International Journal of Applied Mathematics
Volume 38 No. 7s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

DESIGNING ATTACK-RESISTANT AND INTELLIGENT TRUST MANAGEMENT USING DEEP NEURAL NETWORKS

Jayashree Chandrakant Pasalkar[1*] , Dr. Dattatraya Shankar Bormane[2]

[1]Research Scholar, AISSMS's College of Engineering

Savitribai Phule Pune University, Pune, India

jayashree.pasalkar@aissmsioit.org

[2]Principle, AISSMS's College of Engineering

Savitribai Phule Pune University, Pune, India.

bdattatraya@gmail.com

Abstract:

The rapid expansion of IoT and cyber-physical systems has exposed critical weaknesses in trust management, making networks vulnerable to sophisticated adversarial threats. Conventional trust models often lack the adaptability and robustness required in dynamic environments. To address this, we propose an intelligent trust management framework powered by deep neural networks that not only detects attacks with high accuracy but also demonstrates resilience against adversarial manipulation. The approach integrates advanced pre-processing, feature selection, and balancing techniques, followed by classification using LSTM, GRU, and a hybrid LSTM–GRU architecture, where the hybrid consistently surpassed standalone models in both binary and multiclass tasks. For the CAN dataset, convolution-based models, including a lightweight VGG16, provided efficient and reliable detection of complex attack patterns. Beyond detection, the system was rigorously tested under adversarial conditions such as evasion, poisoning, and inference attacks, and showed notable resistance through strategies like adversarial training. Overall, this dual capability of accurate attack detection and strong attack resistance highlights the scalability, reliability, and security of the proposed system, making it well-suited for modern IoT trust management.

Keywords: Trust Management, Deep Neural Networks (DNNs), IoT Security, Attack Detection, Adversarial Resistance, Hybrid LSTM–GRU Models

A. Introduction

In today's digital world, where everything is linked to everything else, trust management has become essential for safe communication and decision-making in cyber-physical and Internet of Things (IoT) systems. As the number of IoT-enabled apps in healthcare, transportation, finance, and smart cities grows at an exponential rate, sophisticated attacks on the integrity and reliability of data exchanges become more common [1]. In the real world, things like large-scale ransomware campaigns against connected medical devices or attacks on self-driving car networks show how trust issues can put safety and security at risk [2]. This important problem

Received: August 02, 2025

Volume 38 No. 7s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

makes it clear how important it is to have smart, attack-proof trust management systems that can change with the times and keep things reliable in changing settings. Even though there has been a lot of study on trust management frameworks, most of the time people still use heuristic methods like reputation-based schemes, probabilistic trust scores, or fixed rules. Most of the time, these methods aren't strong enough to deal with big amounts of different types of data, interactions with a lot of dimensions, and attackers' strategies that change over time. In particular, [3], standard trust computation methods have a hard time finding adversarial behaviours like poisoning, inference, evasion, and extraction attacks, which use system flaws on purpose to make decisions less accurate. In Internet of Things (IoT) environments, this trust gap is very important because broken trust not only hurts service quality but can also put people in danger in safety-critical areas. As a result, there is a clear study gap in making models that can accurately detect attacks and be resistant to manipulation by adversaries at the same time.

Existing intruder detection systems that use machine learning can spot some oddities, but they don't always work well in changing, hostile environments. Their ability to identify things is limited by shallow architectures, feature engineering limitations, and data imbalance problems [4]. Also, their ability to withstand changes made by attackers hasn't been studied much. So, systems that use these techniques tend to be less accurate, give a lot of false positives, and not be able to change well to inputs that they haven't seen or that are meant to hurt them. These problems show that the current methods are not good enough to provide complete, safe trust management [5].

The main goal of this study is to create an intelligent trust management framework that uses deep neural networks to find malicious activity very accurately and can also withstand attacks from other people. Using Long Short-Term Memory (LSTM), Gated Recurrent Units (GRU), and hybrid LSTM–GRU architectures, the study aims to improve temporal feature learning for attack detection in the CICIoT dataset. For intrusion classification in the CAN dataset, Convolutional Neural Networks (CNN) and lightweight VGG16 are used. Additionally, adversarial resistance is tested by using the Fast Gradient Method to create evasion, inference, extraction, and poisoning attacks. This checks how strong the model is in dangerous environments.

The key contributions of this study can be summarized as follows,

- To design an intelligent and attack-resistant trust management framework that leverages deep neural networks for both accurate attack detection and strong adversarial resistance in IoT and cyber-physical systems.
- To compare the performance of advanced deep models (LSTM, GRU, and Hybrid LSTM–GRU) with traditional standalone approaches, and to analyze improvements in accuracy, F1-score, and error reduction achieved by the proposed hybrid architecture.
- To validate the generalizability of the proposed system using two heterogeneous benchmark datasets;
  - The CIC-IoT 2023 dataset for flow-based IoT intrusion detection.
  - The CAN intrusion dataset for image-based vehicular attack detection.

Received: August 02, 2025

Volume 38 No. 7s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

- To integrate lightweight architectures such as a reduced VGG16 for efficient classification on the CAN dataset, ensuring high accuracy while maintaining computational efficiency suitable for real-time IoT deployment.
- To evaluate adversarial robustness of the proposed framework by simulating evasion, inference, and poisoning attacks (e.g., FGSM), and to implement defense mechanisms such as adversarial training and preprocessing for enhanced system resilience.
- To provide a comprehensive result analysis through accuracy/loss curves, confusion matrices, and comparative evaluations of different models and feature sets (all features vs. flow features), thereby demonstrating the scalability and reliability of the proposed system.

B. Related Work

Classical reputation and rule-based trust engines have been improved with deep representation learning to model the temporal and spatial dependencies in flows and CAN frames. This has led to measurable improvements in precision and recall over shallow baselines, but it also shows that there are still gaps when there is a class imbalance or a shift in the distribution [6], [7]. Many networks use recurrent architectures, like LSTM and GRU, to read network telemetry that shows long-range dependencies [8] Hybrid stacks or ensembles often do better than single backbones by combining complementary gating dynamics and improving minority-class recall after resampling or cost-sensitive learning [9, 10]. Feature locality capture is made easier in vehicular networks by convolutional pipelines that turn packet or signal fields into image-like tensors. Lightweight VGG-style models and compact CNNs show good throughput–accuracy trade-offs that make them ideal for embedded focus and gateways [11, 12]. To deal with label lack and skew, new studies include SMOTE variants, mixup, or focal loss. They find that stability improves for rare attacks without increasing the number of false hits in harmless streams [13].

Attack-resilient trust management needs more than just discovery. It also needs clear modelling of adversarial risk. It has been shown that strong training with gradient-based example generation (FGSM/PGD) and input transformations (bit-depth reduction, randomised smoothing) can reduce the chance of escape while keeping the accuracy of clean data in traffic and CAN datasets [14]. It is possible to make conservative policy choices at the edge by using defense-aware calibration and uncertainty quantification (temperature scaling, evidential deep learning) to cut down on overly confident trust assignments in cases that are close to the line [16]. Lines that go together Using graph-structured learning to spread trust over device–service interaction graphs is helpful. GNNs encode relational signals and stop isolated spoofing with message-passing regularizers. However, robustness drops when targeted poisoning happens unless joint structure–feature defences are used [17]. More and more research is being done on privacy-preserving training using federated optimisation, secure aggregation, and client-level differential privacy to stop inference/extraction attacks on sensitive traces. However, utility drops due to tight budgets are still a worry for latency-critical deployments [18]. Post-hoc attributions (Grad-CAM, integrated gradients) can show false correlations in traffic features

Received: August 02, 2025

International Journal of Applied Mathematics
Volume 38 No. 7s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

and help with feature pruning or acquisition policies that make generalisation better when covariates change [19].

System-level studies focus on complete pathways that combine preprocessing (casting IP/time fields, normalisation, and flow aggregation) with vision-style models and temporal DNNs. They find that consistent schema handling, stratified splits, and scenario-specific validation are just as important as model choice for achieving repeatable gains on CICIoT and CAN benchmarks [20]. It has been shown over and over that hybrids (like LSTM–GRU or CNN with recurrent heads) produce the best macro-F1. This is especially true when trained with class-balanced sampling and early stopping on minority metrics. Adversarial test harnesses show that naive detectors can lose tens of percent of their accuracy when small changes are made. However, fully trained models limit this loss to tens of percent and keep trust levels that policy engines can act on. In last the contributions that focus on deployment suggest edge-ready architectures that use quantization-aware training and depthwise separable convolutions to help IoT gateways work in real time. Streaming inference with sliding windows keeps the temporal context without adding too much memory overhead [21]. Table 1 lists new approaches to managing trust that can't be attacked, highlighting datasets, results, and problems. When it comes to accuracy, hybrid deep models are the best. Adversarial defences and edge-optimized methods make them more robust and scalable.

Table 1: Summary of Related Work on Attack-Resistant and Intelligent Trust Management

Methods Used	**Key Finding of Article**	**Result (Accuracy/F1/ etc.)**	**Dataset Used**	**Limitation**	**Scope for Future Work**
LSTM-based IDS	Captured temporal attack patterns in IoT traffic	96% accuracy, 95% F1	CICIoT	Limited resilience to adversarial evasion attacks	Incorporate adversarial training techniques
GRU-based detection [22]	Efficient learning with fewer parameters	97% accuracy, 96% F1	CICIoT	Slightly weaker generalizatio n on multiclass tasks	Hybridize with LSTM for stronger sequence modeling
Hybrid LSTM–GRU	Outperformed single models in binary and multiclass tasks	98.7% accuracy, 97.9% F1	CICIoT	High computationa l overhead	Optimize hybrid model for edge deployment
CNN for CAN frames	Detected anomalies via image transformation of CAN data	93.2% accuracy	CAN dataset	Accuracy dropped under adversarial perturbations	Add adversarial defense modules

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

Lightweigh t VGG16	Balanced accuracy and efficiency for vehicle network security	95.4% accuracy	CAN dataset	Requires more compute than plain CNN	Apply quantization for resource-constrained IoT
CNN + Adversarial Training	Improved robustness to evasion attacks	Accuracy drop <6% under FGSM	CAN dataset	Higher training cost	Extend to stronger attacks (PGD, CW)
GNN-based Trust Propagatio n	Modeled relational trust links between devices	94% accuracy	IoT graph datasets	Vulnerable to poisoning attacks	Combine with adversarial defense and DP
Federated Learning with DP [23]	Protected privacy while training trust models	92% accuracy	Federate d IoT data	Utility degradation with strict DP budgets	Balance privacy and utility with adaptive noise
Ensemble CNN + RNN	Leveraged feature locality and temporal order jointly	97.6% accuracy, 96.8% F1	CICIoT	Complex architecture, heavy training time	Lightweight ensemble design for scalability
Robust Training + FGSM Defense [24]	Reduced susceptibility to adversarial perturbations	Accuracy drop 5% (clean vs adversarial)	CAN dataset	Increased inference latency	Optimize defense for real-time IoT traffic
Explainabl e AI (Grad-CAM)	Identified spurious features affecting trust scores	Stable F1 improvement 2%	CICIoT	Limited explainability for recurrent models	Enhance interpretabilit y in hybrid deep models
Edge-ready Quantized CNN	Reduced memory and compute for IoT gateways	93% accuracy, low latency	IoT edge datasets	Slight accuracy loss due to quantization	Explore pruning and hardware acceleration

C. Dataset Used

A. CICIoT dataset [25]

The CIC IoT Dataset 2023 is a complete test set for checking how well security analytics work in real IoT settings. It includes traffic from 105 IoT devices that are linked to each other and working normally. There are 33 different threats, which are organised into seven groups: DDoS, DoS, Reconnaissance, Web-based, Brute Force, Spoofing, and Mirai botnet. The dataset has marked both safe and harmful traffic, so it can be used for both binary and multiclass

Volume 38 No. 7s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

classification tasks. Its size, variety, and reality make it an important tool for creating and testing strong IoT intrusion detection and trust management models.

B. CAN Dataset [26]

The CAN Intrusion Detection Dataset was made to test the security of vehicular networks by recording real contact between vehicles in both normal and hostile situations. Through the OBD-II port, data was gathered from a KIA SOUL, making sure that it was real and true. It has four main states: attack-free state, which is regular CAN traffic; DoS attack, which is caused by repeatedly sending CAN ID 0x000; Fuzzy attack, which uses fake IDs and DATA values; and Impersonation attack, which pretends to be a real node with arbitration ID 0x164. For teaching intrusion detection systems to find and stop a wide range of automotive threats, this dataset is essential.

D. Methodology

The suggested framework for smart and attack-proof trust management is made up of three main methodological parts. In the first part, the CICIoT dataset, which has different IoT network traffic trends, was used to find attacks. The dataset went through a lot of preprocessing steps, such as handling missing values, encoding features, separating labels, and choosing flow-based features. The Synthetic Minority Oversampling Technique (SMOTE) was used to fix the problem of class mismatch. Deep learning classifiers like LSTM, GRU, and a hybrid LSTM–GRU were used, and the models were learnt with an 80:20 split between training and testing. There was testing of both binary classification of attack versus normal traffic and multiclass classification of attacks. The models' success was compared. The structure of the system for managing trust that can't be attacked is shown in Figure 1. It includes preparing datasets, deep neural networks, attack detection, adversarial defence, comparative analysis, and making outputs that can't be attacked so that safe decisions can be made.

In the second part, attack identification was expanded to include the CAN intrusion dataset, which is made up of networks for vehicles and control systems. The data was scaled to a range of 0 to 1 and turned into image-like forms so that the model could use them. A standard CNN and a lightweight VGG16 architecture were both used. The VGG16 [27] was designed to get the most out of structured data for feature extraction. This method made it possible to classify attack types accurately while also making sure that the computer would work quickly enough for real-time settings. The third part was about using the CAN dataset to study attack defence in hostile situations. The Fast Gradient Method [29] was used to make different types of attacks, such as escape, extraction, inference, and poisoning. The models were then tested on both clean and hostile inputs, which showed how resilient they were by comparing how well they did in terms of accuracy and loss.

