomat.v10i2.13631.

[21] "The TON_IoT Datasets | UNSW Research," *research.unsw.edu.au*. https://research.unsw.edu.au/projects/toniot-datasets

[22] N. Moustafa, Marwa Keshky, Essam Debiez, and H. Janicke, "Federated TON_IoT Windows Datasets for Evaluating AI-Based Security Applications," Dec. 2020, doi: https://doi.org/10.1109/trustcom50675.2020.00114.

[23] N. Moustafa, "A New Distributed Architecture for Evaluating AI-based Security Systems at the Edge: Network TON_IoT DatasetsNourMoustafaa1∗TON_IoT datasets: A distributed architecture for evaluating Artificial Intelligence-based security systems in IoT networks," *Sustainable Cities and Society*, vol. 72, p. 102994, May 2021, doi: https://doi.org/10.1016/j.scs.2021.102994.

[24] T. M. Booij, I. Chiscop, E. Meeuwissen, N. Moustafa, and F. T. H. den Hartog, "ToN_IoT: The Role of Heterogeneity and the Need for Standardization of Features and Attack Types in IoT Network Intrusion Data Sets," *IEEE Internet of Things Journal*, vol. 9, no. 1, pp. 485–496, Jan. 2022, doi: https://doi.org/10.1109/jiot.2021.3085194.

[25] A. Alsaedi, N. Moustafa, Z. Tari, A. Mahmood, and A. Anwar, "TON_IoT Telemetry Dataset: A New Generation Dataset of

IoT and IIoT for Data-driven Intrusion Detection Systems," *IEEE Access*, vol. 8, pp. 1–1, 2020, doi: https://doi.org/10.1109/access.2020.3022862.

[26] N. Moustafa, M. Ahmed, and Sherif Sayed Ahmed, "Data Analytics-Enabled Intrusion Detection: Evaluations of ToN_IoT Linux Datasets," Dec. 2020, doi: https://doi.org/10.1109/trustcom50675.2020.00100.

[27] N. Moustafa, "New Generations of Internet of Things Datasets for Cybersecurity Applications based Machine Learning: TON_IoT Datasets," Sep. 2019, doi: https://doi.org/10.26190/5d7ac9bfe8487.

[28] N. Moustafa, "A Systemic IoT-Fog-Cloud Architecture for Big-Data Analytics and Cyber Security Systems: A Review of Fog Computing," *arXiv (Cornell University)*, May 2019.

[29] J. Ashraf *et al.*, "IoTBoT-IDS: A novel statistical learning-enabled botnet detection framework for protecting networks of smart cities," *Sustainable Cities and Society*, vol. 72, p. 103041, Sep. 2021, doi: https://doi.org/10.1016/j.scs.2021.103041.

[30] P. Malhotra, A. Ramakrishnan, G. Anand, L. Vig, P. Agarwal, and G. R. Shroff, "LSTM-based Encoder-Decoder for Multi-sensor Anomaly Detection," Jul. 2016, doi: https://doi.org/10.48550/arxiv.1607.00148.

[31] C. Yin, Y. Zhu, J. Fei, and X. He, "A Deep Learning Approach for Intrusion Detection Using Recurrent Neural Networks," *IEEE Access*, vol. 5, pp. 21954–21961, 2017, doi: https://doi.org/10.1109/access.2017.2762418.

[32] N. Shone, T. N. Ngoc, V. D. Phai, and Q. Shi, "A Deep Learning Approach to Network Intrusion Detection," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 2, no. 1, pp. 41–50, Feb. 2018, doi: https://doi.org/10.1109/tetci.2017.2772792.

[33] A. Javaid, Q. Niyaz, W. Sun, and M. Alam, "A Deep Learning Approach for Network Intrusion Detection System," *Proceedings of the 9th EAI International Conference on Bio-inspired Information and Communications Technologies (formerly BIONETICS)*, 2016, doi: https://doi.org/10.4108/eai.3-12-2015.2262516.

[34] Z. Ahmad, A. Shahid Khan, C. Wai Shiang, J. Abdullah, and F. Ahmad, "Network intrusion detection system: A systematic study of machine learning and deep learning approaches," *Transactions on Emerging Telecommunications Technologies*, vol. 32, no. 1, pp. 1–29, Oct. 2020, doi: https://doi.org/10.1002/ett.4150.

[35] B. Roy and H. Cheung, "A Deep Learning Approach for Intrusion Detection in Internet of Things using Bi-Directional Long Short-Term Memory Recurrent Neural Network," *IEEE Xplore*, Nov. 01, 2018. https://ieeexplore.ieee.org/document/8615294/

[36] S. S. Roy, A. Mallik, R. Gulati, M. S. Obaidat, and P. V. Krishna, "A Deep Learning Based Artificial Neural Network Approach for Intrusion Detection," *Communications in Computer and Information Science*, pp. 44–53, 2017, doi: https://doi.org/10.1007/978-981-10-4642-1_5.

[37] R. Abdulhammed, M. Faezipour, A. Abuzneid, and A. AbuMallouh, "Deep and Machine Learning Approaches for Anomaly-Based Intrusion Detection of Imbalanced Network Traffic," *IEEE Sensors Letters*, vol. 3, no. 1, pp. 1–4, Jan. 2019, doi: https://doi.org/10.1109/lsens.2018.2879990.

[38] C. Zhang, F. Ruan, L. Yin, X. Chen, L. Zhai, and F. Liu, "A Deep Learning Approach for Network Intrusion Detection Based on NSL-KDD Dataset," *2019 IEEE 13th International Conference on Anti-counterfeiting, Security, and Identification (ASID)*, Oct. 2019, doi: https://doi.org/10.1109/icasid.2019.8925239.

[39] S. S. Bamber, A. V. R. Katkuri, S. Sharma, and M. Angurala, "A hybrid CNN-LSTM approach for intelligent cyber intrusion detection system," *Computers & Security*, vol. 148, p. 104146, Jan. 2025, doi: https://doi.org/10.1016/j.cose.2024.104146.

[40] A.-A. hadi, A. Saeed, and Ahmed Mohsin Mahdi, "IoT Cybersecurity Threats and Detection Mechanisms: A Review," *Wasit Journal of Pure sciences*, vol. 2, no. 2, pp. 231–250, Jun. 2023, doi: https://doi.org/10.31185/wjps.136.

[41] K. S. Dr.R.Venkatesh Dr. Uma Maheswari N, "Network Anomaly Detection for NSL-KDD Dataset Using Deep Learning," *INFORMATION TECHNOLOGY IN INDUSTRY*, vol. 9, no. 2, pp. 821–827, Mar. 2021, doi: https://doi.org/10.17762/itii.v9i2.419.

[42] M. Shi and Y. Choi, "Comparison of the elevator traffic flow prediction between the neural networks of CNN and LSTM," *Intelligent Control and System Engineering*, vol. 2024, no. 1, p. 1871, doi: https://doi.org/10.59400/icse1871.