

GATv2-Based Artificial Intelligence Framework for Intrusion Detection in UAVCAN-Enabled Unmanned Aerial Vehicle Networks

Hasan Alkahtani^{1,*}

¹ College of Computer Science and Information Technology, King Faisal University, P.O. Box 400, Al-Ahsa 31982, Saudi Arabia.

* Corresponding author: hsalkahtani@kfu.edu.sa

Received: 12 -8-2025 Accepted: 5-9- 2026 Published: 06 October 2026

DOI: 10.15849/ijasca.199

License: Open access under CC BY 4.0. Authors retain copyright.

Publisher: Al-Zaytoonah University of Jordan (ZUJ), Amman, Jordan.

Cite this article: H. Alkahtani, "GATv2-based artificial intelligence framework for intrusion detection in UAVCAN-enabled unmanned aerial vehicle networks," *Int. J. Advance Soft Compu. Appl.*, vol. 18, no. 3, pp. 196–221, 2026, doi: 10.15849/ijasca.199.

ABSTRACT

Network-level attacks targeting unmanned aerial vehicles (UAVs) that use the UAVCAN (DroneCAN) protocol are becoming increasingly attractive due to the Controller Area Network (CAN) bus's broadcast nature. We apply and empirically evaluate Graph Attention Network (GATv2) on the task of binary intrusion detection classification over the nine scenarios of the publicly available UAVCAN attack dataset, including Flooding (scenarios 1–2), Fuzzy (scenarios 3–5), Replay (scenario 6), and compound multi-vector attacks (scenarios 7–9). Under the high-density Flooding attack in Scenario 1, the model achieves a peak weighted F1-score of 99.80% and an accuracy of 99.66%, and its performance remains above 98% across five other scenarios. We observe performance degradation under long-term Fuzzy injection (Scenario 5: accuracy 90.57%) and covert Replay attacks (Scenario 6: accuracy 91.93%), which exhibit two distinct failure modes: false-positive-dominant under statistical boundary blur and false-negative-dominant under structurally secure traffic. The performance path across attack types and compositional complexity is quantified in a comparison table for each scenario. The findings establish GATv2 as a strong baseline for graph-based UAV IDS and highlight six focused recommendations to refine the model toward full-scenario robustness.

Keywords: UAV security; UAVCAN; DroneCAN; GATv; Graph Attention Network; intrusion detection; CAN bus security.

1 Introduction

Due to rapid technological advancements and significant shifts in public policy, autonomous Unmanned Aerial Vehicles have become among the most challenging and promising solutions in both the military and civilian sectors. Drones, also known as UAVs, can be used for many purposes, such as remote sensing [1], surveillance, package delivery, and precision farming [2,3]. UAV is a general term for flying vehicles, such as fixed-wing, multirotor, and Vertical Takeoff and Landing (VTOL) aircraft. These vehicles can be further divided into groups based on their range, size, payload capacity, and altitude [4].

UAVs have a natural edge over Unmanned Ground Vehicles (UGVs) because they can change altitude, giving them greater freedom [5]. People know that UAVs are cheap, fast, and easy to set up. VTOL has become more useful for stable hovering over the target since its initial development [6]. They differ from other things because they can fly at different speeds, work near other things, carry the sensors they need, and fly both inside and outside. These abilities let them take the place of people in dangerous, hard-to-reach, complicated, or expensive places.

UAVs that fly need to communicate with each other and have sensors and control loops that operate autonomously. These things make it easy for the enemy to get in, which is a shame. Over the past ten years, more people and the military have used drones (UAVs). Many kinds of businesses use these technologies, including those that care for the environment, run smart cities, farm more, and respond to disasters. The military needs better, safer ways to communicate as it uses more UAVs. You should know how robots talk to each other in the air these days. This set of rules enables the ship's sensors, motors, motion controls, and guidance systems to communicate quickly and in real time. This is very helpful for

many of them. People who use CAN connections can currently communicate with each other via UAVCAN. It is small, safe, and reliable. People like it for robots, electronics, and UAVs because it is robust, can handle errors, and can communicate with people over long distances. Setting up UAVCAN platforms that let many devices on the same LAN communicate with each other is simple. This makes it easy to add new devices and keep them up to date. UAVCAN networks have many benefits, but they can still be hacked. UAVs are more likely to be hacked because they connect to more outside networks, such as cloud platforms and ground control units. An enemy could take over important UAV functions, change the data that the UAV uses to sense things, launch a denial-of-service attack, or send bad data through a weak connection. Such a strike could damage the plane, business, or personal information [7,8,9,10,11,12,].

IDS are a simple way to handle hackers and protect the network. However, such measures will not always work against advanced attacks, e.g., using zero-day vulnerabilities or insider threats. With an IDS, it is easier to keep UAVs safe. It uses minimal network information and detects any anomalous attempts to access the network. IDSs can monitor network traffic for suspicious patterns or indications of an imminent attack. This helps network administrators identify malicious sites before they cause significant damage. Deep Learning (DL) is a type of machine learning (ML) well-suited to security jobs that require analyzing data and patterns. A lot of research has been done on DL as a possible solution to these issues. DL systems can easily assemble models of large groups of objects. In this way, they might be able to find odd patterns that point to a hack. There are many good ways to find attackers who want to hack into networks. This group includes neural networks, recurrent neural networks (RNNs), convolutional neural networks (CNNs), and graph neural networks (GNNs) [13,14,15,16,17].

The new models are better at finding problems, whether they are known or not, and at showing how different things are connected. DL approaches that detect attackers attempting to breach UAV networks can be helpful in many ways. With little help from others, they can learn to locate things, handle large volumes of data, and adjust their responses to emerging trends. It can continuously detect problems and monitor conditions by integrating DL models into existing UAV transmission systems. Devices don't have enough resources, or files don't have enough names. GATv2-Based Intrusion Detection Framework for UAVCAN Networks of UAVs. In this work, we propose and evaluate a Graph Attention Network version 2 (GATv2) based model to detect intrusion attacks transmitted over the UAVCAN protocol in UAV communication systems. We model inter-node bus traffic as a graph to capture relational anomalies invisible to conventional sequential classifiers. The main objective of this project is to apply a Graphical neural network to understand how the UAVCAN data networks could have been breached. The proposed system will be used to identify trends in UAVCAN traffic. This will help find attackers who may be trying to hack something. This system can detect anomalous behavior among different parts of the UAV that communicate with one another. The main contributions of this study can be summarized as follows:

- Development of a deep learning-based IDS model tailored for UAVCAN communication networks.
- Design of a Graphical neural network that can help to detect attacks in the UAVCAN system.
- Evaluation of the proposed IDS using relevant performance metrics such as accuracy, precision, recall, and F1-score.
- Demonstration of the effectiveness of GATv2 in improving the detection of cyber threats in UAV communication systems.

2 Background of the Study

Unmanned Aerial Vehicles (UAV) have evolved from special military platforms to general cyber physical systems. Nowadays, they are used for forestry observation and environmental mapping [1], precision agriculture [2,3], vision-based navigation [4], cooperative UAV-UGV missions [5], and vertical take-off and landing operations [6]. This expansion has increased the amount of sensing, computing, and communication performed in flight. Hence, a modern UAV is not simply an aircraft but a distributed network of flight controllers, sensors, actuators, navigation modules, payload devices and communication links, whose proper interaction is essential for safe operation. Communication is the foundation of autonomous and cooperative operation of UAVs. Mobility, limited bandwidth, changing topology, interference, latency, and energy constraints are some of the enduring design challenges in UAV communication networks [7]. Similar concerns arise in flying ad hoc networks, where nodes are aircraft that need to maintain links as they move quickly and network membership changes [9]. The growing deployment of UAVs for civilian applications also increases these challenges, as platforms need to communicate with ground-control stations, cloud services, satellite or cellular links, and other aircraft [8]. Each additional connection can increase mission capability but also increase the system's attack surface.

UAVCAN (also known in its widely deployed earlier form as DroneCAN) provides communication between embedded components on board the aircraft using the Controller Area Network (CAN) bus [10]. CAN was conceived for efficient and reliable message exchange in resource-constrained control systems. Its broadcast operation, arbitration mechanism, and compact frame structure are useful for real-time embedded communication. However, conventional CAN frames do not provide strong source authentication and confidentiality by default. Thus an attacker that is illegitimately accessing the bus can send high-rate frames, inject malformed payloads or replay previously valid traffic. It has been demonstrated in the broader automotive security literature that interconnected control networks can expose multiple experimental attack surfaces [11]. These observations are relevant to UAVCAN, since a successful intrusion can compromise safety-critical sensing and control rather than mere information confidentiality.

Classical security principles like authentication, integrity protection, access control, secure key management and defense in depth are still valid [12]. However, the introduction of comprehensive cryptographic protection to an existing UAV platform can be challenging due to legacy devices, computational constraints, interoperability requirements and real-time deadlines. So an intrusion-detection system (IDS) adds another layer of defense. An IDS does not assume that it can prevent all attacks. It monitors traffic or system behavior and looks for patterns that do not match legitimate operation. Traditionally, intrusion detection consists of misuse detection (matching known attack signatures) and anomaly detection (searching for deviations from learned normal behavior). The advantages and disadvantages of both approaches are discussed in a thorough manner in IDS reviews [13] and hybrid systems have been proposed to combine anomaly and misuse evidence [14]. Signature-based methods can be accurate for previously encountered attacks but can fail when the attack is new or deliberately modified. Anomaly-based systems can detect behavior that is not known, but can give false alarms when there is a change in the normal traffic. UAV intrusion detection inherits this trade-off but also imposes strict requirements for low latency, limited computational overhead, and reliable identification of rare but potentially dangerous malicious frames.