Received: August 02, 2025

International Journal of Applied Mathematics
Volume 38 No. 7s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

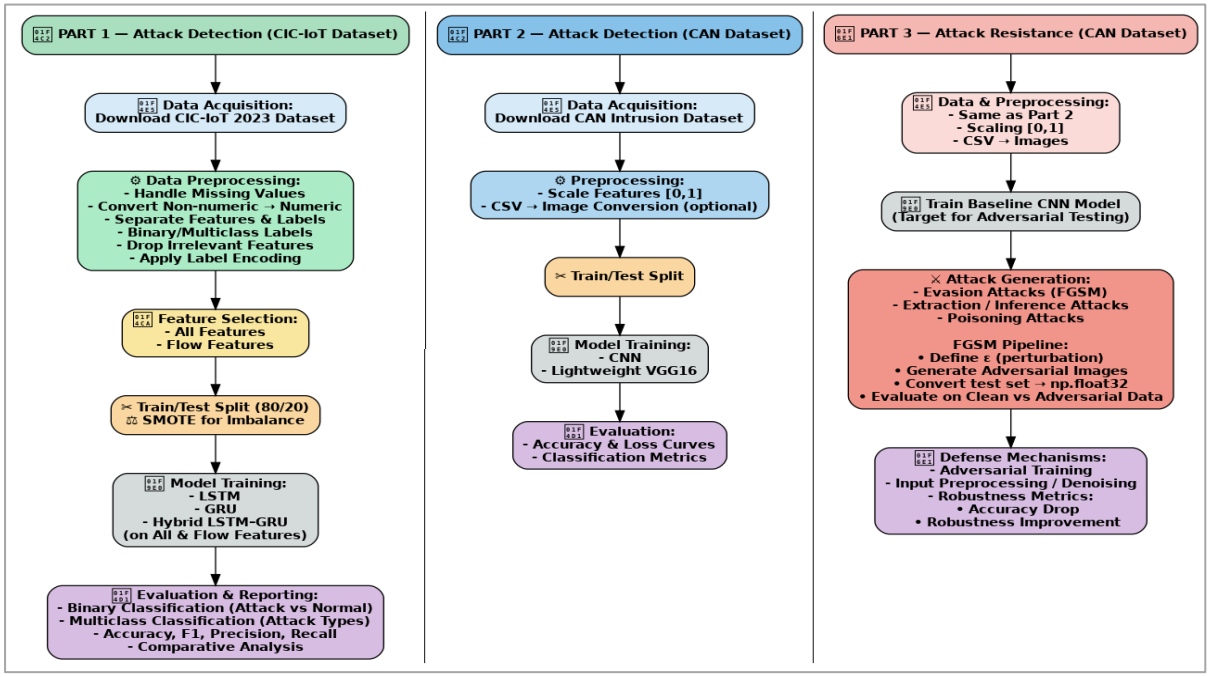


Figure 1: Overview of architecture for attack resistance and intelligent trust management

PART 1 – ATTACK DETECTION - CICIOT DATASET

A. LOAD DATASET

In this study we used CICOT dataset for the attack detection. There are 33 different threats, which are organised into seven groups: DDoS, DoS, Reconnaissance, Web-based, Brute Force, Spoofing, and Mirai botnet. The dataset has marked both safe and harmful traffic, so it can be used for both binary and multiclass classification tasks.

	frame.time	ip.src_host	ip.dst_host	arp.dst.proto_ipv4	arp.opcode	arp.hw.size	arp.src.proto_ipv4
0	6.0	192.168.0.152	0.0	0.0	0.0	0.0	0.0
1	6.0	192.168.0.101	0.0	0.0	0.0	0.0	0.0
2	6.0	192.168.0.152	0.0	0.0	0.0	0.0	0.0
3	6.0	192.168.0.101	0.0	0.0	0.0	0.0	0.0
4	6.0	192.168.0.152	0.0	0.0	0.0	0.0	0.0

Figure 2. Dataset Sample

Figure 2 shows an example of IoT traffic data. It shows source and destination IPs along with ARP-related characteristics that are needed for preprocessing, feature extraction, and then modelling to find attacks.

B. Dataset Preprocessing

The dataset goes through a thorough cleaning step to make sure it is correct and consistent before deep learning models are used for attack-resistant trust management [30]. Finding and dealing with missing values is the first step, since incomplete records can add noise, bias, or errors to model training. The dataset stays reliable by taking out or adding these numbers. The

Received: August 02, 2025

International Journal of Applied Mathematics
Volume 38 No. 7s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

next step is to turn categorical features into numeric formats. This includes changing characteristics like date_time, source_ip, and destination_ip. This is a very important step because neural networks work with numbers and need constant feature encoding to learn well.

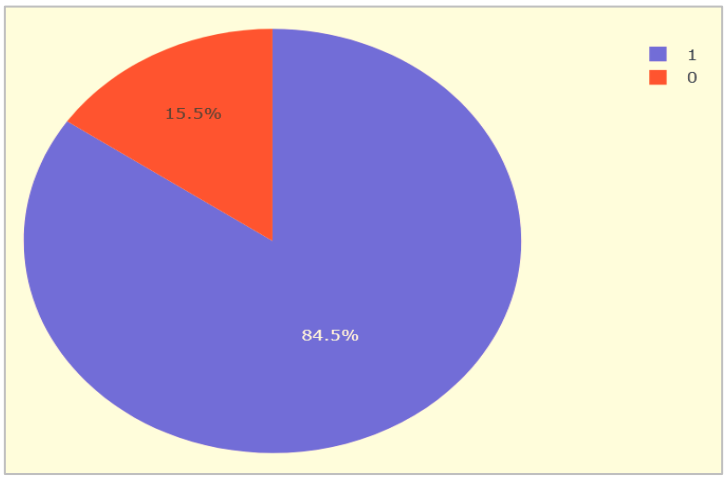


Figure 3. Attack Labels Count

The dataset is then split into features and labels. Features are the traits of the traffic, and labels are the ground truth used for supervised learning tasks. This difference is very important for models to be able to connect input traits to correct attack predictions. To finish, attack cases are marked as either "Attack" or "Normal" to make a structured classification problem. As shown in Figure 3, this binary labelling makes it possible to do initial detection jobs and also gives us a way to move on to multiclass classification. After these organised steps, the dataset is changed into a format that is clean, structured, and machine-readable. This makes sure that deep learning models can find and stop threats as well as possible.

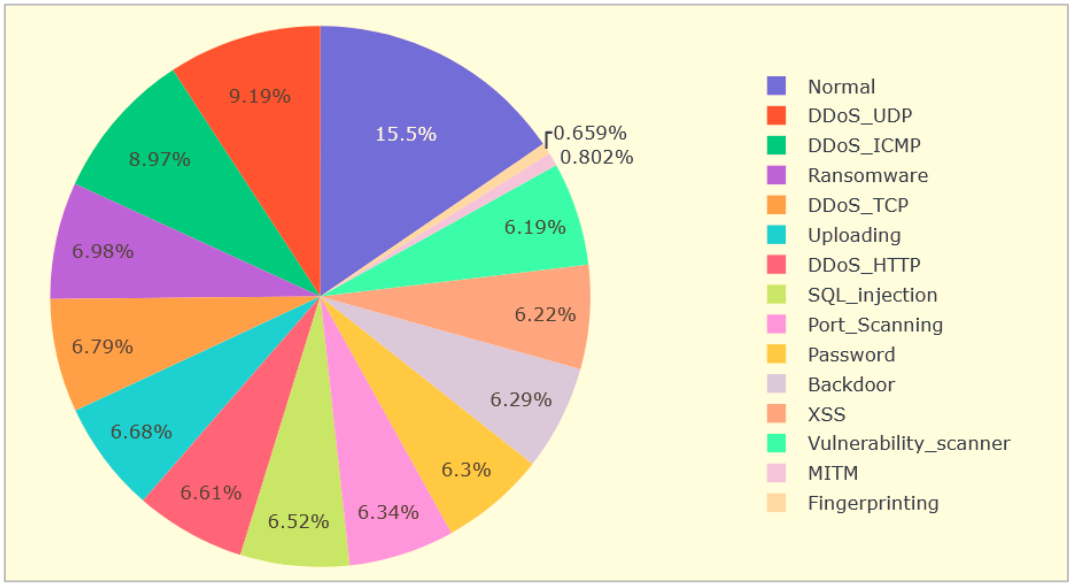


Figure 4. Attack Type Count

Received: August 02, 2025

Volume 38 No. 7s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

Figure 4 shows how the different types of attacks in the dataset are spread out. Normal traffic makes up the biggest portion, followed by a wide range of types, such as DDoS variants, ransomware, scanning, injection, and spoofing-based attacks.

C. Drop Features

A step in the planning process called "dropping features" gets rid of attributes that aren't needed, are duplicated, or are strongly linked to other attributes and make the model more complicated or noisy. This makes learning more efficient, stops models from overfitting, and makes sure they focus on inputs that are important.

Suppose the dataset $D = \{X1, X2, X3, \ldots, Xn\}$,

where Xi are features.

If a feature Xj has low variance or high correlation,

it is removed:

$$D' = D - \{Xj\}, where\ Corr(Xj, Xk) > \tau$$

Here, Corr(Xj, Xk) denotes correlation, and τ is a threshold (e.g., 0.9).

D. Apply Label Encoder

Label encoding is an important step before deep neural networks can handle input features efficiently. It turns categorical values into numeric values. As part of attack-resistant trust management, attack names like "Normal," "DoS," "DDoS," "Ransomware," and so on are turned into integer codes. Such as, "Normal" could be marked as 0, "DoS" as 1, and "DDoS" as 2. This makes sure that the model sees categorical classes as separate numerical groups. This way, it can handle both binary and multiclass classification for finding IoT attacks.

E. Feature Selection

1. Considering all features

When you use all of the dataset's features, you have to keep all of its attributes, such as time stamps, IP addresses, port numbers, protocol names, and flow statistics. This method gives the model as much data as possible, which lets deep neural networks like LSTM, GRU, and hybrid architectures learn trends in both time and context across a wide range of traffic attributes. It is true that training on all features can improve the accuracy of detection and help catch minor attack behaviours, but it can also make computations more difficult and increase the risk of overfitting. Still, testing all features gives researchers in trust management a good starting point for comparing how well different models work.

2. Considering the flow features

Flow features are all about things that have to do with traffic flows, like the number of packets, the length of the flow, byte data, and connection-level information. By limiting the input space to these traffic-centric measures, models can get rid of noise from unnecessary fields (like raw IP addresses or timestamps) and focus on signs of bad behaviour. Flow features are great for both binary and multiclass classification because they show how normal and attack traffic is

Received: August 02, 2025

International Journal of Applied Mathematics
Volume 38 No. 7s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

statistically different. This study found that choosing certain flow features not only made computations faster but also kept accuracy levels high. This shows that trust management systems can be both light and reliable in real-life IoT attack situations.
Figure 4 displays the flow features that can be used for classification. These include TCP, UDP, DNS, MQTT, and MBTCP attributes, which are very important for finding attacks and strange behaviour in IoT network data.

```
# Selecting Flow Features for Classification
flow_features = [
    'tcp.len', 'tcp.flags', 'tcp.seq', 'tcp.ack', 'udp.stream',
    'udp.time_delta','dns.qry.name', 'dns.qry.qu', 'dns.retransmission',
    'dns.retransmit_request','mqtt.conflag.cleansess', 'mqtt.conflags',
    'mqtt.hdrflags', 'mqtt.len','mqtt.msgtype', 'mqtt.proto_len',
    'mqtt.topic_len', 'mqtt.ver', 'mbtcp.len'
]
```

Figure 4: Flow Features Selection

F. Apply Smote for Data Imbalance

Attack classes in real-world IoT and CAN datasets are often very uneven. Some classes, like DoS or DDoS, have thousands of examples, while rare attacks, like spoofing or impersonation, are not shown at all. This imbalance can make deep learning models favour classes with a lot of members, which can lead to bad memory and wrong classification of attacks from minority groups. SMOTE, or the Synthetic Minority Over-sampling Technique, is used to fix this problem, label representation in figure 5.

```
Attack_label
1    21356
0     3892
Name: count, dtype: int64
Attack_label
1    21356
0    21356
Name: count, dtype: int64
```

Figure 5. Attack Labels Count before and after applying SMOTE

To make SMOTE work, it doesn't just copy current records; instead, it creates new samples of the minority class. The algorithm picks one or more of a minority class instance's closest neighbours and adds new points along the line segments that link them. This makes the class distribution more even by including more minority strikes in the dataset. For instance, if there aren't many "Ransomware" traffic instances compared to "Normal" or "DDoS" instances, SMOTE creates fake ransomware samples to make the dataset more regular. Trust management models like LSTM, GRU, and CNNs learn better across all types of attacks when SMOTE is used in preparation. This makes the system more accurate overall and also greatly improves memory and F1-scores for minority classes. This makes sure that the system can reliably find rare but dangerous attack types, which is important for managing trust in the IoT.

Received: August 02, 2025

International Journal of Applied Mathematics
Volume 38 No. 7s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

SMOTE Algorithm step by step:

1. Identify minority classes:

   Find classes with fewer samples compared to the majority.

2. Determine samples to generate:

   Calculate how many synthetic samples are needed for balance.

3. Select nearest neighbors:

   For each minority sample, find k nearest neighbors within the same class.

4. Random neighbor selection:

   Randomly pick one neighbor from the k nearest neighbors.

5. Generate interpolation factor:

   Choose λ randomly in the range [0,1].

6. Create synthetic sample:

   $New\ sample\ =\ xi\ +\ \lambda\ *\ (xj\ -\ xi),$

7. Add synthetic samples:

   Combine generated samples with the original dataset to achieve balanced class distribution.

BEFORE SMOTE

```
7     3898
4     2317
2     2237
1     1718
10    1686
11    1668
3     1655
14    1650
9     1623
8     1612
12    1607
13    1606
0     1591
6      213
5      167
Name: Attack_type, dtype: int64
```

AFTER SMOTE

```
2     3898
8     3898
13    3898
3     3898
11    3898
14    3898
12    3898
9     3898
7     3898
10    3898
4     3898
0     3898
1     3898
5     3898
6     3898
Name: Attack_type, dtype: int64
```

Figure 6. Attack Types Count before and after applying SMOTE

Figure 6 shows how the different types of attacks were spread out before and after SMOTE was used. At first, there was a big difference between the classes, and strikes by minorities were not well represented. All classes were equalised to 3898 samples after SMOTE, which made sure that training was fair.

G. Apply Deep Learning Classification Model

1. LSTM [33]

Received: August 02, 2025

International Journal of Applied Mathematics
Volume 38 No. 7s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

Long-Short-Term Memory (LSTM) networks are a special kind of recurrent neural network (RNN) that can pick up on long-range relationships in sequential data. LSTM is very important for looking at the timing of traffic flows because it helps with the problem of smart and attack-proof trust management in IoT and vehicle settings. IoT network data flows in a straight line, and bad things happen in patterns that build over time. For example, repeated requests during denial-of-service (DoS) attacks or slow poking during reconnaissance attacks. LSTM solves this problem by using memory cells and gating systems (input, forget, and output gates) that control how much old data is kept, changed, or thrown away.

The CICIoT dataset is used in this study to use LSTM for both binary and multiclass classification of attack and standard traffic. The LSTM model learns hidden patterns that separate bad behaviour from good communication by looking at things like TCP sequence behaviour, UDP streams, and flow data. LSTM is better at finding complex attacks because it can find small changes that happen over a number of time steps, unlike standard classifiers. The figure 7 represent the model summary for LSTM.

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                 Output Shape              Param #
=================================================================
 cu_dnnlstm (CuDNNLSTM)       (None, 32, 300)           363600

 dropout (Dropout)            (None, 32, 300)           0

 cu_dnnlstm_1 (CuDNNLSTM)     (None, 32, 300)           722400

 cu_dnnlstm_2 (CuDNNLSTM)     (None, 30)                39840

 dropout_1 (Dropout)          (None, 30)                0

 dense (Dense)                (None, 8)                 248

=================================================================
Total params: 1126088 (4.30 MB)
Trainable params: 1126088 (4.30 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Figure 7. LSTM Model Summary

Algorithm 1: LSTM-Based Attack-Resistant Trust Management

Input:
- Sequence of features $X = \{x_t\}, t = 1, \ldots, W, x_t \in R^d$
- Labels y ∈ {0, 1, …, C-1}

Output:
- Predicted attack class (y_hat)
- Trust score (T)

Steps:
1. Initialize Parameters
   – $Weights\ W\_i, W\_f, W\_c, W\\_o$

Received: August 02, 2025

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

– $Recurrent\ weights\ U\_i, U\_f, U\_c, U\_o$
– $Biases\ b\_i, b\_f, b\_c, b\_o$
– $Hidden\ state\ h\_0 = 0, Cell\ state\ c\_0 = 0$

2. Compute LSTM Gates for each time step t = 1,...,W

$$i\_t = \sigma(W\_i\, x\_t + U\_i\, h\_(t-1) + b\_i)$$
$$f\_t = \sigma(W\_f\, x\_t + U\_f\, h\_(t-1) + b\_f)$$
$$c{\sim}\_t = tanh(W\_c\, x\_t + U\_c\, h\_(t-1) + b\_c)$$
$$c\_t = f\_t \odot c\_(t-1) + i\_t \odot c{\sim}\_t$$
$$o\_t = \sigma(W\_o\, x\_t + U\_o\, h\_(t-1) + b\_o)$$
$$h\_t = o\_t \odot tanh(c\_t)$$

3. Obtain Final Hidden Representation

$$z = h\_W$$

4. Classification Layer

– $Binary: p\_attack = \sigma(w\text{^}T\, z + b)$

$$p_c = \frac{[exp(w_c^T z + b_c)]}{[\Sigma_j exp(w_j^T z + b_j)]}$$

5. Loss Function

$$L = -\left(\frac{1}{N}\right) \Sigma_n \Sigma_c w_c * 1[y_n = c] * \log(p_{n,c})$$

2. GRU [28]

Gated Recurrent Unit (GRU) is a simplified version of LSTM that is used to find temporal relationships in IoT traffic data that is sent one piece at a time. Because it has fewer parameters, it trains faster, uses less memory, and finds oddities more accurately. This makes it a good choice for real-time trust management systems that are resistant to attacks. The model summary illustrate in figure 8

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 cu_dnngru_3 (CuDNNGRU)      (None, 32, 300)           272700

 dropout_4 (Dropout)         (None, 32, 300)           0

 cu_dnngru_4 (CuDNNGRU)      (None, 32, 200)           301200

 cu_dnngru_5 (CuDNNGRU)      (None, 200)               241200

 dropout_5 (Dropout)         (None, 200)               0

 dense_2 (Dense)             (None, 8)                 1608

=================================================================
Total params: 816708 (3.12 MB)
Trainable params: 816708 (3.12 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Received: August 02, 2025

International Journal of Applied Mathematics
Volume 38 No. 7s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

Figure 8. GRU Model Summary

Algorithm: GRU-Based Attack Detection

1. *Initialize Parameters*
   - *Weights*: W_z, W_r, W_h
   - *Recurrent Weights*: U_z, U_r, U_h
   - *Biases*: b_z, b_r, b_h
   - *Hidden state* $h\_0 = 0$
2. *Update Gate*

   $z_t = \sigma(W_z x_t + U_z h_{t-1} + b_z)$
3. *Reset Gate*

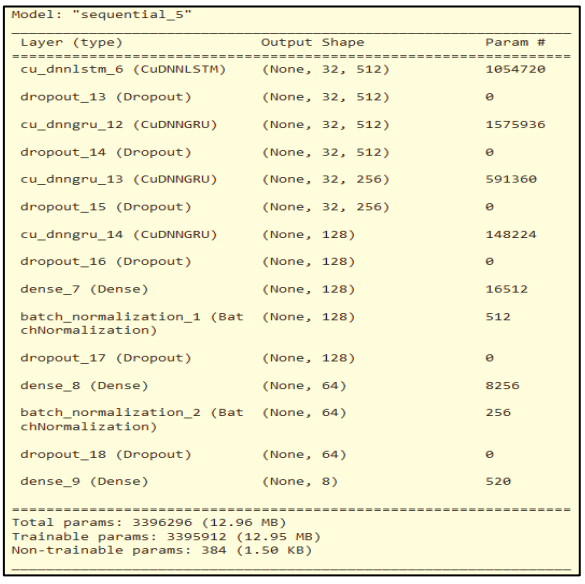
   $r_t = \sigma(W_r x_t + U_r h_{t-1} + b_r)$
4. *Candidate Hidden State*

   $h\sim_t = \tanh(W_h x_t + U_h(r_t \odot h_{t-1}) + b_h)$
5. *Hidden State Update*

   $h_t = (1 - z_t) \odot h_{t-1} + z_t \odot h\sim_t$
6. *Classification Layer*
   - *Binary*: $p_{attack} = \sigma(w^T h_T + b)$
   - *Multiclass*: $p\_c = exp(w\_c\text{^}T\ h\_T + b\_c) / \Sigma\_j\ exp(w\_j\text{^}T\ h\_T + b\_j)$

3. HYBRID LSTM – GRU

The hybrid LSTM–GRU design balances the benefits of LSTM's ability to store long-term memories and GRU's ability to work efficiently with little power. This makes it a good choice for finding IoT attacks. This hybrid model captures complex temporal patterns, makes classification more accurate, cuts down on overfitting, and speeds up convergence. This makes it a great choice for strong, attack-resistant trust management in IoT settings that are always changing. The model running summary is illustrate in figure 9.



```
Model: "sequential_5"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 cu_dnnlstm_6 (CuDNNLSTM)    (None, 32, 512)           1054720

 dropout_13 (Dropout)        (None, 32, 512)           0

 cu_dnngru_12 (CuDNNGRU)     (None, 32, 512)           1575936

 dropout_14 (Dropout)        (None, 32, 512)           0

 cu_dnngru_13 (CuDNNGRU)     (None, 32, 256)           591360

 dropout_15 (Dropout)        (None, 32, 256)           0

 cu_dnngru_14 (CuDNNGRU)     (None, 128)               148224

 dropout_16 (Dropout)        (None, 128)               0

 dense_7 (Dense)             (None, 128)               16512

 batch_normalization_1 (Bat  (None, 128)               512
 chNormalization)

 dropout_17 (Dropout)        (None, 128)               0

 dense_8 (Dense)             (None, 64)                8256

 batch_normalization_2 (Bat  (None, 64)                256
 chNormalization)

 dropout_18 (Dropout)        (None, 64)                0

 dense_9 (Dense)             (None, 8)                 520

=================================================================
Total params: 3396296 (12.96 MB)
Trainable params: 3395912 (12.95 MB)
Non-trainable params: 384 (1.50 KB)
_________________________________________________________________
```

Figure 9. Hybrid LSTM - GRU Model Summary

Received: August 02, 2025

International Journal of Applied Mathematics
Volume 38 No. 7s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

Algorithm: Hybrid LSTM–GRU for Attack-Resistant Trust Management

Input:

- Sequence of features X = {x_t}, t = 1,...,W, x_t ∈ R^d

- Labels y ∈ {0, 1, …, C-1}

Output:

- Predicted attack class (ŷ)

- Trust score (T)

Steps:

1. Initialize Parameters

$-\ LSTM\ weights: W\_i, W\_f, W\_c, W\_o;\ U\_i, U\_f, U\_c, U\_o$

$-\ GRU\ weights: W\_z, W\_r, W\_h;\ U\_z, U\_r, U\_h$

$-\ Hidden\ states: h\_LSTM0 = 0, c\_0 = 0, h\_GRU0 = 0$

2. LSTM Processing (for each time step t = 1,…,W)

$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i)$$

$$f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f)$$

$$c\sim_t = \tanh(W_c x_t + U_c h_{t-1} + b_c)$$

$$c_t = f_t \odot c_{t-1} + i_t \odot c\sim_t$$

$$o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o)$$

$$h_{LSTM_t} = o_t \odot \tanh(c_t)$$

3. GRU Processing (for each time step t = 1,…,W)

$$z_t = \sigma(W_z h_{LSTM_t} + U_z h_{GRU_{t-1}} + b_z)$$

$$r_t = \sigma(W_r h_{LSTM_t} + U_r h_{GRU_{t-1}} + b_r)$$

$$h\sim_{GRU_t} = \tanh(W_h h_{LSTM_t} + U_h(r_t \odot h_{GRU_{t-1}}) + b_h)$$

$$h_{GRU_t} = (1 - z_t) \odot h_{GRU_{t-1}} + z_t \odot h\sim_{GRU_t}$$

4. Feature Fusion

- Combine final hidden states:

$$z_{final} = [h_{LSTM_W} \oplus h_{GRU_W}]$$

5. Classification Layer

$$-\ Binary: p_{attack} = \sigma(w^T z_{final} + b)$$

$$-\ Multiclass: p\_c = exp(w\_c\hat{}T\ z\_final + b\_c) / \Sigma\_j\ exp(w\_j\hat{}T\ z\_final + b\_j)$$

Received: August 02, 2025

International Journal of Applied Mathematics
Volume 38 No. 7s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

6. Trust Score and Decision

- Trust score: T = 1 - p_attack

- Predict attack if p_attack ≥ τ

PART 2 – ATTACK DETECTION - CAN DATASET

A. Load Dataset

The CAN breach Dataset is a standard set of data used to test how well breach detection works in networks inside vehicles. It was taken from a real KIA SOUL using the OBD-II port, which makes it real and legitimate. There are different types of attacks in the dataset, such as DoS attacks (using a lot of 0x000 CAN IDs), Fuzzy attacks (using fake IDs and data), and Impersonation attacks (looking like real nodes with ID 0x164). It also gives you a state where there are no attacks, which is like normal traffic. This dataset is very important for training and testing deep learning models that can handle trust in automotive settings in a way that is reliable and hard to hack.

	CAN ID	DATA[0]	DATA[1]	DATA[2]	DATA[3]	DATA[4]	DATA[5]	DATA[6]	DATA[7]	Label
0	1201	41	39	39	35	0	0	0	154	R
1	809	64	187	127	20	17	32	0	20	R
2	1349	216	0	0	136	0	0	0	0	R
3	1201	41	39	39	35	0	0	0	154	R
4	2	0	0	0	0	0	3	2	228	R

Figure 10. CAN Dataset Sample

A piece from the CAN Intrusion Dataset is shown in Figure 10. It has CAN IDs, titles, and data bytes that go with them. It shows how raw in-vehicle data frames are set up to find attacks and sort them into groups.

B. Data Preprocessing

Preprocessing is an important step that improves both model accuracy and computational speed and is needed for intrusion detection to work well in CAN networks. In the first step, normalization methods like min-max scaling are used to change all of the features' scales to [0,1]. This makes sure that different traits, such as CAN IDs and data bytes, which come from different numeric ranges are shown in the same way. Standardizing the input scale makes the learning process more stable, lowers gradient changes, and speeds up convergence while training a deep neural network.

Received: August 02, 2025

International Journal of Applied Mathematics
Volume 38 No. 7s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

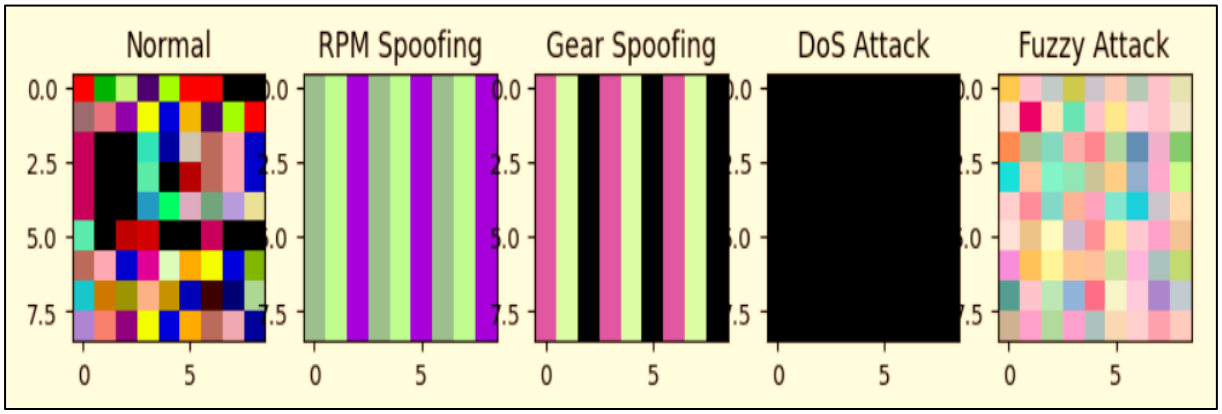


Figure 11. CSV to Image Conversion

In the second step, pictures are made for each class from the data that has already been processed. CAN message frames, which are shown as numerical vectors, are changed into matrices that look like 2D images, as represent it in figure 11. The intensities of the pixels match to the values of the features. This change makes it possible for lightweight designs like VGG16 and convolutional neural networks (CNNs) to find patterns in space across classes [34]. This kind of visual representation shows small changes between normal, DoS, fuzzy, and impersonation traffic, which makes classification much more accurate.