Machine-learning and deep-learning methods have become important because network traffic exhibits nonlinear, high-dimensional patterns that are difficult to characterize with manual rules. Recurrent neural networks have been applied to sequential intrusion-detection data [15], and previous deep-learning IDS frameworks have been shown to learn representations directly from network observations [16,17]. These studies showed the general benefit of representation learning for intrusion detection.

However, their results are not directly applicable to UAVCAN traffic due to differences in protocol, frame structure, operation timing, class imbalance, and error impact compared with traditional enterprise or Internet traffic.

In recent security studies of UAVs, various artificial intelligence methods are used. Yang et al. [18] surveyed AI-enabled safety and security for UAVs, from machine-learning models to emerging large-language-model applications. The work shows how standard artificial intelligence can help with threat detection, assisting decision-making and responding autonomously, but also highlights that robustness and trustworthiness remain concerns to address. Such a broad review is important for differentiating it from a specific experimental IDS. Research on DDoS and botnet detection in Internet of Things environments also provides insights for UAV security. Mahdi et al. [19] proposed real-time detection of DDoS attacks in IoT networks using machine learning. Chukwukelu et al. [20] have compared machine learning techniques for DDoS intrusion detection. Adewuyi et al. showed the overall convergence of data analytics, IoT and cybersecurity in smart ecosystems [21]. These studies provide useful methodological references on traffic classification, class imbalance, and data-driven monitoring. However, they should not be considered direct UAVCAN experiments, as their titles and scopes of work relate to IoT or smart ecosystems rather than the specific UAVCAN bus used in this work. Research on reinforcement learning for missile evasion and penetration guidance [22,23] shows the increasing importance of AI in autonomous aerial decision-making. Their relevance to the current study is indirect: they concern decisions about guidance and control, not intrusion detection in a network. They still reflect the value of cyber integrity. For an autonomous controller to make safe decisions, it must be able to trust the sensory and communication data it receives. The references are considered as adjacent autonomy research for convenience and not used as performance baselines for IDS performance comparison.

Work more closely related has studied intrusion detection in UAV communication environments. Vuong et al. [24] proposed an autoencoder-based feature extraction method followed by machine-learning classification for UAV communication intrusion detection. This approach uses representation learning to extract discriminative features from complex traffic observations prior to classification. The need to learn from distributed participants without collecting all raw traffic at a central location is reflected in the work of Ceviz et al., who studied a federated learning IDS to enhance UAV privacy and security [25]. Aldossary et al. [26] proposed a cross-layer convolutional attention method for intrusion detection in drone-to-drone and drone-to-base-station. The results showed an accuracy of 98.4%, a recall of 98.7% and an F1-score of 98.1%, which demonstrated the usefulness of attention mechanisms when information from multiple communication layers is available.

Traffic classifiers are complemented by architecture-level defenses. Guerber et al. [27] proposed a software-defined network architecture to enhance security for drone swarms. By separating control logic from forwarding behavior, SDN can enable centralized policy enforcement and dynamic network management, but it also introduces a critical dependency on the controller. Altawy and Youssef [28] surveyed the security, privacy and safety threats affecting civilian drones, showing that UAV protection must be considered together with physical compromise, wireless attacks, malicious control, privacy exposure and software vulnerabilities. Sharma et al. [29] studied ultra-dense cloud-drone networks. The flexibility of cloud-supported operations poses additional challenges for interference, monitoring, autonomy, and reliable network coordination. These works together illustrate that UAV security is a multi-layer problem, but none of these architectural contributions deal with malicious-frame detection on the internal UAVCAN bus.

The UAVCAN Attack Dataset provides a focused basis for studying attacks in a real UAVCAN/DroneCAN communication setting [30]. It includes normal traffic and controlled Flooding, Fuzzy, Replay and compound attack scenarios. Flooding attacks send excessive amounts of data to take control of bus access, Fuzzy attacks introduce random or malformed data, and Replay attacks retransmit

previously valid frames. Different types of attack also differ in visibility. Flooding can produce a strong traffic-volume signature, while Replay may preserve valid frame structure and thus be difficult to detect without temporal context. The different scenarios make the dataset useful to test whether a model performs consistently under attack conditions, rather than reporting a single aggregate score. Hong and Yoo proposed feature selection, heterogeneous ensemble, and SHAP explanations for multiple intrusion detection on the UAVCAN dataset [31]. Their framework combined prescriptive modeling with feature-level interpretation. They reported about 97% accuracy and F1-score. This study is important because it shows that an ensemble can obtain strong detection and explain which features influence the prediction. However, the relationships between individual CAN frames are not explicitly modeled by ensemble feature importance. Majumder et al. [32] investigated the graph-powered UAVCAN defense based on GCN, GAT, GraphSAGE and graph-transformer models and an LSTM baseline. Their scenario-level analysis shows that graph representation is a promising direction for UAVCAN intrusion detection. It also shows that in highly imbalanced scenarios, accuracy can be misleading on its own: some settings with high accuracy yield significantly lower F1-scores.

3 Methodology

Graph Attention Network (GATv2) extends the original GAT by replacing its static, content-agnostic attention with dynamic, jointly computed coefficients that depend on both source and target node features. This property is particularly valuable for UAVCAN graphs, where the relevance of a neighboring frame to the current node's classification depends on the combined feature context of both endpoints. The present study provides a scenario-by-scenario evaluation of GATv2 across nine reported scenarios from the UAVCAN attack dataset, including classification metrics, confusion matrix analyses, and a cross-scenario comparison. The proposed system describes the technical approach to an IDS for analyzing the UAVCAN Attack, as shown in Figure 1. The proposed system employs the GATv2 architecture for effective classification of network traffic. The methodology consists of four main stages: Data Preprocessing, Graph Construction, Model Architecture, and Training and Evaluation.

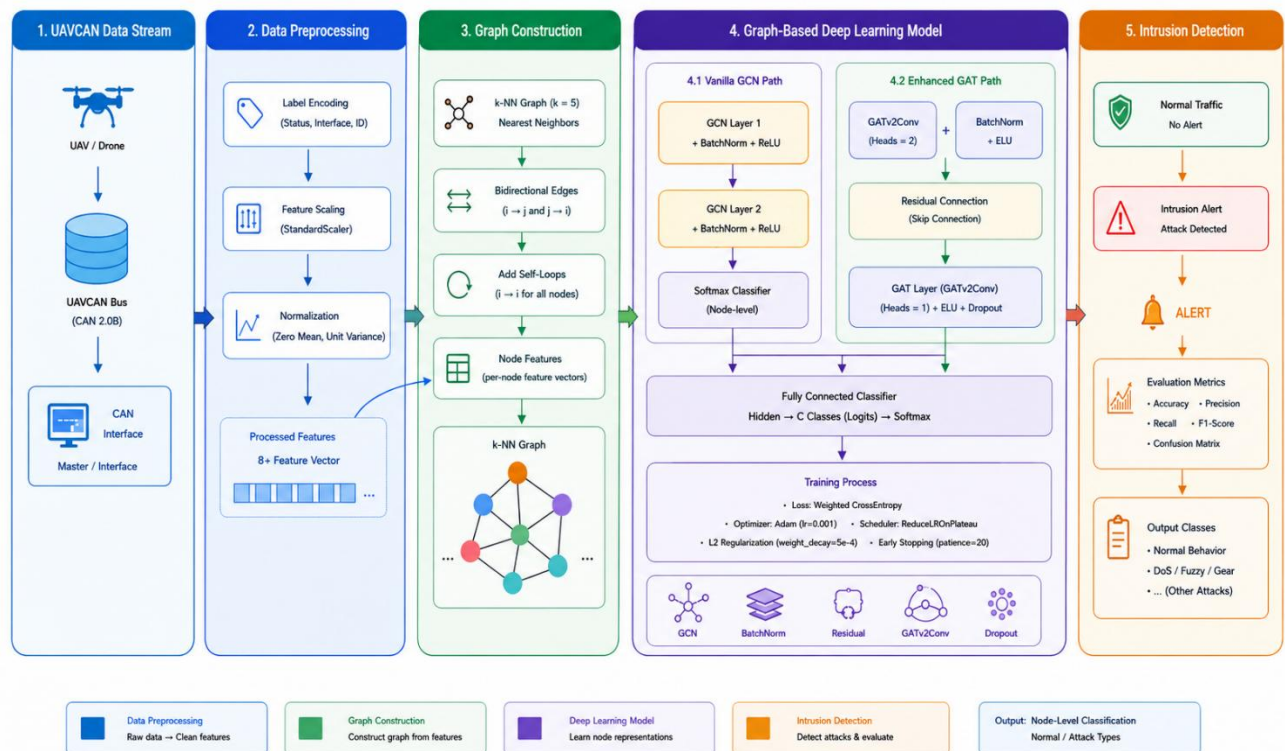


Figure 1. Framework of the proposed system

3.1 Dataset:

The UAVCAN attack dataset was generated in a physical testbed of PX4 Firmware drones equipped with UAVCAN v0 (DroneCAN) transceivers. Standard flight operations were recorded for normal CAN traffic; adversarial traffic was injected according to a controlled experimental protocol that includes three main attack types and their pairwise and triple combinations. In total, ten scenarios were designed, including intervals of 0.0015 s and 0.005 s, collection durations of 180-280 s, and frame counts of about 29,000 to more than 163,000 per class per scenario. We have used nine scenarios. Figure 2 shows the complete scenario metadata. The dataset is publicly available at the OCSLAB repository. The dataset features are shown in Table 1.