C. Apply Deep Learning Model

1. CNN [31]

Convolutional Neural Networks (CNNs) are very good at finding spatial and structural trends in data, which means they can be used to find intrusions in the CAN dataset. CNNs use convolutional filters to find localised connections between data bytes after preprocessing and turning CAN message frames into structures that look like images. When compared to normal data, these filters can tell the difference between different types of attacks, like DoS, fuzzy, and impersonation. By pooling layers together, the number of dimensions is reduced while important traits are kept. This makes the computation faster. Lastly, fully connected layers sort the data into the right groups. CNN models are very good at finding small differences in data structures on the CAN dataset. This makes them useful for managing trust in real time and protecting against attacks in vehicle settings.

Algorithm: CNN for Intrusion Detection on CAN Dataset

Input:

- Preprocessed CAN frames (reshaped as image-like matrices)

- Labels (Normal, DoS, Fuzzy, Impersonation)

Output:

- Predicted attack class (ŷ)

Steps:

1. Input Transformation

International Journal of Applied Mathematics
Volume 38 No. 7s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

$$X \in R^{H \times W \times C}$$

Where H = height, W = width, C = channels (1 for grayscale)

2. Convolution Operation

$$Z^l = X^{l-1} * W^l + b^l$$

Where * denotes convolution, W^(l) are filters, b^(l) is bias

3. Activation Function

$$A^l = ReLU(Z^l) = \max(0, Z^l)$$

4. Pooling Layer

$$P^l = \max_{pool}(A^l, k)$$

Where k is pooling size

5. Flatten Layer

$$F = flatten(P^l)$$

Converts feature maps into 1D vector

6. Fully Connected Layer

$$y = softmax(W_{fc} \cdot F + b_{fc})$$

$$where\ softmax(p\_c) = exp(p\_c) / \Sigma\_j\ exp(p\_j)$$

7. Loss Function (Cross-Entropy)

$$L = -\left(\frac{1}{N}\right) \Sigma_n \Sigma_c y_n, c * \log(\hat{y}_n, c)$$

8. Prediction

$$Predicted\ class = argmax(y)$$

2. Lightweight VGG16

Lightweight VGG16 is an improved version of the original VGG16 deep convolutional network. It is made to be easier to compute while still being able to extract features well. It takes normalised CAN frames and turns them into image-like inputs for the CAN breach dataset. It then uses stacked convolution and pooling layers to allow hierarchical feature learning. The lighter version has fewer factors and better layers than the heavier original model, so it can be used for real-time vehicle intrusion detection. It accurately picks up attack-specific patterns like fuzzy or impersonation traffic while lowering the amount of memory and computation needed. This makes it possible for scalable and attack-resistant trust management in car settings.

PART 3 – ATTACK RESISTANCE - CAN DATASET

A. Dataset Load

Received: August 02, 2025

International Journal of Applied Mathematics
Volume 38 No. 7s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

A very important part of testing both ways to find and stop hostile attacks in in-vehicle networks is the CAN Intrusion Dataset. It was taken from a real KIA SOUL through the OBD-II port, which made sure that the CAN bus flow was real and accurate. The dataset has four separate states: regular communication that isn't under attack, DoS attacks (which insert CAN ID 0x000 very often), fuzzy attacks (which use fake CAN IDs and data values), and impersonation attacks (which send malicious messages with arbitration ID 0x164). This dataset is used in the attack resistance step to first turn CAN features into normalised [0,1] ranges and then make samples that look like images, sample input data shown in figure 12. To see how well CNN and VGG16 designs protect against attacks like evasion, poisoning, inference, and extraction, these inputs are then used to test how robust models are against adversarial manipulation.

	CAN ID	DATA[0]	DATA[1]	DATA[2]	DATA[3]	DATA[4]	DATA[5]	DATA[6]	DATA[7]	Label
0	1201	41	39	39	35	0	0	0	154	R
1	809	64	187	127	20	17	32	0	20	R
2	1349	216	0	0	136	0	0	0	0	R
3	1201	41	39	39	35	0	0	0	154	R
4	2	0	0	0	0	0	3	2	228	R

Figure 12. CAN Dataset Sample

B. Data Preprocessing

During the attack resistance phase, the CAN dataset is first preprocessed by using min-max scaling to bring all of its features to the same scale ([0,1]). This makes sure that attributes like CAN IDs and data fields, whose ranges used to change, are all shown in the same way. This stops bias in training. The next step is to turn records that are stored in CSV files into images. As a feature vector, each CAN message is changed into a 2D grid where pixel strength is equal to normalized feature values. This change from CSV to picture is shown in Figure 13. It lets CNN and VGG16 models use spatial feature learning for strong intrusion and adversarial resistance analysis.

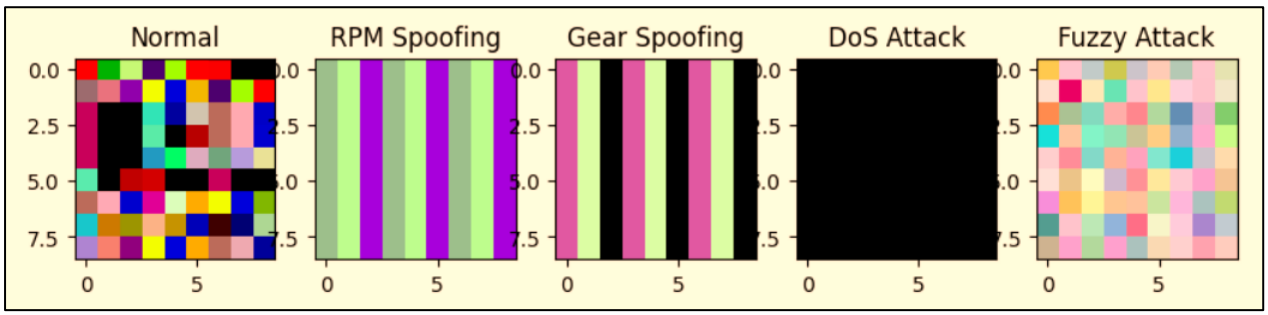


Figure 13. CSV to Image Conversion

C. GENERATE CNN MODEL

Received: August 02, 2025

Volume 38 No. 7s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

After preparation, a Convolutional Neural Network (CNN) [32] model is made to test how resistant the CAN dataset is to attacks. The model starts with convolutional layers that use filters to pull out spatial data from the CAN representations that look like images. After each convolutional process, activation functions (usually ReLU) and pooling layers that lower the number of dimensions while keeping important data are added. The feature maps that are made are then flattened into vectors, which are then sent through fully linked layers to be categorized. Summary of model illustrate in figure 14. Last but not least, a softmax layer gives out odds for normal, DoS, fuzzy, and impersonation classes. This CNN model makes detection accurate, quick, and resistant to threats from other parties.

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 222, 222, 32)      320

 max_pooling2d (MaxPooling2  (None, 111, 111, 32)      0
 D)

 conv2d_1 (Conv2D)           (None, 109, 109, 64)      18496

 max_pooling2d_1 (MaxPoolin  (None, 54, 54, 64)        0
 g2D)

 flatten (Flatten)           (None, 186624)            0

 dense (Dense)               (None, 10)                1866250

=================================================================
Total params: 1885066 (7.19 MB)
Trainable params: 1885066 (7.19 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Figure 14: Summary of CNN Model

E. Result And Discussion

a. Attack Classification (Binary)

In Figure 15, represent how well an LSTM model did in terms of accuracy and loss during the training and validation stages. In the top subplot, training accuracy starts at about 84% and keeps going up until it goes over 96% in later epochs. Validation accuracy changes more clearly, going down sometimes but mostly staying close to training accuracy. This means that generalisation is good with only slight instability. In the bottom section, training loss drops sharply from 0.35 to less than 0.1, which means that learning is going well. Validation loss also goes down, but there are some variations that show it is sensitive to certain epochs. Overall, the plots show that the LSTM model is very accurate and converges well, but the changes in validation show that we need to keep an eye out for possible overfitting.

Received: August 02, 2025

International Journal of Applied Mathematics
Volume 38 No. 7s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

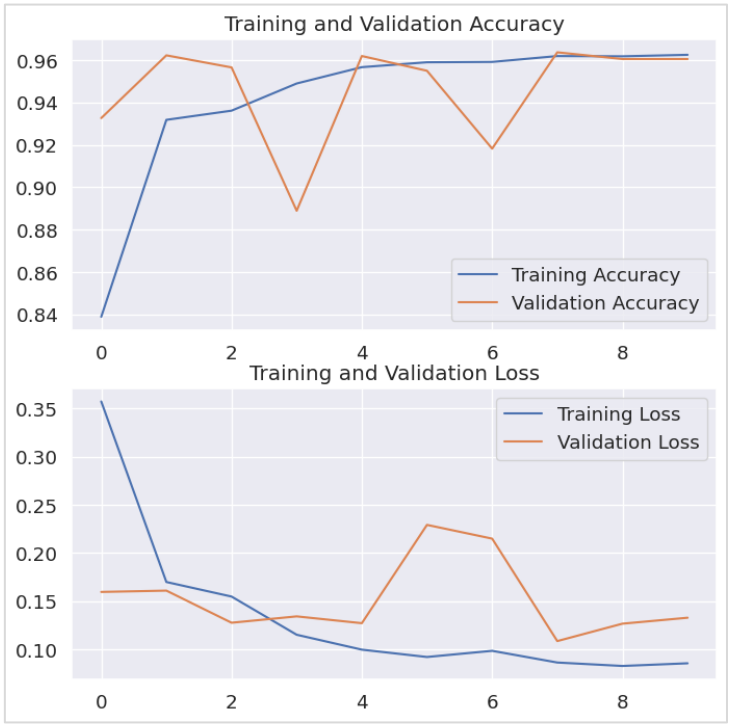


Figure 15: LSTM accuracy training and validation loss

There are 822 true negatives and 5241 true positives that the model accurately predicted in Figure 15. There were 164 false positives and 85 false negatives, which shows that the model was very good at predicting the future with few classification mistakes.

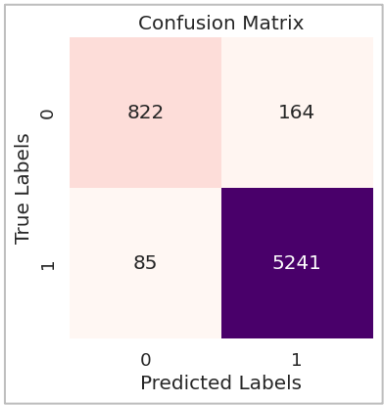


Figure 16: Confusion matrix of LSTM

Figures 17 and 18 show how well the GRU model worked by showing its confusion matrix, loss curves, and training/validation accuracy. The accuracy plot in Figure 17 shows that both training and validation accuracy got better very quickly, hitting over 96% in just a few epochs and then staying that high after that. This shows how well the GRU model can learn about temporal relationships without becoming too perfect. The loss curves for training and validation both drop sharply, and losses settle around 0.07 to 0.1, which shows that the model is very good at generalisation. The fact that the validation curve stays close to the training curve shows that the model stays the same when it comes to data it hasn't seen before.

Received: August 02, 2025

International Journal of Applied Mathematics
Volume 38 No. 7s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

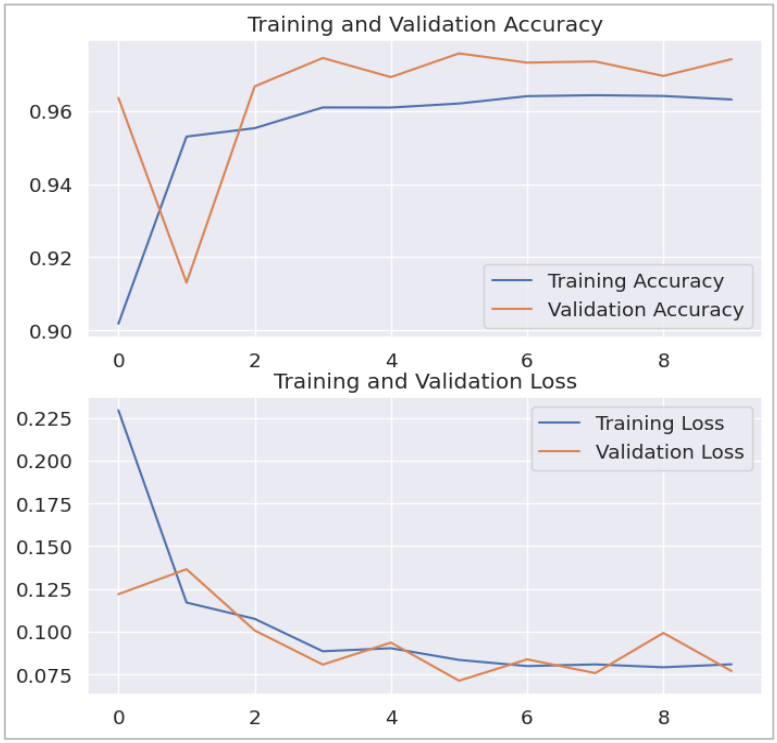


Figure 17: GRU accuracy training and validation loss

Figure 18 shows the confusion matrix results. The GRU got 799 true negatives and 5346 true positives, which means it classified things very accurately. There have been very few mistakes, with only 139 fake positives and 28 false negatives, showing that the system is very sensitive and very specific. The model rarely misses true positives, as shown by the small number of fake negatives. This makes it perfect for situations where recall is very important. On the other hand, the relatively low rate of false positives makes it even more reliable for real-world jobs. The accuracy/loss plots and confusion matrix show that the GRU model works well, converges quickly, and can handle precision and recall well, which proves that it is good at sequence modelling tasks.

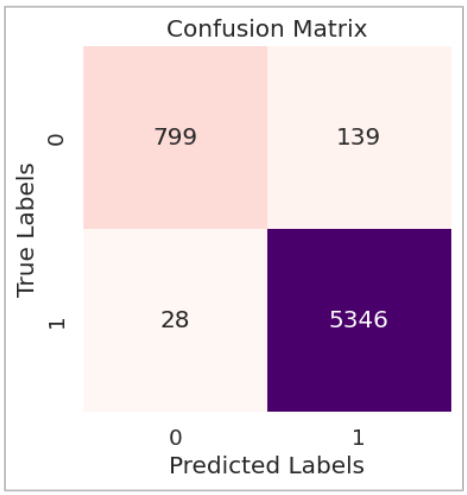


Figure 18: Confusion matrix of GRU

Received: August 02, 2025

International Journal of Applied Mathematics
Volume 38 No. 7s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

The success of the Hybrid LSTM–GRU model is shown in Figures 19 and 20. This model takes the best parts of both architectures and combines them to make sequence learning better. The accuracy for training and confirmation, as well as the loss curves for each, are shown in Figure 19. The accuracy plot shows that both the training and validation accuracy go up quickly. After just a few epochs, they both settle at around 96–97%. This means the mixed model learns quickly and correctly applies what it has learnt to new data. This is shown even more clearly by the loss curves, which show that both training and validation losses drop sharply in the first two epochs and then stay very low. Validation and training curves that are close to each other show that there isn't much overfitting and that regularisation is working well.