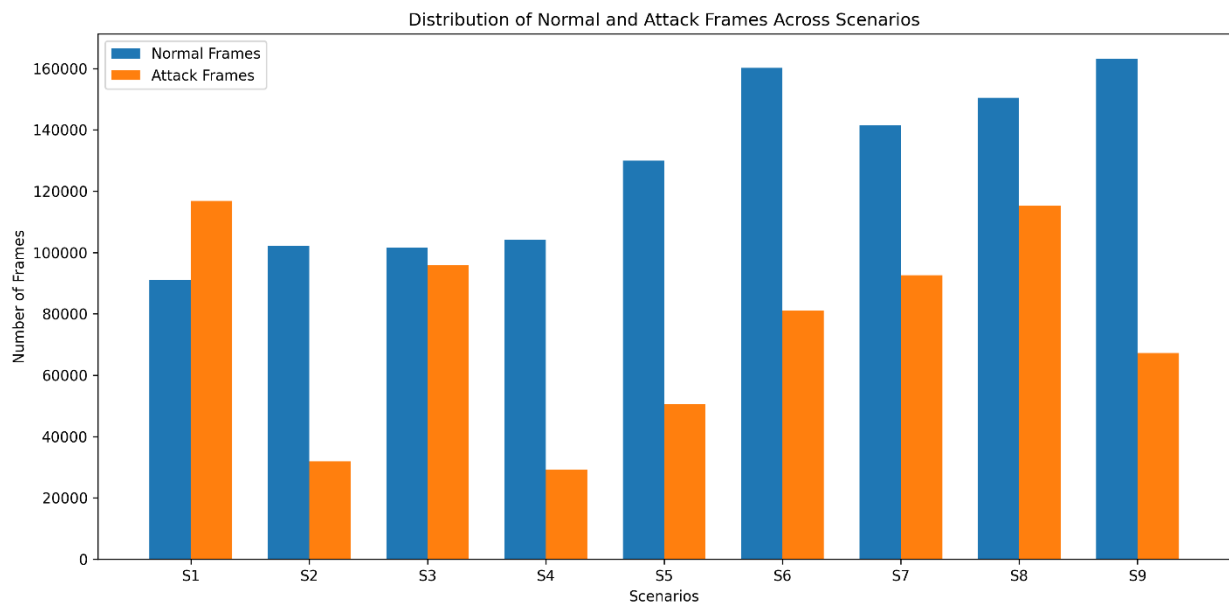


Figure 2. scenario metadata

Table 1. data features

Field name	Description
data	a hexadecimal string encoding the 8-byte CAN payload
Length	
Interface, ID	categorical metadata fields identifying the bus interface and message ID.

3.2 Preprocessing

The dataset consists of CAN bus traffic records stored in a CSV file with the following key columns:

3.2.1 Hex Payload Parsing

Each hexadecimal payload string is parsed into 8 integer-valued byte fures. For a raw payload string s_i composed of space-separated tokens $t_{i,1}, t_{i,2}, \dots, t_{i,8}$, the feature vector is defined as:

$$\mathbf{b}_i = [\text{int}(t_{i,1}, 16), \text{int}(t_{i,2}, 16), \dots, \text{int}(t_{i,8}, 16)]^T \in \mathbb{Z}^8 \quad (1)$$

where $\text{int}(\cdot, 16)$ converts a hexadecimal token to its decimal integer value. Any malformed token is replaced by zero to ensure robustness.

3.2.2 Feature Matrix Construction

Let N be the total number of samples. The categorical features *Interface* and *ID* are ordinally encoded via LabelEncoder. Combined with the parsed byte features, the raw feature matrix is:

$$\mathbf{X}_{\text{raw}} \in \mathbb{R}^{N \times d_{\text{raw}}} \quad (2)$$

where d_{raw} denotes the total number of numeric features (at least 10: 2 categorical + 8 byte features plus any remaining columns). Infinite and missing values are replaced by zero before normalization.

3.2.3 Standardization

Each feature column is standardized using zero-mean unit-variance normalization. For the j -th feature:

$$x'_{i,j} = \frac{x_{i,j} - \mu_j}{\sigma_j}, \quad \mu_j = \frac{1}{N_{\text{train}}} \sum_{i=1}^{N_{\text{train}}} x_{i,j}, \quad \sigma_j = \sqrt{\frac{1}{N_{\text{train}}} \sum_{i=1}^{N_{\text{train}}} (x_{i,j} - \mu_j)^2 + \varepsilon} \quad (3)$$

Parameters μ_j and σ_j are estimated exclusively on the training partition and applied to validation and test sets to prevent data leakage.

3.2.4 Data Splitting

The dataset is partitioned using a two-step stratified split: 70% for training, 10% for validation, and 20% for testing. This is a frame-level protocol and may expose the partitions to temporally correlated traffic from the same recording.

$$\text{Train: 70\%} \quad \text{Validation: 10\%} \quad \text{Test: 20\%} \quad (4)$$

Stratification preserves the original class distribution in every partition. Formally, for each class c with N_c samples:

$$N_c^{\text{split}} \approx r_{\text{split}} \cdot N_c, \quad r_{\text{split}} \in \{0.70, 0.10, 0.20\} \quad (5)$$

3.3 Graph Construction via K-Nearest Neighbours

Each CAN frame is treated as a node in a graph. The node feature matrix for a partition of n samples is $\mathbf{X} \in \mathbb{R}^{n \times d}$ and the class labels form the vector $\mathbf{y} \in \{0, 1, \dots, C-1\}^n$ where C is the number of classes.

3.3.1 K-NN Edge Construction

Edges are constructed independently within each partition using K-nearest neighbors in the normalized feature space. No edge connects training, validation, and test nodes, and test labels are not used during graph construction. The resulting graph is an artificial feature-similarity graph rather than a direct representation of the physical UAVCAN topology. The current $k=5$ choice is retained as the baseline setting and requires sensitivity analysis. Feature vector of node i , $\mathbf{x}_i$, is the operation used to select the indices of the top k nearest neighbors. For each node i , its k -nearest neighbors are identified: $kix_i x_j \text{arg topk}$

$$\mathcal{N}_k(i) = \arg \text{top}_{j \neq i} - k \frac{1}{\|\mathbf{x}_i - \mathbf{x}_j\|_2} \quad (6)$$

The edge set is made undirected (bidirectional) and augmented with self-loops:

$$\mathcal{E} = \{(i, j) \mid j \in \mathcal{N}_k(i)\} \cup \{(j, i) \mid j \in \mathcal{N}_k(i)\} \cup \{(i, i) \mid i = 1, \dots, n\} \quad (7)$$

Where i is the index of the sample dataset, start equation j is the neighboring node connected to start equation i , and start equation k is the number of nearest neighbors that will be considered in the graph. Symbol $\cup$ is combining the edge sets; start equation n is the total number of nodes in the graph. The resulting edge index tensor $\mathbf{E} \in \mathbb{Z}^{2 \times |\mathcal{E}|}$ is stored in COO (coordinate) format as required by PyTorch Geometric. A ring-graph fallback is applied if no K-NN edges can be formed (e.g., single-node partition).

3.3.2 Model Architecture: Improved GAT (GATv2)

Figure 3 displays the architecture of the improved GATv2 for detecting attacks in the UAV system.

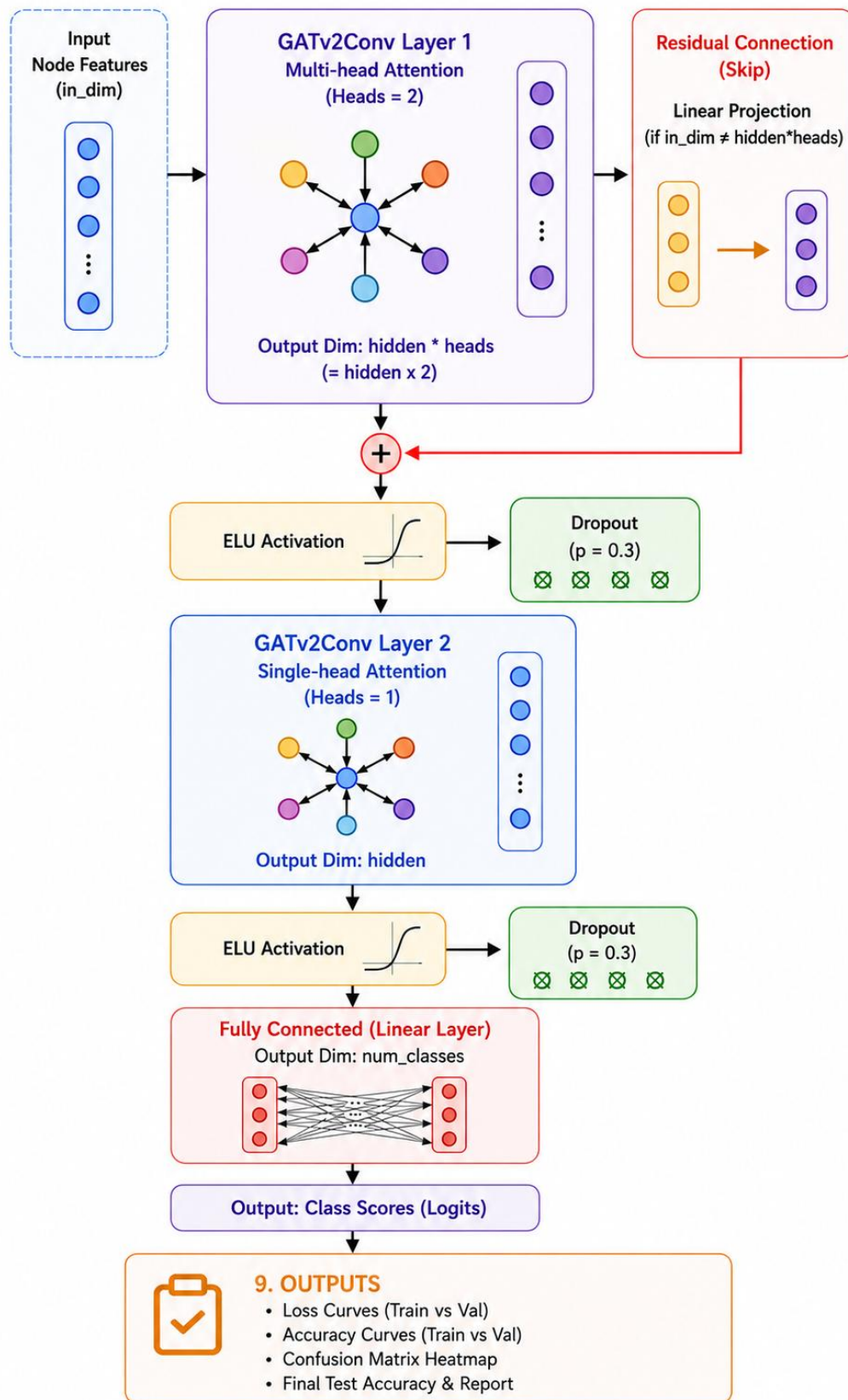


Figure 3. Improved GATv2 structure

3.3.3 Graph Attention Network v2 (GATv2Conv)

The model is built on GATv2Conv layers which addresses the expressiveness limitation of the original GAT by computing dynamic (non-static) attention coefficients. For a neighbor j , the unnormalised attention score is:

$$e_{ij} = \mathbf{a}^\top \text{LeakyReLU}(\mathbf{W}_1 \mathbf{x}_i + \mathbf{W}_2 \mathbf{x}_j) \quad (8)$$

where $\mathbf{W}_1, \mathbf{W}_2 \in \mathbb{R}^{d' \times d}$ are learnable projection matrices and $\mathbf{a} \in \mathbb{R}^{d'}$ is a learnable attention vector. Unlike standard GAT, both source and target features are transformed before the non-linearity, making the attention dynamic rather than static. The normalised attention coefficients over the neighbourhood are:

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k \in \mathcal{N}(i) \cup \{i\}} \exp(e_{ik})} \quad (9)$$

Where normalized between node i and node j , e_{ij} is Raw attention between node i and node j . The symbol $\exp$ is the exponential function, whereas $\cap \mathcal{N}(i) \cup \{i\}$ is the set of neighboring nodes connected to node i . $\exp$ is exponential function whereas $\mathcal{N}(i)$ is Set of neighboring nodes connected to node i .

$$\mathbf{h}_i^{(l+1)} = \sigma \left(\sum_{j \in \mathcal{N}(i) \cup \{i\}} \alpha_{ij}^{(l)} \mathbf{W}^{(l)} \mathbf{h}_j^{(l)} \right) \quad (10)$$

where σ is the Exponential Linear Unit activation function and $\mathbf{h}_i^{(0)} = \mathbf{x}_i$, $\mathbf{h}_j^{(l)}$ is indicates to feature vector of neighboring node j at layer l . The $\mathbf{W}^{(l)}, \alpha_{ij}^{(l)}$ weight matrix at layer l and Attention coefficient between nodes i, j neighboring nodes connected to node i and j is target node and neighbor node indexes.

3.3.4 Multi-Head Attention (Layer 1)

The first GATv2Conv layer uses $H = 2$ attention heads with concatenation. Each head $h \in \{1, \dots, H\}$ independently computes attention coefficients $\alpha_{ij}^{(h)}$ and produces an updated representation. The final representation concatenates all heads:

$$\mathbf{h}_i^{(1)} = \parallel_{h=1}^H \sigma \left(\sum_{j \in \mathcal{N}(i) \cup \{i\}} \alpha_{ij}^{(h)} \mathbf{W}^{(h)} \mathbf{h}_j^{(0)} \right) \in \mathbb{R}^{H \cdot d_{\text{hidden}}} \quad (11)$$

where $d_{\text{hidden}} = 64$ is the per-head hidden dimension, giving a concatenated dimension of $H \cdot d_{\text{hidden}} = 128$. $\mathbf{h}_i^{(1)}$ is defined the Output feature representation of node i at first GAT layer.

The $\mathbf{h}_j^{(0)}$ is represented the input feature vector of neighboring node j . $\mathbf{W}^{(h)}$ is weight matrix for attention head h and $\alpha_{ij}^{(h)}$ is represented the attention coefficient between nodes i and j for head h .

3.3.5 Residual Skip Connection

To facilitate gradient flow in deeper networks, a linear projection skip connection is added to the output of Layer 1. Because the input dimension d may differ from $H \cdot d_{\text{hidden}}$, a learnable projection $\mathbf{W}_{\text{skip}} \in \mathbb{R}^{(H \cdot d_{\text{hidden}}) \times d}$ is applied:

$$\tilde{\mathbf{h}}_i^{(1)} = \mathbf{h}_i^{(1)} + \mathbf{W}_{\text{skip}} \mathbf{x}_i \quad (12)$$

This residual addition mitigates the vanishing gradient problem and accelerates convergence.

$\mathbf{h}_i^{(1)}$ Output feature vector of node i GAT layer, $\mathbf{W}_{\text{skip}}$ transformation matrix used in the skip/residual connection. The $\mathbf{x}_i$ is training input feature vector of node i .

3.3.6 Second GATv2Conv Layer

The second GATv2Conv layer uses a single attention head ($H = 1$, no concatenation) and reduces the dimension back to $d_{\text{hidden}} = 64$:

$$\mathbf{h}_i^{(2)} = \sigma \left(\sum_{j \in \mathcal{N}(i) \cup \{i\}} \alpha_{ij}^{(2)} \mathbf{W}^{(2)} \tilde{\mathbf{h}}_j^{(1)} \right) \in \mathbb{R}^{d_{\text{hidden}}} \quad (13)$$

Dropout with probability $p = 0.3$ is applied to $\tilde{\mathbf{h}}_i^{(1)}$ before this layer and to $\mathbf{h}_i^{(2)}$ after this layer, regularising both internal representations.

3.3.7 Classification Head

The final node embeddings are passed through a linear classifier:

$$\hat{\mathbf{y}}_i = \mathbf{W}_{\text{fc}} \mathbf{h}_i^{(2)} + \mathbf{b}_{\text{fc}}, \quad \mathbf{W}_{\text{fc}} \in \mathbb{R}^{C \times d_{\text{hidden}}} \quad (14)$$

where $\hat{\mathbf{y}}_i \in \mathbb{R}^C$ are the raw logit scores for node i and the predicted class is $\hat{c}_i = \text{argmax}_c \hat{\mathbf{y}}_{i,c}$. $\hat{\mathbf{y}}_i$: predicted output for node i , representin the feature vector of the g class scores. $h_i^{(2)}$ is feature vector of final graph layer (layer 2). $W_{\text{fc}}, b_{\text{fc}}$ are the fully connected dense-layer weight matrix that maps hidden features to class space; cap C indicates the number of output classes, whereas d sub hidden I is used for representing the dimensionality of the node feature embedding before classification.

3.3.8 Class-Weighted Cross-Entropy Loss

To handle class imbalance, each class c is assigned an inverse-frequency weight:

$$w_c = \frac{1}{N_c^{\text{train}}}, \quad \tilde{w}_c = \frac{w_c \cdot C}{\sum_{c'=0}^{C-1} w_{c'}} \quad (15)$$

The weighted cross-entropy loss over all training nodes is:

Where C numbers of classes in the UAVs dataset, N_c^{train} number of training UAVs dataset belonging to class c . The w_c Raw inverse-frequency weight for class c and $\tilde{w}_c$ normalized class.

$$\mathcal{L}_{\text{train}} = -\frac{1}{n_{\text{train}}} \sum_{i=1}^{n_{\text{train}}} \tilde{w}_{y_i} \log \frac{\exp(\hat{\mathbf{y}}_{i,y_i})}{\sum_{c=0}^{C-1} \exp(\hat{\mathbf{y}}_{i,c})} \quad (16)$$

where y_i is the true class of node i and $\hat{\mathbf{y}}_{i,c}$ is its logit for class c . $\mathcal{L}_{\text{train}}$ is average training loss over the UAVs dataset, n_{train} number of training in UAVs dataset. The y_i is represented true class label of training i .