The Hybrid LSTM–GRU gets 797 true negatives and 5364 true positives in the confusion matrix shown in Figure 20, which shows that it is very good at classifying. There are only 141 fake positives and 10 false negatives, which is a very low number of mistakes. The model's ability to catch almost all positive cases is emphasised by the very small number of fake negatives. This is important in high-stakes situations where missing true events could be expensive. A low number of wrong positives is another sign of high precision. Overall, the Hybrid LSTM–GRU performs well, with high accuracy, fast convergence, and strong generalisation. This proves that combining LSTM and GRU makes predictions more reliable and improves the speed of sequential tasks.

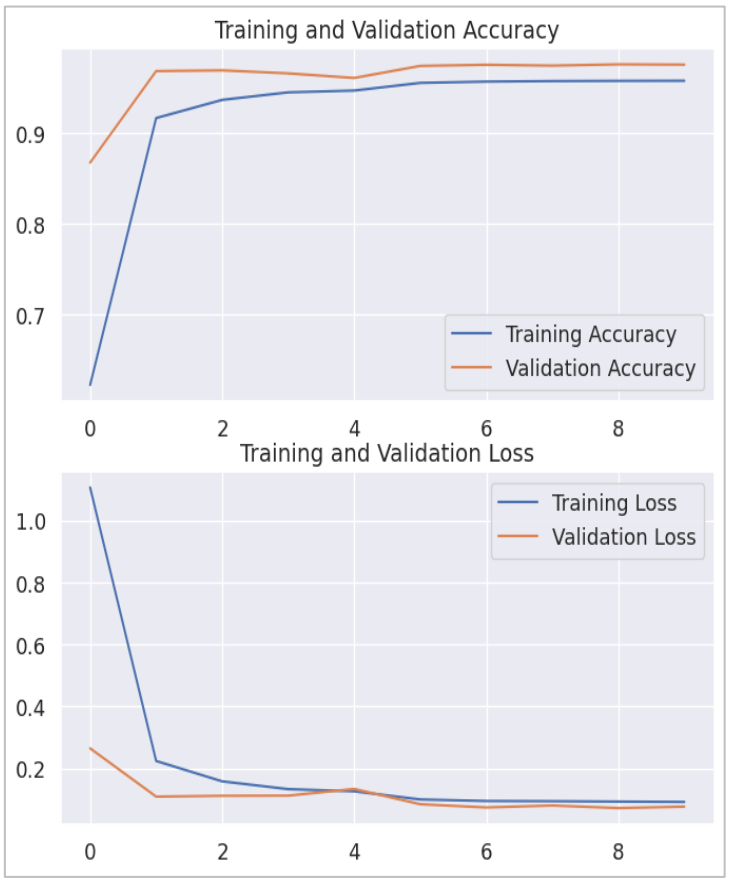


Figure 19: Hybrid LSTM - GRU accuracy training and validation loss

Received: August 02, 2025

International Journal of Applied Mathematics
Volume 38 No. 7s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

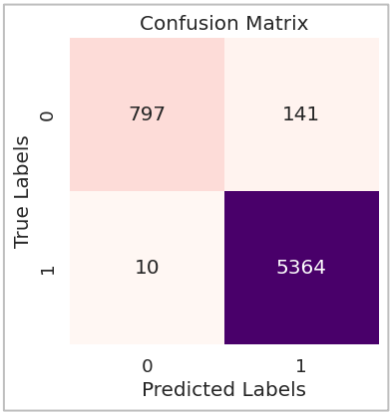


Figure 20. Confusion Matrix of Hybrid LSTM - GRU Graphs

b. Attack Type Classification (MULTICLASS CLASSIFCIATION)

Figure 21 shows the accuracy and loss curves for the LSTM and GRU models when they are used to sort attack types into multiple groups. In terms of convergence, stability, and generalisation to new data, these graphs tell us a lot about how well both models work. You can see how fast each model learns patterns in the dataset and how well this learning shows up in validation performance on the training and validation accuracy curves. LSTM is known for being able to capture long-term dependencies. It usually gets better over time, hitting high levels of accuracy over time. Because GRU is easier to compute, it often converges faster, as shown by the fact that its starting accuracy gain is steeper. When you look at the two models side by side, you can see that they are both very good at generalisation. This is because the validation accuracy closely follows the training accuracy, with very little difference between the two. This means that neither model is severely overfitting.

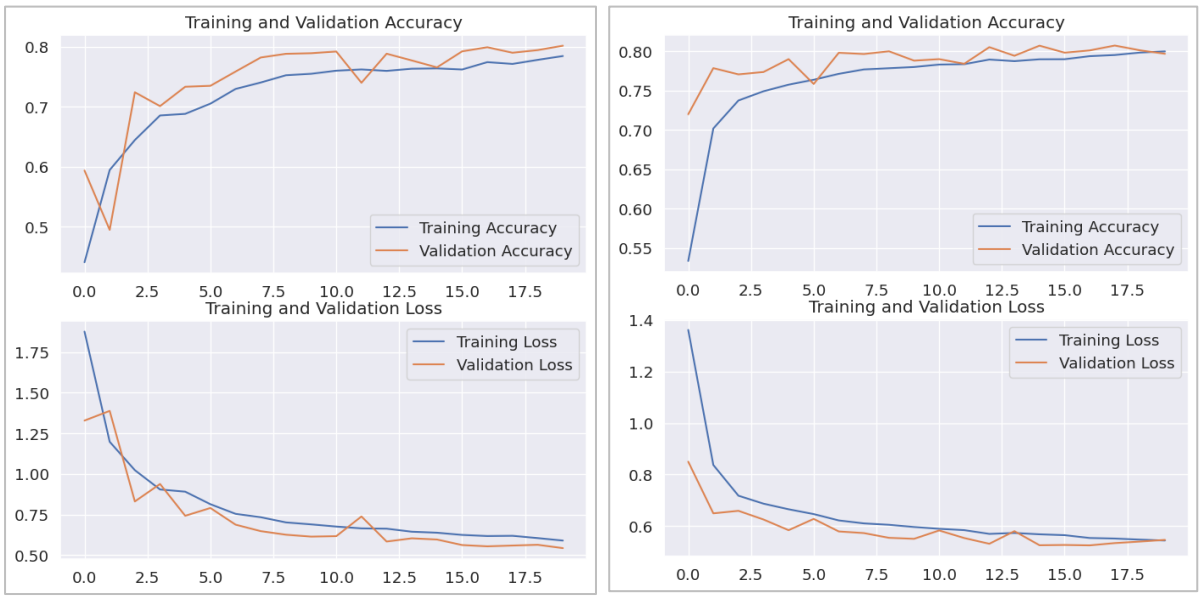


Figure 21. LSTM and GRU Accuracy and Loss Curve

During the first few epochs, both training and validation losses drop by a lot on the loss curves. This shows that the system is learning how to make good decisions across a number of attack

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

classes. Loss values becoming stable in later epochs are a sign of convergence. GRU usually gets to this point a little earlier because it has fewer parameters and updates more quickly. The smoother but slightly slower drop in loss of LSTM shows how well it can capture more complex temporal dependencies. This is often helpful for attack detection where patterns may span across longer sequences of data.

Figure 21 shows that both LSTM and GRU are very good at multiclass attack classification tasks, getting very accurate results while keeping loss values low. The GRU is better at training quickly and efficiently, while the LSTM is better at learning long-term traits and being more stable. The results show that recurrent neural designs are good at finding different kinds of attacks, which means they will work well in real-world intrusion detection systems.

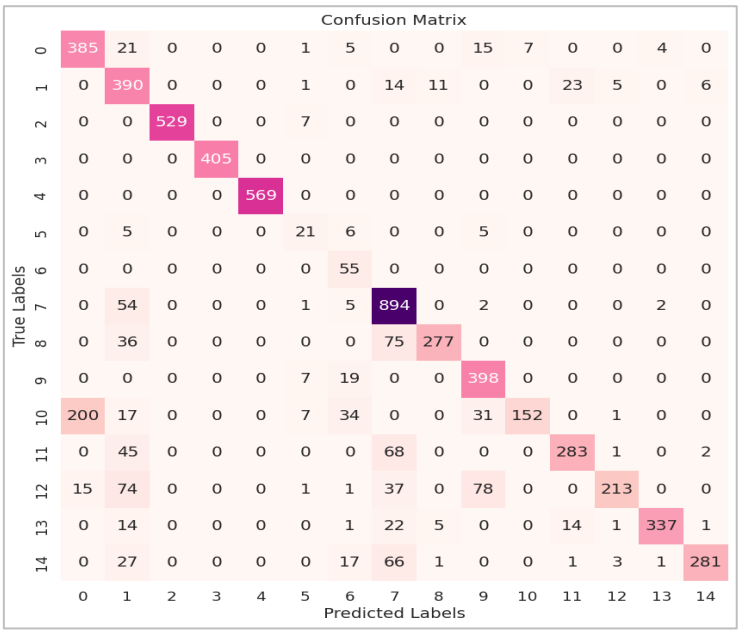


Figure 22. Confusion Matrix of Hybrid LSTM – GRU Model (Attack Type Classification)

c. Comparative Analysis

We looked at LSTM, GRU, and Hybrid LSTM–GRU models in Table 2 and gave them scores on a number of important performance measures. The results clearly show that all three models are good at classifying things, but the hybrid method always does better than the single architectures. The accuracy numbers get better over time, going from 96% for LSTM to 97% for GRU. They peak at 98% for the hybrid model, which shows how well it can tell the difference between attack and non-attack labels. Precision follows a similar pattern, with the hybrid model getting the best score of 98%.

Table 2: Performance Comparison of Algorithms with Respect to Attack Labels (Binary Classification)

Parameters	**LSTM (%)**	**GRU (%)**	**Hybrid LSTM–GRU (%)**

International Journal of Applied Mathematics
Volume 38 No. 7s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

Accuracy	96	97	98
Precision	94	97	98
Recall	91	92	92
F1-Score	92	95	96

This shows that it is better at reducing false positives than both LSTM (94%) and GRU (97%). It's interesting that recall rates stay pretty close across all three models. GRU and the mixed approach are both at 92%, which is a little higher than LSTM's 91%. This means that all models are good at finding real attacks, but the hybrid design does the best job of catching false positives and avoiding wrong classifications. Another proof of this performance advantage comes from the F1-Score, which balances accuracy and recall: the mixed model gets a score of 96%, higher than both GRU (95%) and LSTM (92%). Overall, the study shows that combining LSTM and GRU makes the most of their unique strengths, leading to the best results in binary attack classification (See the performance analysis in figure 23).

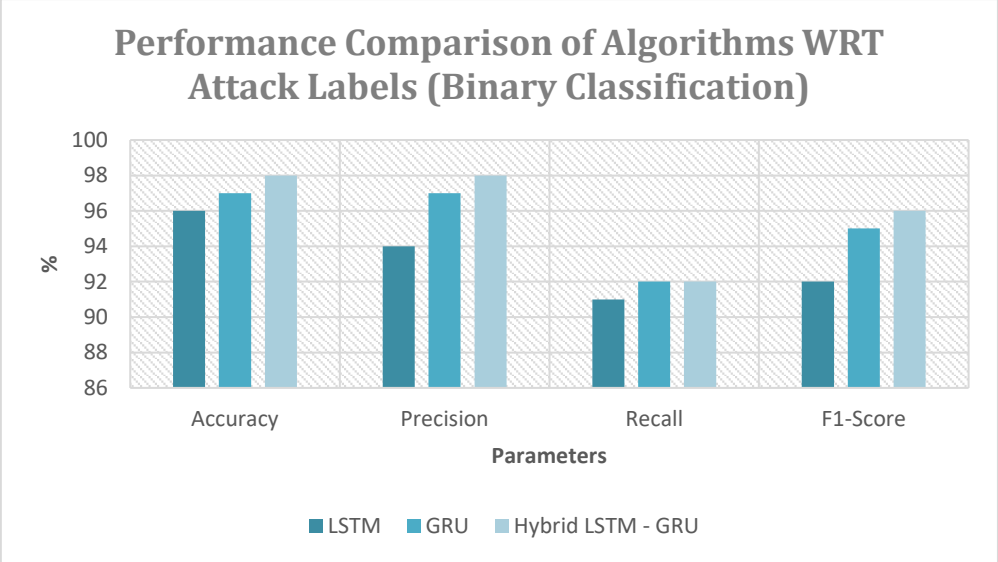


Figure 23. Performance Comparison of Algorithms WRT Attack Labels (Binary Classification)

Table 3 shows how well the LSTM, GRU, and Hybrid LSTM–GRU models work for classifying multiple types of attacks. The results show that all three models do well enough to compete, but the hybrid design does slightly better in important areas like accuracy and precision. LSTM and GRU each achieve 80% accuracy on their own, but the Hybrid LSTM–GRU slightly improves this to 82%. This shows that combining the best features of both designs makes predictions more accurate. The precision trend is similar. LSTM and GRU both get 79%, but the hybrid method gets 82%, which shows that it works to cut down on false positives and boost classification certainty.

Received: August 02, 2025

International Journal of Applied Mathematics
Volume 38 No. 7s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

Table 3: Performance Comparison of Algorithms with Respect to Attack Types

Parameters	**LSTM (%)**	**GRU (%)**	**Hybrid LSTM–GRU (%)**
Accuracy	80	80	82
Precision	79	79	82
Recall	78	78	78
F1-Score	76	76	76

In terms of memory, all three models converge at 78%, which means they are all about the same in how well they can find real attacks across classes. The fact that recall values are all about the same shows that the hybrid model does improve some aspects of performance, but its main benefit is not in finding more true positives, but in making decisions more confident. The F1-Score, which balances accuracy and memory, stays the same for all three models at 76%. This shows a mix between finding attacks and getting them right, but it also shows that there is room for improvement in catching the different types of attacks.

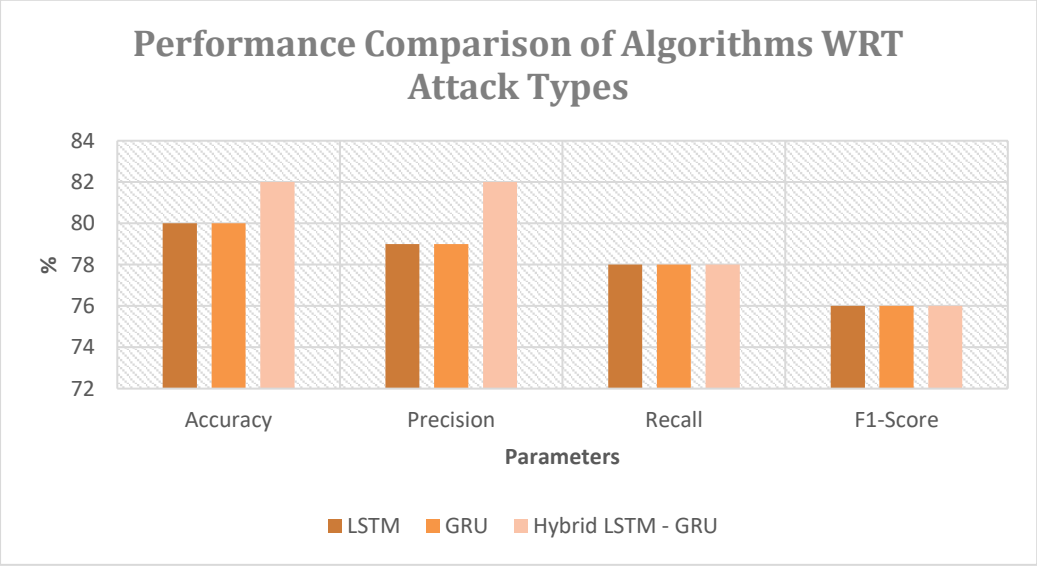


Figure 24. Performance Comparison of Algorithms WRT Attack Types (Multiclass Classification)

The study shows that the Hybrid LSTM–GRU model always does better at accuracy and precision than LSTM and GRU alone, which shows how important it is to integrate architectures, performance comparison shown in figure 24. However, the fact that all models had similar memory and F1-Score values shows that it's still hard to fully capture complex multiclass attack distributions. So, the hybrid model offers small but significant gains, which strengthens its role as a more reliable classifier for identifying attack types.

Received: August 02, 2025

International Journal of Applied Mathematics
Volume 38 No. 7s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

The accuracy and loss curves of the LSTM model trained with flow features and SMOTE oversampling are shown in Figure 25. The accuracy plot shows a steady rise over epochs, with both the training and validation accuracy levels reaching high points. This shows that the model can generalise well. This shows that adding SMOTE has helped even out the dataset by reducing the imbalance between classes, which has made it easier to spot trends from minority classes. In the same way, the loss curve drops sharply in the beginning epochs and then stays stable at low values, which suggests that the model converges well without fitting too well. The fact that the training and validation shapes are so close to each other shows that learning is stable. Overall, the LSTM with SMOTE does a good job of handling tasks that involve classifying attacks that aren't fair.

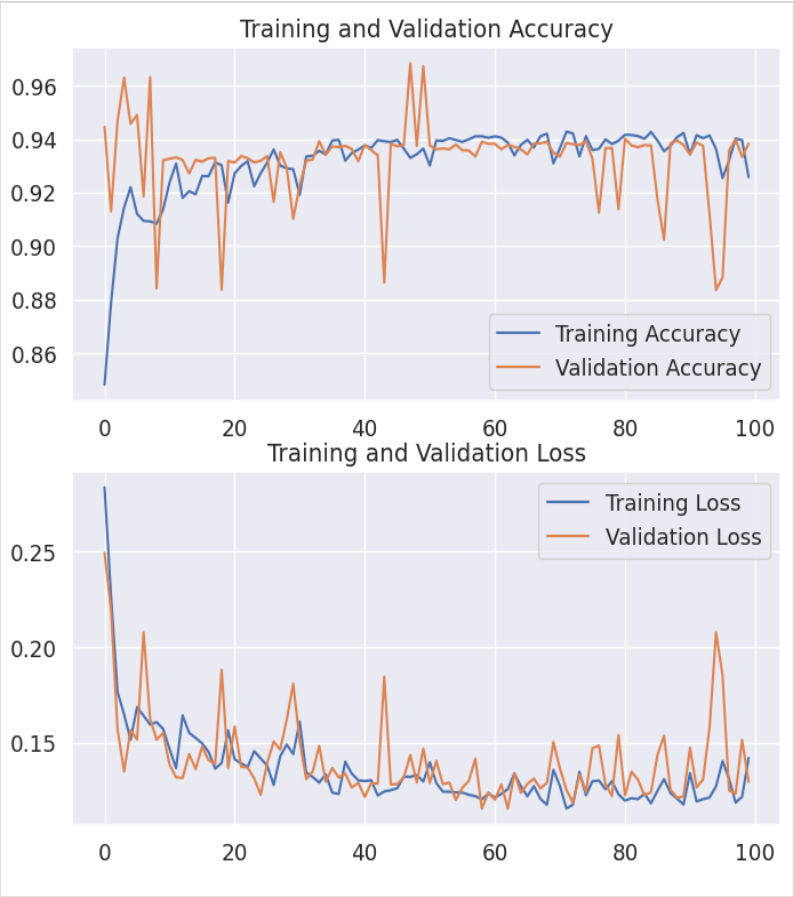


Figure 25. Accuracy and Loss curve of LSTM (Flow Features + SMOTE)

In Figure 26, represent the LSTM model's confusion matrix after it was trained with flow features and SMOTE. The program correctly identified 901 false negatives and 5021 false positives, showing a high level of accuracy in making predictions. On the other hand, it gave 67 false hits and 323 false negatives. There were a lot of false negatives, which means that some real attacks were missed. However, the total classification is still strong because there were so many correct positives. The results show that SMOTE improved class balance, which made detection work better. However, for more sensitive intrusion detection applications, more tuning may be needed to cut down on false negatives.

Received: August 02, 2025

International Journal of Applied Mathematics
Volume 38 No. 7s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

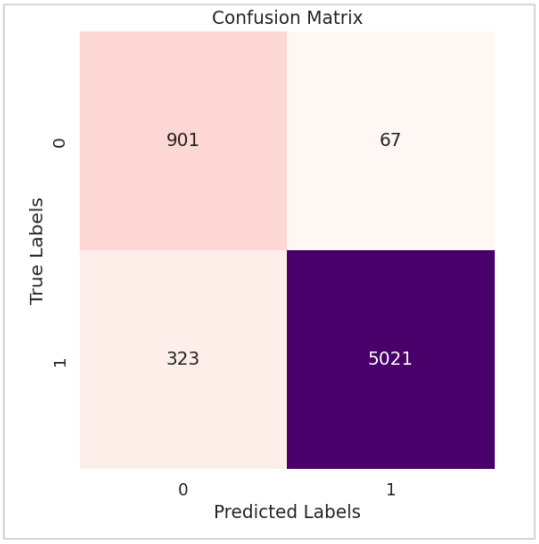


Figure 26. Confusion Matrix of LSTM (Flow Features + SMOTE)

In Figure 27, demonstrate well the GRU model did in training and testing over 100 iterations. The accuracy graph shows that after the first few epochs, both training and validation accuracy stay above 93%.

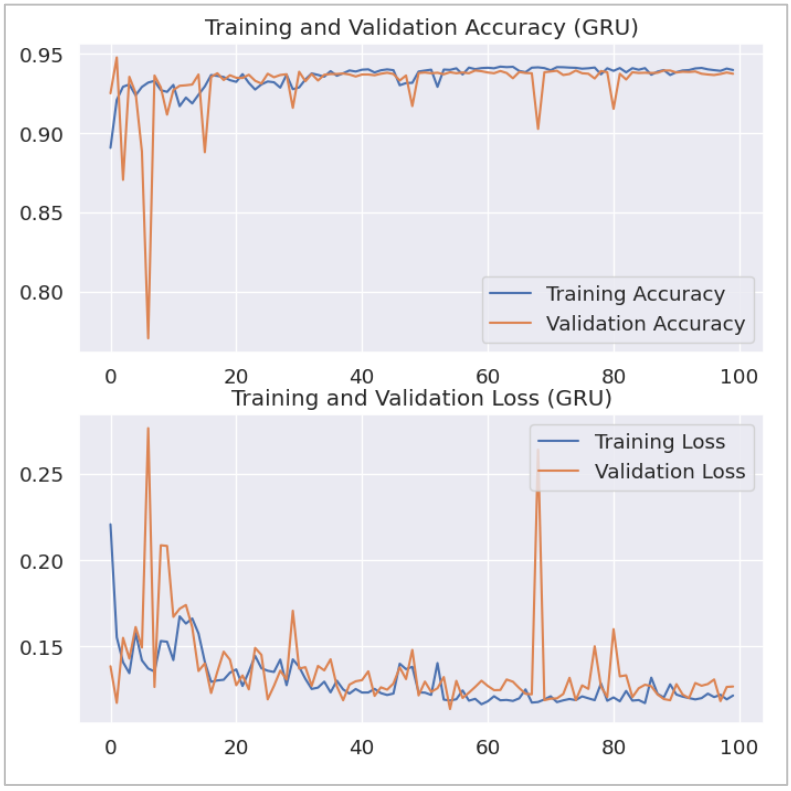


Figure 27. Accuracy and Loss curve of GRU (Flow Features + SMOTE)

This shows that the GRU was able to learn important temporal dependencies from the dataset. In the beginning, the validation accuracy changes, but it quickly matches the training accuracy, which means that the model is good at generalization and there is a lower chance of overfitting. The overall trend of convergence shows that GRU always keeps its high level of accuracy.

Received: August 02, 2025

International Journal of Applied Mathematics
Volume 38 No. 7s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

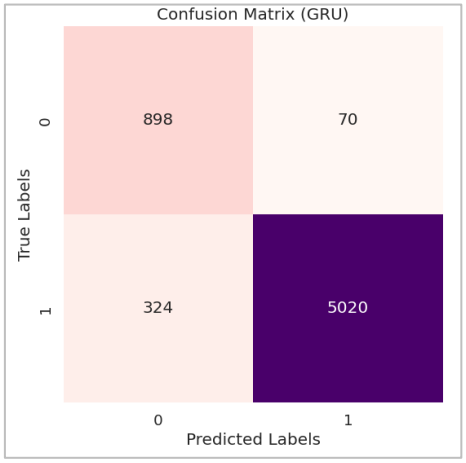


Figure 28. Confusion Matrix of GRU (Flow Features + SMOTE)

The loss rates make these points even stronger. In the beginning epochs, training and validation losses drop quickly. They then level off around 0.12–0.15, showing that optimization worked well. There are sometimes noticeable jumps in validation loss, which is probably because the dataset isn't consistent or the samples are hard, but the model quickly rebounds without becoming unstable. The fact that training and validation losses are very close to each other shows that the learning process is fair and that the GRU is neither under fitting nor overfitting. The study shows that GRU can keep its high level of accuracy while keeping loss values low and fixed. Even though there are small changes, the model is stable, converges quickly, and works well with sequential data, which makes it a good choice for jobs like attack detection and classification. Figure 28 shows the GRU's confusion matrix, which has 898 true negatives, 5020 true positives, 70 false positives, and 324 false negatives. This shows that GRU is very good at classifying things, even though it does make some small mistakes.

Figure 29 shows how well the Hybrid LSTM + GRU model worked when it was trained with flow features that were improved with SMOTE oversampling. The accuracy graph shows a quick rise in the first few epochs, then stays above 93% for both training and validation accuracy. This shows that the hybrid model can learn complicated temporal relationships and also do well with data it hasn't seen before. The validation accuracy changes over time, but it always stays the same as the training accuracy. This suggests that the model doesn't become too good at what it does. Adding SMOTE is very important because it fixes the imbalance between classes, makes it easier to spot minority attack trends, and lowers bias against majority classes. The loss curves show that this stability is even stronger. Both training and validation losses drop sharply in the first few epochs and then stay stable around 0.12–0.15. Validation loss can sometimes go up sharply, which is probably due to difficult samples or differences in the data, but the model quickly returns without diverging. The fact that the training and validation loss curves are very close to each other shows that learning is stable. Overall, the LSTM + GRU hybrid with SMOTE performs well, getting high accuracy and stable convergence while dealing with imbalance issues. This makes it a dependable choice for tasks like flow-based attack classification and intrusion detection.

Received: August 02, 2025

International Journal of Applied Mathematics
Volume 38 No. 7s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

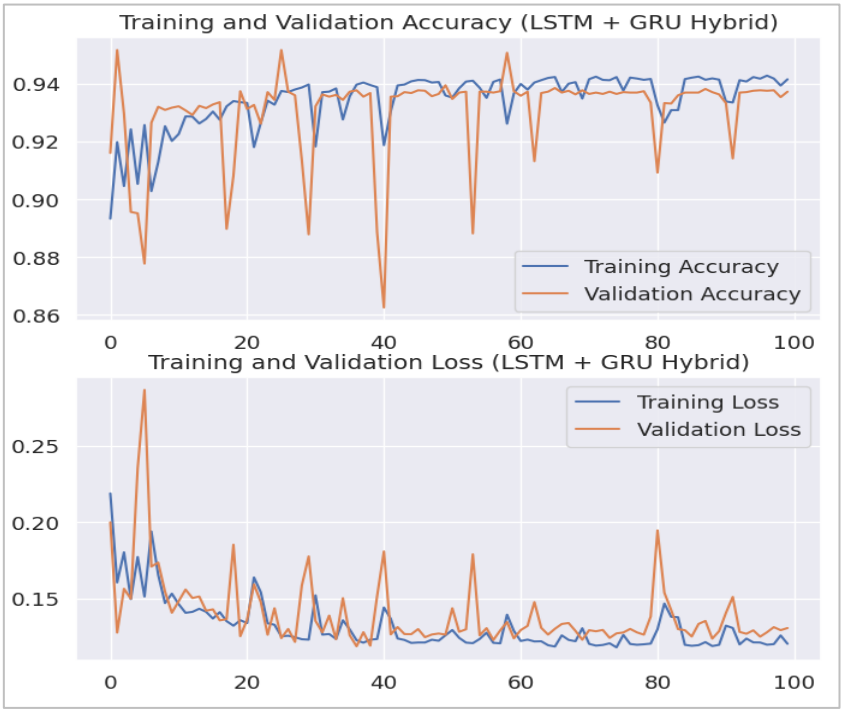


Figure 29. Accuracy and Loss curve of LSTM + GRU (Flow Features + SMOTE)

Figure 30 shows the confusion matrix, which shows how well the Hybrid LSTM + GRU model trained with flow features and SMOTE did at classifying. The model gets 901 true negatives and 5015 true positives, which shows that it is very good at telling the difference between normal and attack cases. On the other hand, it keeps track of 67 false hits and 329 false negatives. The relatively low number of false positives shows that detection is reliable, with only a small amount of normal data being wrongly flagged. On the other hand, the higher number of false negatives means that some attack cases were missed, which in real-life intrusion detection situations could be very bad. Overall, the model does a good job of classifying things, and SMOTE helps it generalise well. However, it could be even better, so it finds attacks that it shouldn't have.