3.3.9 Optimization

Parameter updates use the Adam optimizer with weight decay (L2 regularisation). The per-parameter update rule at step t is:

$$\mathbf{m}_t = \beta_1 \mathbf{m}_{t-1} + (1 - \beta_1) \nabla_{\theta} \mathcal{L}, \quad \mathbf{v}_t = \beta_2 \mathbf{v}_{t-1} + (1 - \beta_2) (\nabla_{\theta} \mathcal{L})^2 \quad (17)$$

$$\hat{\mathbf{m}}_t = \frac{\mathbf{m}_t}{1 - \beta_1^t}, \quad \hat{\mathbf{v}}_t = \frac{\mathbf{v}_t}{1 - \beta_2^t} \quad (18)$$

$$\theta_t = \theta_{t-1} - \eta \frac{\hat{\mathbf{m}}_t}{\sqrt{\hat{\mathbf{v}}_t + \epsilon}} - \eta \lambda \theta_{t-1} \quad (19)$$

with hyperparameters $\eta = 0.001$ (learning rate), $\lambda = 5 \times 10^{-4}$ (weight decay), $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 10^{-8}$. The $\hat{\mathbf{m}}_t$ is bias-corrected first moment and $\hat{\mathbf{v}}_t$ is represented bias-corrected second moment

3.3.10 Gradient Clipping

To prevent exploding gradients common in graph neural networks, gradients are clipped by global norm before each parameter update:

$$\mathbf{g} \leftarrow \mathbf{g} \cdot \min \left(1, \frac{\tau}{\|\mathbf{g}\|_2} \right), \quad \tau = 1.0 \quad (20)$$

where $\mathbf{g}$ is the concatenated gradient vector over all model parameters, whereas τ Gradient clipping threshold.

3.3.11 Learning Rate Scheduling

A ReduceLROnPlateau scheduler monitors the validation loss. If no improvement is observed for $P_{\text{sched}} = 10$ consecutive epochs, the learning rate is scaled by a factor $\gamma = 0.5$:

$$\eta_{t+1} = \begin{cases} \gamma \cdot \eta_t & \text{if } \mathcal{L}_{\text{val}}^{(t)} \geq \min_{s \leq t - P_{\text{sched}}} \mathcal{L}_{\text{val}}^{(s)} \\ \eta_t & \text{otherwise} \end{cases} \quad (21)$$

Where η_t Learning rate at iteration/epoch t , η_{t+1} Updated learning rate and γ Decay factor and $\mathcal{L}_{\text{val}}^{(t)}$ is represented the validation loss at iteration t

3.3.12 Early Stopping

Training terminates when the validation loss fails to improve for $P_{\text{stop}} = 20$ consecutive epochs. The best model state θ^* is restored at the end of training:

$$\theta^* = \theta_{t^*}, \quad t^* = \underset{t}{\operatorname{argmin}} \mathcal{L}_{\text{val}}^{(t)} \quad (22)$$

Table 2 summarizes all key hyperparameters of the proposed framework.

Table 2: Summary of hyperparameters used in the proposed GATv2 IoT Botnet Detection framework.

Hyperparameter	Value
K-NN neighbours	5
Hidden dimension	64
Attention heads (L1)	2
Dropout probability	0.3
Learning rate	0.001
Weight decay	5e-4
Gradient clip norm	1.0
LR decay factor	0.5
LR patience	10
Early-stop patience	20
Max epochs	200
Train / Val / Test split	70% / 10% / 20%

3.4 Evaluation Metrics

Model performance is assessed using five complementary metrics computed per class and in aggregate. For a given class c with True Positives (TP), False Positives (FP), and False Negatives (FN):

$$\text{Precision}_c = \frac{TP_c}{TP_c + FP_c} \quad (24)$$

$$\text{Recall}_c = \frac{TP_c}{TP_c + FN_c} \quad (25)$$

$$F1_c = \frac{2 \cdot \text{Precision}_c \cdot \text{Recall}_c}{\text{Precision}_c + \text{Recall}_c} \quad (26)$$

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \quad (27)$$

The weighted F1-score averages per-class F1 scores weighted by class support. Confusion matrices are analyzed to quantify the False Negative Rate (FNR) and False Positive Rate (FPR), which carry distinct

operational costs in safety-critical UAV deployments: FNR governs the risk of undetected attacks; FPR governs alert fatigue.

4 Experimental Results and Analysis

This section shows the GATv2 classification results for all nine reported scenarios, aggregated by attack category. In each subsection, the per-class precision, recall, and F1-score from the classification report are given and followed by an analysis of the corresponding confusion matrix.

4.1 Experimental Setup

The proposed model was implemented and evaluated in a Kaggle cloud environment. The exact accelerator, software versions, CUDA/PyTorch/PyTorch-Geometric versions, random seed, and number of repetitions must be reported from the run record; the phrase “T4 GPU (or GPU-100-class accelerator)” is intentionally not used because it is not reproducible. This configuration enabled efficient processing of large-scale datasets and deep learning computation.

4.2.1 Scenarios 1 and 2: Flooding Attack

The flooding attack is a common DoS technique in UAV communication networks, in which an attacker floods the network with numerous UAVCAN frames to overload the bus and disrupt legitimate telemetry. Scenarios 1 and 2 are two distinct realizations of the flooding traffic, and they are evaluated separately using a train-test-validation split of the UAVCAN attack dataset. The GATv2 model was trained on a cloud-based Kaggle infrastructure and evaluated by standard classification metrics: precision, recall, F1 score, and overall accuracy. Table 3 shows the results of the GATv2 model. It is observed that the GATv2 model scores 99.66 in Scenario 1 and 98.56 in Scenario 2.

Table 3. Classification report for GATv2 on Scenarios 1 and 2 (Flooding Attack).

Scenario	Class	Precision (%)	Recall (%)	F1-Score (%)	Accuracy (%)	Weighted F1 (%)
Scenario 1	Attack	99.65	100.00	99.82	99.66	99.80
	Normal	100.00	99.55	99.77		
Scenario 2	Attack	94.42	100.00	97.14	98.56	98.59
	Normal	100.00	98.15	99.06		

Figure 4 shows the confusion matrices for Scenario 1 (plot a) and Scenario 2 (plot b). In Scenario 1, the GATv2 model achieved an attack-class recall of 1.00, with 23,363 true positives (TP), 0 false negatives (FN = 0), 81 misclassifications of normal frames as attacks (FP = 81), and 18,128 true negatives (TN). The corresponding false negative rate (FNR = 0.00%) and false positive rate (FPR = 0.45%) indicate that the model is very conservative in terms of missed detections under this scenario. In Scenario 2, the number of false negatives was still 0 (FN = 0), with all 6,386 attack samples being correctly identified (TP = 6,386). On the other hand, the number of false positives increased to 377 among 20,377 normal-class instances (TN = 20,000), yielding a false-positive rate of 1.85%. The FPR in Scenario 2 is higher than in Scenario 1, but no missed attack detection is observed in either scenario, indicating that the GATv2 model maintains full attack-class sensitivity across both flooding scenarios. The slight increase of false positives in Scenario 2 implies a slight overlap in traffic distribution between the benign and

attack feature spaces, which could be due to differences in flood packet density or inter-arrival time patterns between the two experimental setups.

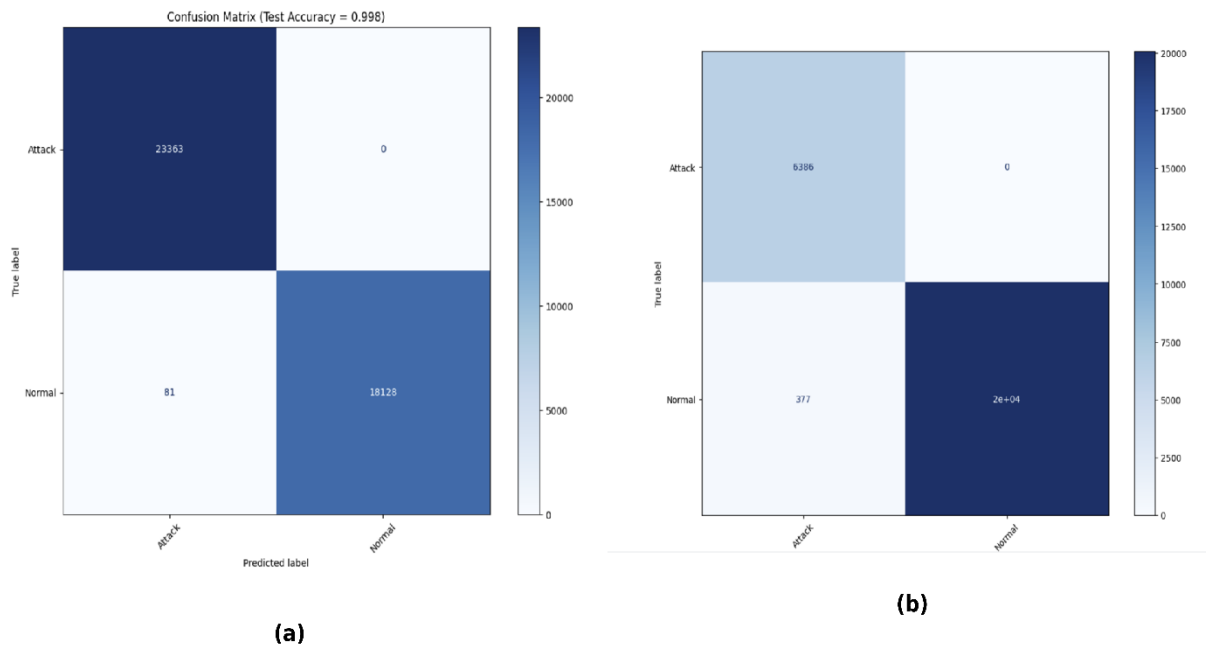


Figure 4. Confusion matrices for GATv2 on Flooding attack scenarios: (a) Scenario 1, (b) Scenario 2