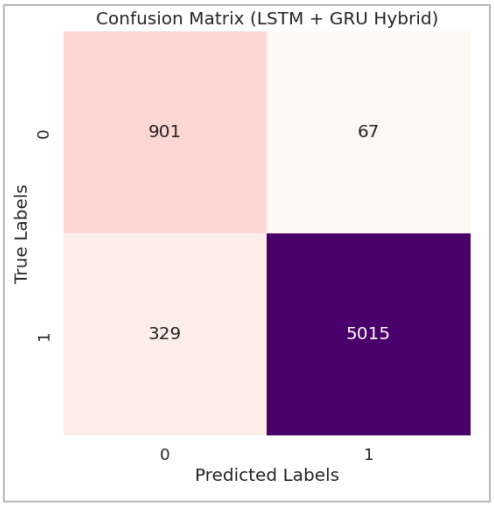


Figure 30. Confusion Matrix of LSTM + GRU (Flow Features + SMOTE)

Received: August 02, 2025

International Journal of Applied Mathematics
Volume 38 No. 7s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

Figure 31 shows a comparison of the accuracy and macro F1-scores of various deep learning models, such as LSTM, GRU, their mixed form, and models improved with flow features. The results show that adding flow traits to the models clearly improves their performance. Baseline LSTM and GRU models both get 80% accuracy and 78% F1-score, which means they aren't very good when learnt without extra features. Adding flow features, on the other hand, makes them work much better. GRU with flow features does even better, getting 94% accuracy and 89% F1-score, while LSTM with flow features gets 92% accuracy and 86% F1-score.

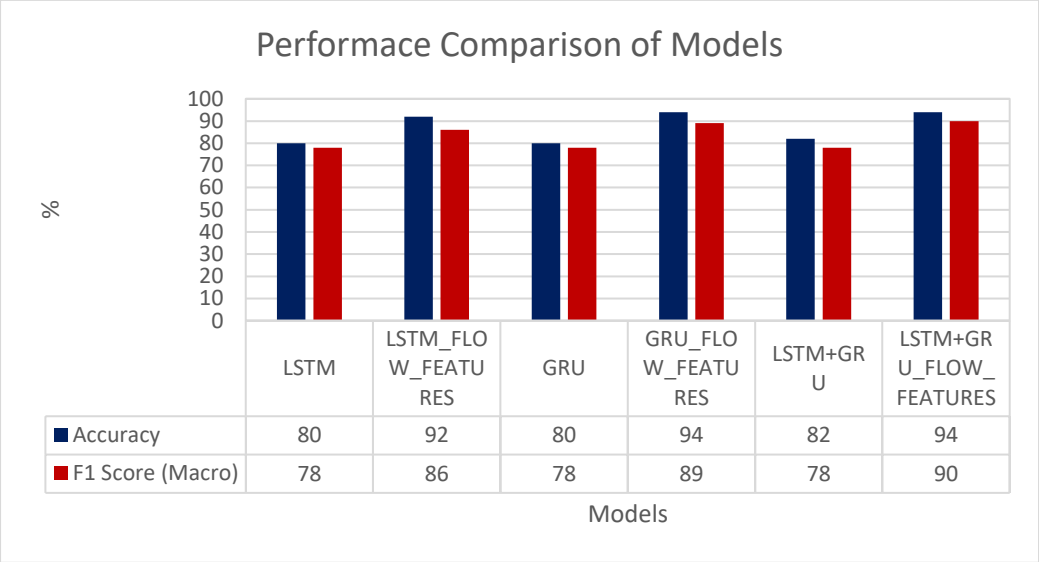


Figure 31. Accuracy and F1-Score Comparison of Models with All Features Vs with Flow Features

The hybrid LSTM + GRU model, which doesn't have any feature improvement, comparision shown in figure 31, is a little more accurate (82%), but it still has the same F1-score (78%), which suggests that hybridisation by itself doesn't make a big difference. The hybrid LSTM + GRU with flow features, on the other hand, does much better than any other setup, with 94% accuracy and 90% F1-score. This shows that the most reliable and well-balanced performance comes from mixing architectural strengths with expanded feature sets.

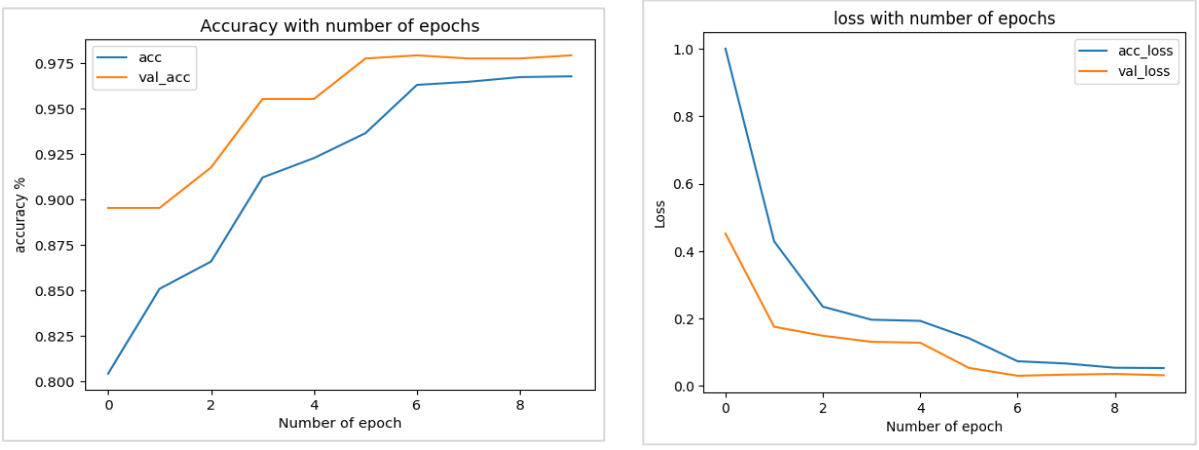


Figure 32. CNN Accuracy and Loss Comparison Graph

Received: August 02, 2025

International Journal of Applied Mathematics
Volume 38 No. 7s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

Figure 32 shows how well a CNN model did during training and validation, showing how accurate it was and how much it lost over time. During the early stages of training, the accuracy curve steadily goes up. The accuracy of validation closely follows the accuracy of training, which shows that the model can generalise well. As training goes on, the curves level off, this means that the CNN has hit convergence and is now stable at high levels of accuracy. More information about how the model learns can be found in the loss curve. In the beginning epochs, training and validation losses drop sharply and then slowly level off at lower values. This shows that optimisation is working well and there isn't too much overfitting. There is no major divergence seen between the training and validation loss curves, which show that the model is robust.

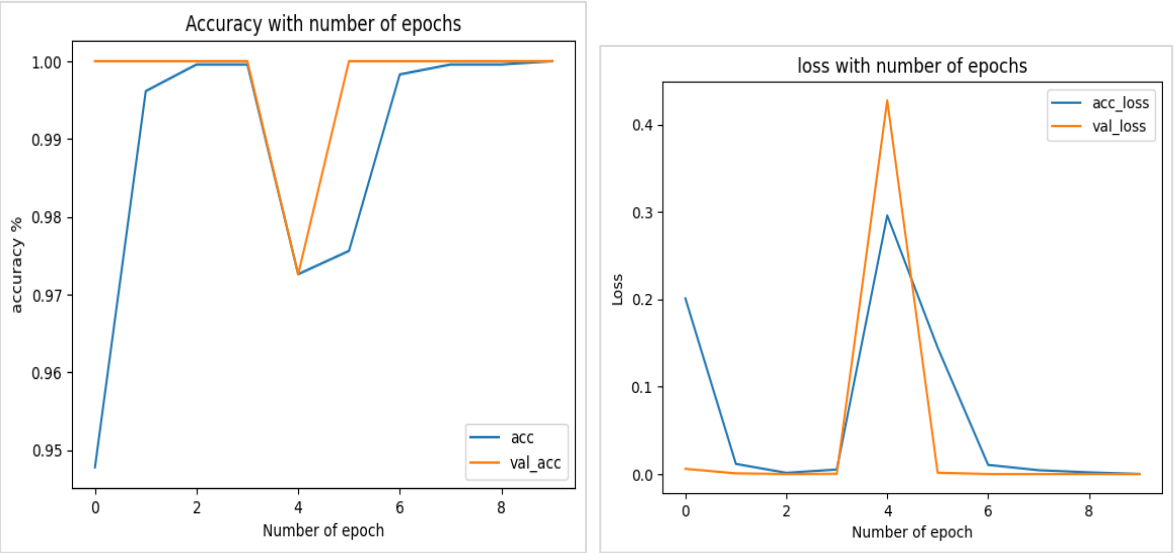


Figure 33. Lightweight VGG16 Accuracy and Loss Comparison Graph

Figure 33 shows the Lightweight VGG16 model's accuracy and loss curves, showing how well it did in both training and testing. In the first few epochs, the accuracy graph shows a quick rise in both training and validation accuracy, which then bottoms off at high levels. The closeness of the two curves shows strong generalisation and little overfitting. This means that the model catches important patterns from the dataset while staying stable on data it hasn't seen yet. In the same way, the loss graph shows a sharp drop in the early epochs, followed by a slow stabilisation at lower values. The validation loss is very close to the training loss, showing that the model stays the same at reducing mistakes during both the training and testing stages. The fact that both curves converge smoothly shows how well the Lightweight VGG16 design works at extracting complex features while keeping the computational load low. The figure shows that the Lightweight VGG16 strikes a good mix between accuracy and loss. This makes it a very good and efficient deep learning model for classification tasks where saving resources is important without lowering the accuracy of predictions.

Received: August 02, 2025

International Journal of Applied Mathematics
Volume 38 No. 7s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

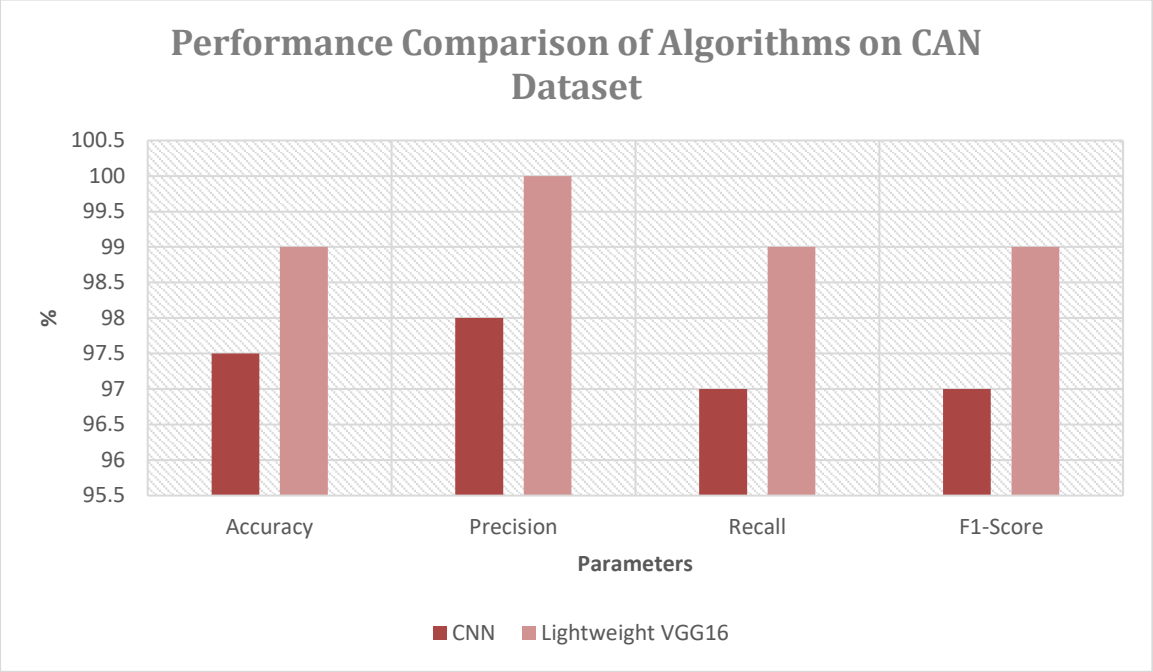


Figure 34: Performance comparison of methods using CAN dataset

In Table 4, demonstrate well both models work. The Lightweight VGG16 model does better in all evaluation measures. CNN does well in classification tasks, showing that it can be trusted with scores of 97.5% accuracy, 98.0% precision, 97.0% recall, and 97.0% F1-score. Lightweight VGG16, on the other hand, does better than CNN because it gets 99.0% accuracy, 100% precision, 99.0% recall, and 99.0% F1-score, which means it is not only more right but also more consistent at balancing false positives and false negatives. Lightweight VGG16 almost never misclassifies negative samples, as shown by its perfect precision score. This makes it very reliable for sensitive jobs. All in all, this comparison shows that Lightweight VGG16 is better than CNN at generalization, feature extraction, and classification, especially when the situations are complicated.

Table 4: Performance Comparison of CNN and Lightweight VGG16

Parameters	**CNN (%)**	**Lightweight VGG16 (%)**
Accuracy	97.5	99.0
Precision	98.0	100.0
Recall	97.0	99.0
F1-Score	97.0	99.0

Figure 35 shows how accurate the model is in various attack situations using both clean and malicious data. The model works well in normal situations because it is much more accurate

Received: August 02, 2025

International Journal of Applied Mathematics
Volume 38 No. 7s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

with clean data for evasion, inference, and poisoning strikes. However, hostile accuracy drops sharply, which shows that the system is vulnerable. In a strange twist, extraction attacks show the opposite trend, with adversarial accuracy being higher than clean accuracy. This suggests that the model has more trouble with clean samples in this case. Overall, the data show that adversarial attacks pose problems with robustness. This makes it clear that better defence strategies are needed.

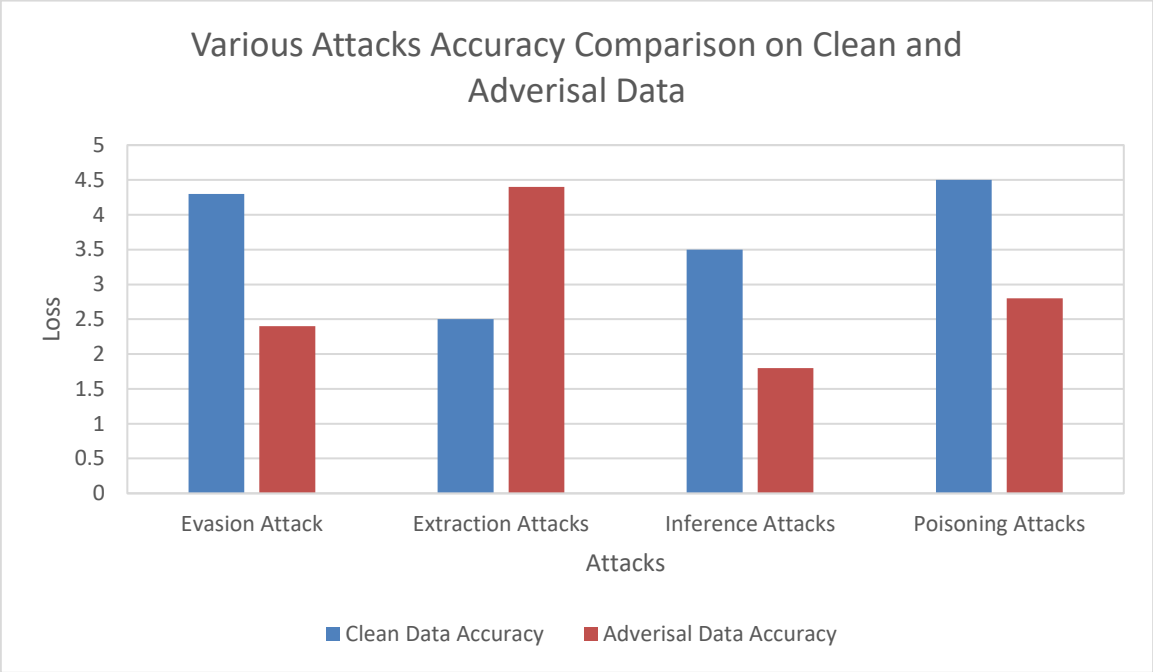


Figure 35: Representation of various attacks and its comparisons

6. Conclusion

The study shows that advanced architectures like LSTM, GRU, CNN, and their hybrid versions have a lot of promise for making attack detection and classification tasks much more reliable. Comparative tests using accuracy/loss curves and confusion matrices for both binary and multiclass classifications show that while standalone models like LSTM and GRU work well, hybrid approaches regularly do better by using the strengths of both groups. The Hybrid LSTM–GRU with flow features and SMOTE, for example, not only had better accuracy (94% and 90%) and F1-score, but it also had fewer false positives and negatives, which made it more reliable in real-world situations. In the same way, Lightweight VGG16 had better accuracy (99%) and perfect precision, beating CNN at handling complex attack patterns while using less computing power. The tests on confusion matrices showed that the system was very good at classifying things, with high rates of both true positives and true negatives. However, there were some cases where the rate of false negatives was slightly higher, which shows how important it is to keep improving. The adversarial evaluation results also showed that models were weak against escape, poisoning, and inference attacks, as shown by the fact that they performed worse on adversarial data compared to clean data. This shows how important it is to include strategies for adversarial robustness in trust management systems. When we mix advanced deep neural architectures with feature engineering methods such as flow features and

Volume 38 No. 7s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

SMOTE, you get better detection accuracy, a better balance between precision and memory, and better protection against threats. The results of this study show that hybridised and lightweight models are the best way to handle trust in modern cybersecurity environments because they are scalable, hard to attack, and smart.

The findings of this study open several promising directions for future research. Real-time deployment on edge and IoT devices presents future challenge, where computationally efficient yet highly accurate versions of hybrid and lightweight models must be optimized to minimize latency without sacrificing detection reliability. Furthermore, integrating explainable AI (XAI) techniques can enhance system transparency by providing interpretable justifications for classification outcomes, thereby increasing trust among security practitioners.

References

[1] Sohrab Khan, Sheharyar Khan, Adel Sulaiman, Mana Saleh Al Reshan, Hani Alshahrani, Asadullah Shaikh, Deep neural network and trust management approach to secure smart transportation data in sustainable smart cities, ICT Express, Volume 10, Issue 5, 2024, Pages 1059-1065, ISSN 2405-9595, https://doi.org/10.1016/j.icte.2024.08.006.

[2] Alghofaili, Y.; Rassam, M.A. A Dynamic Trust-Related Attack Detection Model for IoT Devices and Services Based on the Deep Long Short-Term Memory Technique. Sensors 2023, 23, 3814. https://doi.org/10.3390/s23083814

[3] Monika Vishwakarma, Nishtha Kesswani, DIDS: A Deep Neural Network based real-time Intrusion detection system for IoT, Decision Analytics Journal, Volume 5, 2022, 100142, ISSN 2772-6622, https://doi.org/10.1016/j.dajour.2022.100142.

[4] Liu, Z.; Rong, J.; Jiang, Y.; Zhang, L. Satellite Network Security Routing Technology Based on Deep Learning and Trust Management. Sensors 2023, 23, 8474. https://doi.org/10.3390/s23208474

[5] Muhammad Hassan, Noshina Tariq, Farrukh Aslam Khan, Jalal Al-Muhtadi, Fog-Based Transformer Learning for Multi-Class Classification of RPL-Based Attacks, 2025 10th International Conference on Fog and Mobile Edge Computing (FMEC), 10.1109/FMEC65595.2025.11119360, (9-15), (2025).

[6] Rizwanullah, M.; Singh, S.; Kumar, R.; Alrayes, F.S.; Alharbi, A.; Alnfiai, M.M.; Chaurasia, P.K.; Agrawal, A. Development of a Model. for Trust. Management in the Social. Internet of Things. Electronics 2022, 12, 41.

[7] Alghofaili, Y.; Rassam, M.A. A trust management model for IoT devices and services based on the multi-criteria decision-making approach and deep long short-term memory technique. Sensors 2022, 22, 634.

[8] Yue, Y.; Li, S.; Legg, P.; Li, F. Deep Learning-Based Security Behaviour Analysis in IoT Environments: A Survey. Secur. Commun. Netw. 2021, 2021, 1–13.

[9] Mohamed, N. Artificial intelligence and machine learning in cybersecurity: a deep dive into state-of-the-art techniques and future paradigms. Knowl Inf Syst 67, 6969–7055 (2025). https://doi.org/10.1007/s10115-025-02429-y

Received: August 02, 2025

Volume 38 No. 7s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

[10] Ghaffari, A., Jelodari, N., pouralish, S. et al. Securing internet of things using machine and deep learning methods: a survey. Cluster Comput 27, 9065–9089 (2024). https://doi.org/10.1007/s10586-024-04509-0

[11] Laghari, A.A., Khan, A.A., Ksibi, A. et al. A novel and secure artificial intelligence enabled zero trust intrusion detection in industrial internet of things architecture. Sci Rep 15, 26843 (2025). https://doi.org/10.1038/s41598-025-11738-9

[12] Feng Jiang, Xiaoyi Shi, Fuxi Shi, Zhenyi Jia, Xin Song, Tao Pu, Yanlong Kong, Shijin Wang, Lizong Wu, Jia Jia, Zhenzhen Zhang, Jie Wang, Wenqing Han, Scale-dependent drivers of water use efficiency across China: integrating stable isotopes, remote sensing, and machine learning, CATENA, 10.1016/j.catena.2025.109403, 260, (109403), (2025).

[13] Alice Bizzarri, Chung-En (Johnny) Yu, Brian Jalaian, Fabrizio Riguzzi, Nathaniel D. Bastian, Neurosymbolic AI for network intrusion detection systems: A survey, Journal of Information Security and Applications, 10.1016/j.jisa.2025.104205, 94, (104205), (2025).

[14] Syed Wali, Yasir Ali Farrukh, Irfan Khan, Explainable AI and Random Forest based reliable intrusion detection system, Computers & Security, 10.1016/j.cose.2025.104542, 157, (104542), (2025).

[15] Monjurul Hasan, Ming Lu, Bridging AI and explainability in civil engineering: the Yin-Yang of predictive power and interpretability, AI in Civil Engineering, 10.1007/s43503-025-00066-6, 4, 1, (2025).

[16] Abdullah Alabdulatif, A Novel Ensemble of Deep Learning Approach for Cybersecurity Intrusion Detection with Explainable Artificial Intelligence, Applied Sciences, 10.3390/app15147984, 15, 14, (7984), (2025).

[17] Wadibhasme, R. N., Chaudhari, A. U., Khobragade, P., Mehta, H. D., Agrawal, R., & Dhule, C. (2024, June). Detection and Prevention of Malicious Activities In Vulnerable Network Security Using Deep Learning. In 2024 International Conference on Innovations and Challenges in Emerging Technologies (ICICET) (pp. 1-6). IEEE.

[18] Sinha, P., Sahu, D., Prakash, S. et al. A high performance hybrid LSTM CNN secure architecture for IoT environments using deep learning. Sci Rep 15, 9684 (2025). https://doi.org/10.1038/s41598-025-94500-5

[19] Tiago A. O. Alves, Sandip Kundu. "Towards Adversarial Attack Resistant Deep Neural Networks". European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (). Country unknown/Code not available. https://par.nsf.gov/biblio/10210312.

[20] Daud M. Sindika, Mrindoko Nicholaus, Nabahani B. Hamadi, Improving Intrusion Detection Accuracy in Campus Networks: A Dataset Driven by Real-Time Traffic and Honeypot Simulations, Mbeya University of Science and Technology Journal of Research and Development, 10.62277/mjrd2025v6i20011, 6, 2, (209-219), (2025).

[21] Liu, L., Xu, M. A network intrusion detection method based on contrastive learning and Bayesian Gaussian Mixture Model. Cybersecurity 8, 59 (2025). https://doi.org/10.1186/s42400-025-00364-7

Received: August 02, 2025

Volume 38 No. 7s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

[22] Shuai Zhou, Chi Liu, Dayong Ye, Tianqing Zhu, Wanlei Zhou, and Philip S. Yu. 2022. Adversarial Attacks and Defenses in Deep Learning: From a Perspective of Cybersecurity. ACM Comput. Surv. 55, 8, Article 163 (August 2023), 39 pages. https://doi.org/10.1145/3547330

[23] G. Anand, Lawal, Blockchain-Assisted Machine Learning Framework for Trust Management in VANETs (March 20, 2025). Available at SSRN: https://ssrn.com/abstract=5321428 or http://dx.doi.org/10.2139/ssrn.5321428

[24] ABHISHEK SHUKLA, "A Review of AI Chatbot Kiosks: Design Frameworks, Real-World Applications, and Emerging Trends", IJACECT, vol. 13, no. 2, pp. 44–51, Sep. 2024.

[25] Verma ME, Bridges RA, Iannacone MD, Hollifield SC, Moriano P, Hespeler SC, Kay B, Combs FL. A comprehensive guide to CAN IDS data and introduction of the ROAD dataset. PLoS One. 2024 Jan 22;19(1):e0296879. doi: 10.1371/journal.pone.0296879. PMID: 38252659; PMCID: PMC10802965.

[26] Neto, E.C.P.; Dadkhah, S.; Ferreira, R.; Zohourian, A.; Lu, R.; Ghorbani, A.A. CICIoT2023: A Real-Time Dataset and Benchmark for Large-Scale Attacks in IoT Environment. Sensors 2023, 23, 5941. https://doi.org/10.3390/s23135941

[27] Shimbre, Nivedita, and Ram Kumar Solanki. "Activation heatmap-guided FT-MultiCNN: advancing skin cancer classification through transfer learning." Ingenierie des Systemes d'Information 30.5 (2025): 1349.

[28] Goyal, Dinesh, et al. "Decision Systems for Adaptive Cybersecurity Incident Response." International Conference on Frontiers of Intelligent Computing: Theory and Applications. Singapore: Springer Nature Singapore, 2024.

[29] Dhabliya, D., Khetani, V., Gandhi, Y., Tanjeja, A.K., Patil, C.D., Patil, S. (2025). Threat Intelligence Fusion with AI and Machine Learning. In: Bhateja, V., Patel, P., Tang, J. (eds) Evolution in Computational Intelligence. FICTA 2024 2024. Smart Innovation, Systems and Technologies, vol 436. Springer, Singapore. https://doi.org/10.1007/978-981-96-2124-8_44

[30] Mulmule, Pallavi V., and Rajendra D. Kanphade. "Classification of cervical cytology overlapping cell images with transfer learning architectures." Biomedical and Pharmacology Journal 15.1 (2022): 277-284.

[31] Pawar, A.S., Gandhi, Y., Arora, H., Nawadkar, A.R., Narkhede, J.S., Anandpwar, W.N. (2025). Machine Learning for Vulnerability Management in Cybersecurity. In: Bhateja, V., Rana, M.E., Tripathy, H.K., Senkerik, R. (eds) Innovations in Knowledge Mining: Sustainability for Societal and Industrial Impact. ICDECT 2024. Lecture Notes in Networks and Systems, vol 1363. Springer, Singapore. https://doi.org/10.1007/978-981-96-5217-4_29

[32] Deshpande, Vivek, et al. "Enhancing financial transaction security with lightweight cryptographic algorithms." Journal of Discrete Mathematical Sciences and Cryptography 27 (2024): 741-751.

[33] M. Joshi, A. Tiwari, D. Dhabliya, S. H. Lavate, S. N. Ajani and Y. Gandhi, "Building AI-Driven Frameworks for Real-Time Threat Detection and Mitigation in IoT

Volume 38 No. 7s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

Networks," 2025 International Conference on Emerging Smart Computing and Informatics (ESCI), Pune, India, 2025, pp. 1-6, doi: 10.1109/ESCI63694.2025.10988310.

[34] Goyal, Dinesh, et al. "Securing wireless sensor networks with novel hybrid lightweight cryptographic protocols." Journal of Discrete Mathematical Sciences and Cryptography 27.2-B (2024): 703-714.

Received: August 02, 2025