Figure 5 illustrates the training and validation learning curves for Scenario 1 (plot a and b) and Scenario 2 (plots c and d). For Scenario 1, the cross-entropy loss (panel a) begins at approximately 0.62 (training) and 0.48 (validation) at epoch 0, then undergoes a steep monotonic decline during the first 50 epochs, converging to final values of approximately 0.02 and 0.03, respectively, at epoch 200. The validation loss tracks the training loss without divergence, and the train-validation gap remains below 0.01 throughout the 200-epoch schedule. The accuracy curves (panel b) reflect this behavior: both training and validation accuracy rise sharply in the initial phase, reaching above 0.95 within the first 20 epochs and plateauing near 0.998 and 0.995, respectively, showing similar learning trajectories; this does not prove the absence of overfitting. For Scenario 2, the initial loss values are higher (train ≈ 0.84 , validation ≈ 0.70), consistent with the more complex traffic distribution of this scenario. Convergence follows a similar path, with final losses near 0.015 and 0.035. The train-validation gap in loss widens marginally in the second half of training; however, validation accuracy stabilizes at approximately 0.98, while training accuracy remains at 0.99, indicating controlled and acceptable generalization behavior. Across both scenarios, no epoch-level reversal of validation loss is observed, and the absence of overfitting confirms that the 70/10/20 frame-level partition and the adopted regularisation strategy are adequate for the flooding attack detection task.

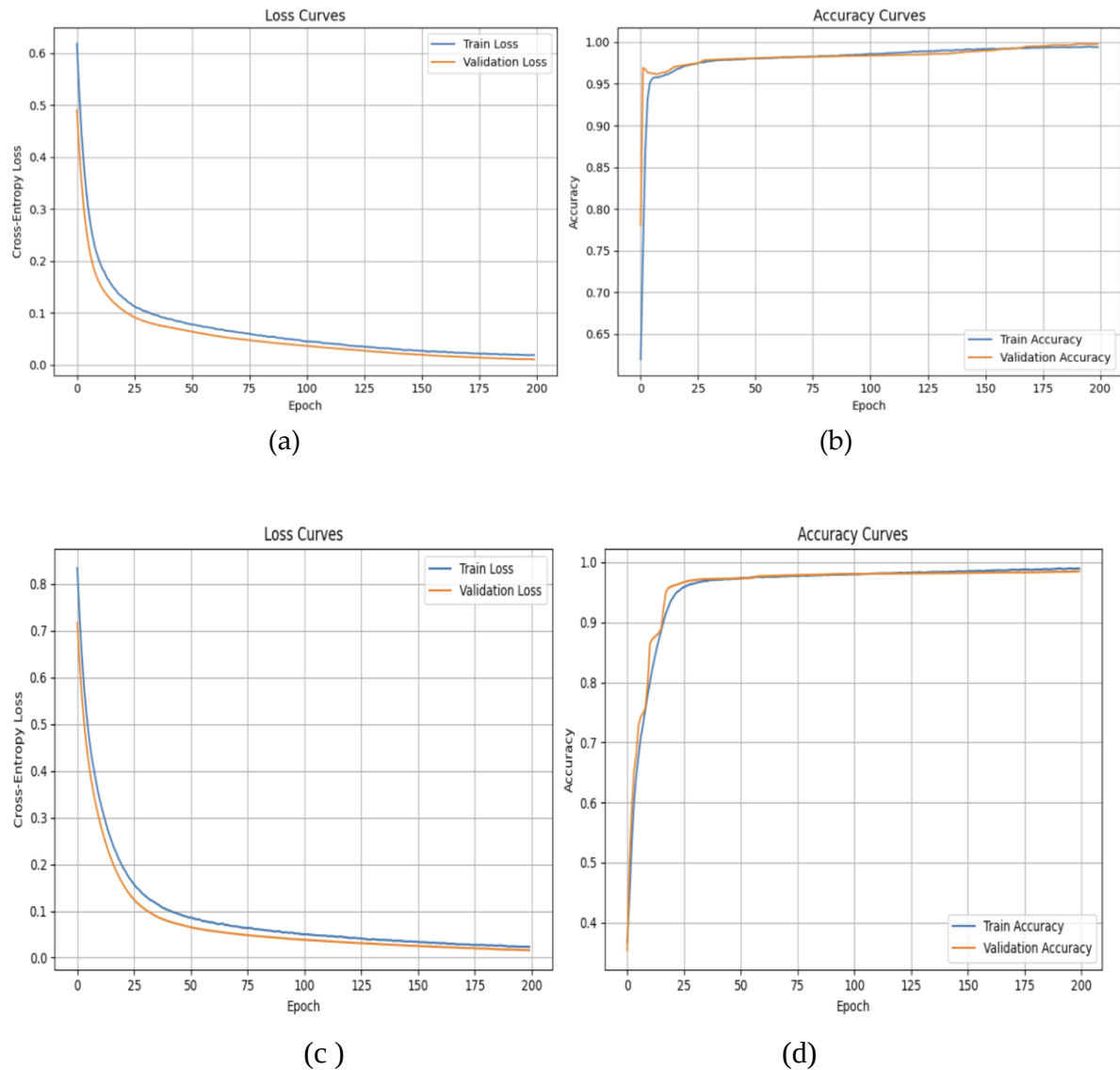


Figure 5. Training and validation learning curves for Flooding attack scenarios: (a) Scenario 1 loss, (b) Scenario 1 accuracy, (c) Scenario 2 loss, (d) Scenario 2 accuracy.

4.2.2 Scenarios 3, 4, and 5: Fuzzy Attack

Fuzzy attacks exploit the semantic integrity of the UAVCAN protocol by transmitting frames with randomized or malformed payload fields, leading to confusion among protocol parsers and reduced operational reliability of drone flight controllers. Scenarios 3, 4, and 5 are three increasingly difficult instances of this class of attack, differing in payload-randomization strategies and payload-injection rates. For each scenario, the GATv2 model was tested individually using the same training and evaluation protocol as applied for the flooding experiments. The GATv2 model achieved scores of 98.97%, 98.32%, and 90.57% in Scenarios 3, 4, and 5, respectively as shown in table 4.

Table 4. Classification report for GATv2 on Scenarios 3, 4, and 5 (Fuzzy Attack).

Scenario	Class	Precision (%)	Recall (%)	F1-Score (%)	Accuracy (%)	Weighted F1 (%)
Scenario 3	Attack	99.02	98.86	98.94	98.97	98.97

	Normal	98.92	99.08	99.00		
Scenario 4	Attack	94.88	97.60	96.22	98.32	98.33
	Normal	99.32	98.52	98.92		
Scenario 5	Attack	78.55	91.26	84.43	90.57	N/A (recompute)
	Normal	96.37	90.29	93.23		

Figure 6 shows the confusion matrices for Scenario 3 (plot a), Scenario 4 (plot b) and Scenario 5 (plot c). In Scenario 3, the model correctly classified 18,958 attack instances (TP = 18,958) with only 218 missed detections (FN = 218) (FNR = 1.14%). The false positive rate was 186/20,186 normal samples (TN = 20,000), obtaining an FPR of 0.92%. These near-symmetric error rates show a well-balanced classifier working under moderate fuzzy traffic conditions. In Scenario 4, the number of TP was 5,694 with 140 FNs (FNR = 2.40%). On the other hand, 307 normal frames were wrongly classified as attacks (FP = 307, TN = 20,534, FPR = 1.47%). The high FNR relative to Scenario 3 indicates that the fuzzy payload patterns in Scenario 4 overlap with signatures of normal traffic, making the model less sensitive. Scenario 5 suffered the most degradation, with 884 attack samples missed (FN = 884, FNR = 8.74%) and 2,522 normal frames flagged incorrectly (FP = 2,522, TN = 23,477, FPR = 9.70%). The large increase in both error types suggests a much more difficult classification boundary for Scenario 5, likely due to greater variability in the fuzzy attack signatures that the GATv2 graph attention mechanism.

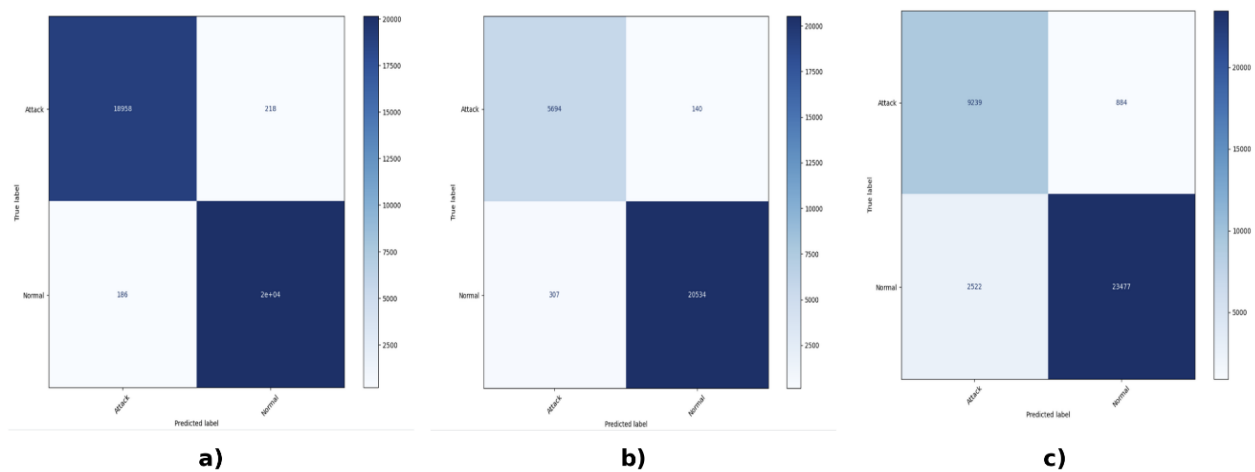
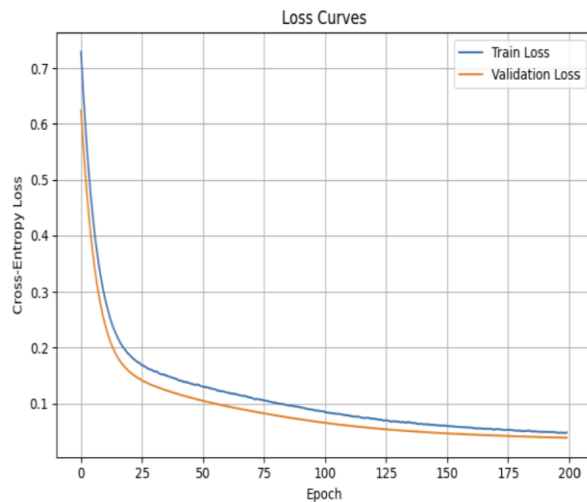


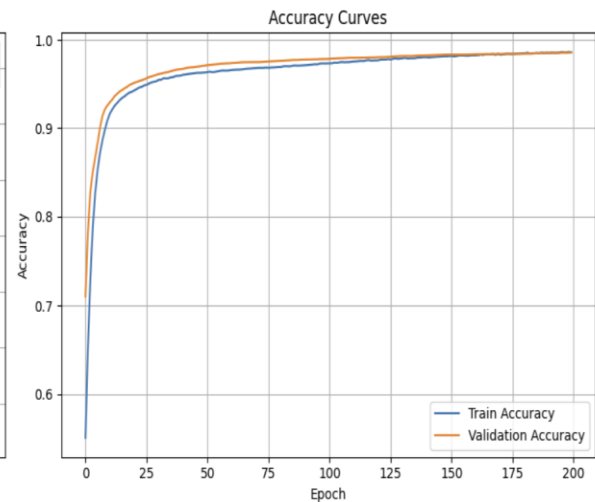
Figure 6. Confusion matrices for GATv2 on Fuzzy attack scenarios: (a) Scenario 3, (b) Scenario 4, and (c) Scenario 5.

The learning curves of Scenario 3 (plots a, b), Scenario 4 (plots c, d), and Scenario 5 (plots f, g) are shown in Figure 7. In Scenario 3 the training loss decreases from about 0.72 and the validation loss from about 0.62, both converging to 0.04 and 0.03, respectively, at epoch 200. A peculiarity of this case is, that the validation loss is always lower than the training loss for the full training schedule, since dropout regularisation is switched off during the validation forward pass. The corresponding accuracy curves confirm this behavior: the validation accuracy (≈ 0.99) is slightly higher than the training accuracy (≈ 0.98) at convergence, indicating strong generalization with no sign of overfitting. Scenario 4 also shows qualitatively similar dynamics (initial losses of approx. 0.87 and 0.76 for training and validation, respectively, converging to approx. 0.06 and 0.04), and final accuracy values of 0.97 (training) and 0.98 (validation) confirm well-controlled learning. On the other hand, Scenario 5 achieves considerably higher residual losses (train ≈ 0.27 , validation ≈ 0.25) and lower terminal accuracy (train ≈ 0.89 , validation ≈ 0.90), indicating a more ambiguous distribution of attack signatures. Training and validation curves are tightly coupled throughout the 200-epoch schedule despite the elevated loss floor, and no

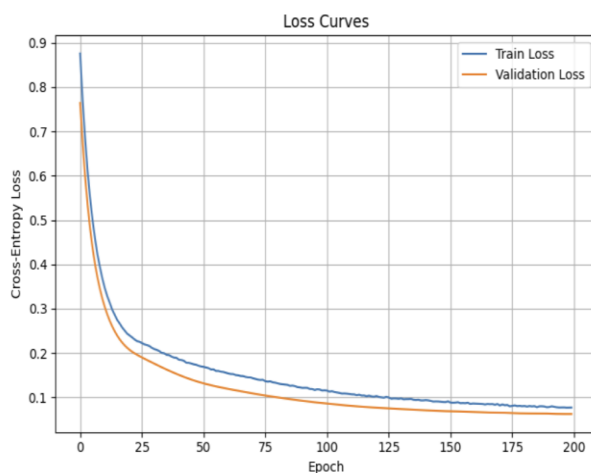
epoch exhibits a diverging validation loss, confirming the model generalizes consistently even under the challenging Scenario 5 conditions. The reduction in loss after epoch 50 in Scenario 5 is smaller, suggesting that performance could be improved by increasing the number of training epochs or by using a larger model architecture.



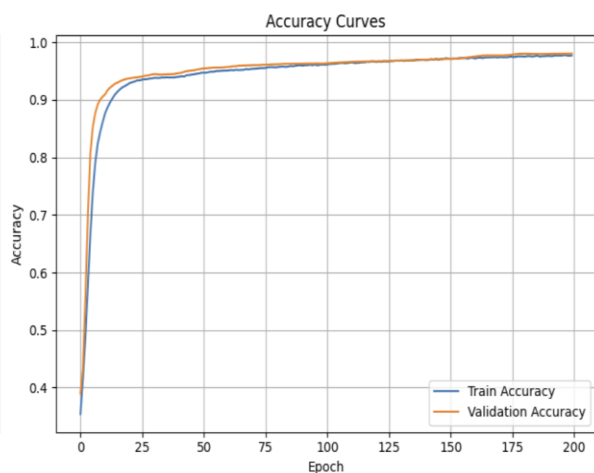
(a)



(b)



(d)



(c)

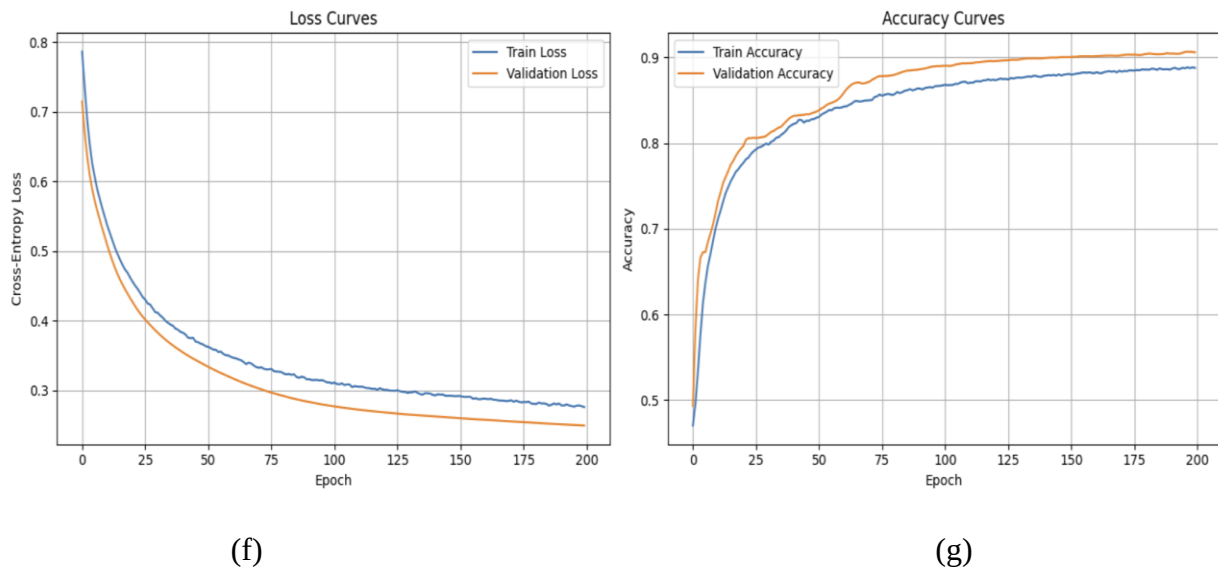


Figure 7. Training and validation learning curves for Fuzzy attack scenarios: (a) Scenario 3 loss, (b) Scenario 3 accuracy, (c) Scenario 4 loss, (d) Scenario 4 accuracy, (f) Scenario 5 loss, (g) Scenario 5 accuracy.

4.2.3 Scenario 6: Replay Attack

In the UAVCAN context, a replay attack consists of capturing and re-injecting previously recorded legitimate bus frames, exploiting the protocol's lack of native sequence-integrity mechanisms. The attack poses a particularly challenging detection problem because the replayed frames are structurally valid; thus, detection relies on temporal and contextual graph-level anomaly signals rather than packet-level syntactic features. Scenario 6 is the only replay-attack scenario in the UAVCAN dataset and is evaluated using the same GATv2 architecture and training protocol as the other scenarios. The proposed system achieved 91.93% in Scenario 6, as shown in Table 5.

Table 5. Classification report for GATv2 on Scenario 6 (Replay Attack).

Scenario	Class	Precision (%)	Recall (%)	F1-Score (%)	Accuracy (%)	Weighted F1 (%)
Scenario 6	Attack	90.29	85.14	87.64	91.93	91.87
	Normal	92.69	95.36	94.12		

Figure 8 shows the confusion matrix for Scenario 6. The GATv2 model correctly detected 13809 attack frames (TP = 13809) but missed 2409 instances (FN = 2409). The model's FNR is 14.86%. FP = 1,484. This shows that 1,484 frames were misclassified as attacks. TN = 30,563. FPR = 4.63%. In the normal traffic class, the very high FNR of Scenario 6 distinguishes it from all other scenarios studied, showing the inherent difficulty of detecting replayed frames that are syntactically indistinguishable from legitimate traffic. The FNR (14.86%) and FPR (4.63%) asymmetry reveals that the GATv2 model is more prone to missing replay attacks than to false alarms on benign traffic. This behavior is consistent with replay attacks, in which injected frames mimic previously observed graph-level communication patterns, thereby diminishing the discriminative power of the attention-weighted edge features that GATv2 relies on. To improve detection in this scenario, it is likely necessary to combine the static graph structural features with temporal sequence modeling or cryptographic message authentication.

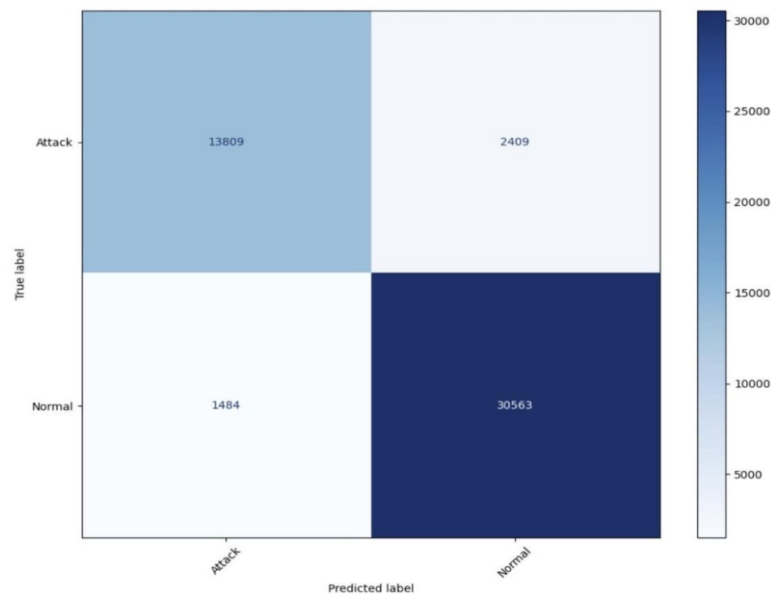


Figure 8. Confusion matrix for GATv2 on Scenario 6 (Replay Attack).

The learning curves for cross-entropy loss and accuracy in Scenario 6 are shown in Figure 9. The training loss is around 0.75 and the validation loss is around 0.69. Both curves show a steep decline in the first 50 epochs, then flatten out and decrease a little. The training loss converges at approximately 0.22 and the validation loss at approximately 0.21 at epoch 200. This residual loss is higher than in the flooding scenarios, in line with the more ambiguous classification boundary in the replay setting. The accuracy curves show a clear transient dip in the validation accuracy from epochs 5 to 15 before the curves resume a steady upward trend. This short-term instability due to stochastic mini-batch sampling in the early optimization phase is not determined. The training accuracy converges to around 0.91 and the validation accuracy converges to around 0.92 by epoch 200. The slightly higher validation accuracy than training accuracy in the late phase of training is consistent with the effect of dropout regularization found in other settings. It is an important confirmation that no overfitting is happening as the validation loss does not increase at any point during the 200 epoch schedule. The final accuracy (≈ 0.92) is not able to close the gap to the theoretical maximum, which suggests that the replay detection task is inherently difficult and that the model can be improved beyond the current GATv2 architecture to reach performance parity with flooding and fuzzy attack scenarios.

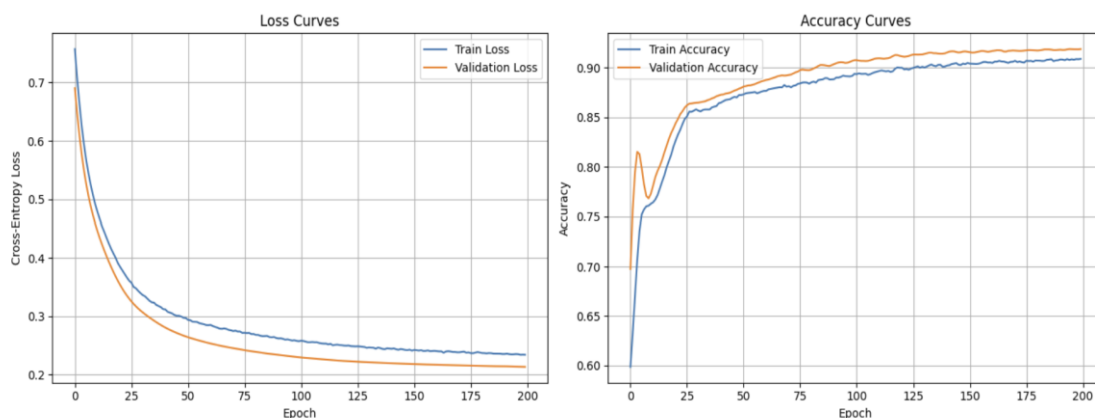


Figure 9. Training and validation learning curves for Scenario 6 (Replay Attack): loss curve and accuracy curve.

4.2.4 Scenarios 7, 8, and 9: Compound Multi-Vector Attacks

Scenarios 7 through 9 are compound attack scenarios that combine two or more different attack vectors within a single evaluation dataset, simulating the realistic threat of coordinated multi-vector intrusions. Scenario 7 combines Flooding and Fuzzy attacks, Scenario 8 Fuzzy and Replay attacks, and Scenario 9 a 3-vector compound attack of Flooding, Fuzzy, and Replay vectors. Such compound scenarios are the most realistic and operationally challenging evaluation conditions in the UAVCAN dataset. In this experiment, the proposed system demonstrated accuracy across three scenarios (7, 8, and 9), with Scenario 7 achieving 98.99%, Scenario 8 achieving 95.67%, and Scenario 9 achieving 94.08%, as shown in Table 6.

Table 6. Classification report for GATv2 on Scenarios 7, 8, and 9 (Compound Multi-Vector Attacks).

Scenario	Class	Precision (%)	Recall (%)	F1-Score (%)	Accuracy (%)	Weighted F1 (%)
Scenario 7	Attack	98.78	98.65	98.72	98.99	98.98
	Normal	99.12	99.20	99.16		
Scenario 8	Attack	97.72	92.17	94.87	95.67	95.65
	Normal	94.25	98.35	96.26		
Scenario 9	Attack	88.60	91.50	90.02	94.08	94.11
	Normal	96.44	95.14	95.79		

The confusion matrices of Scenario 7, Scenario 8, and Scenario 9 are shown in Figure 10 (plots a, b, and c, respectively). In Scenario 7 (Flooding + Fuzzy), GATv2 yielded 18,274 true positives (TP), with 249 missed attacks (FN = 249, FNR = 1.35%) and 224 false alarms (FP = 224, TN = 28,086, FPR = 0.79%). This shows that the model can generalize its flooding and fuzzy discriminative features to the joint attack distribution, with little performance degradation compared to single-vector scenarios. In Scenario 8 (Fuzzy + Replay), the TP count is 21,258, but the FN count is much higher, 1,804, (FNR = 7.83%), with only 495 false positives (TN = 30,000, FPR = 1.62%). The replay attack component is the root cause of the undesirable increase in FNR relative to FPR in Scenario 8, where the semantically valid frame structure conceals the anomaly signal that GATv2 relies on for attack classification. Scenario 9 has an intermediate error profile, with 12,308 true positives (TP), 1,143 false negatives (FN = 1,143, FNR = 8.49%), and 1,583 false positives (FP = 1,583, TN = 31,042, FPR = 4.85%). In Scenario 9, the simultaneous increase of FNR and FPR is a manifestation of the uncertainty introduced by the three-vector attack mixture. The overlap among the feature representations of the flooding, fuzzy, and replay components results in ambiguous decision boundaries. Nevertheless, the overall accuracy of 94.08% indicates that the GATv2 architecture retains strong discriminative power even under the most complex multi-vector conditions.

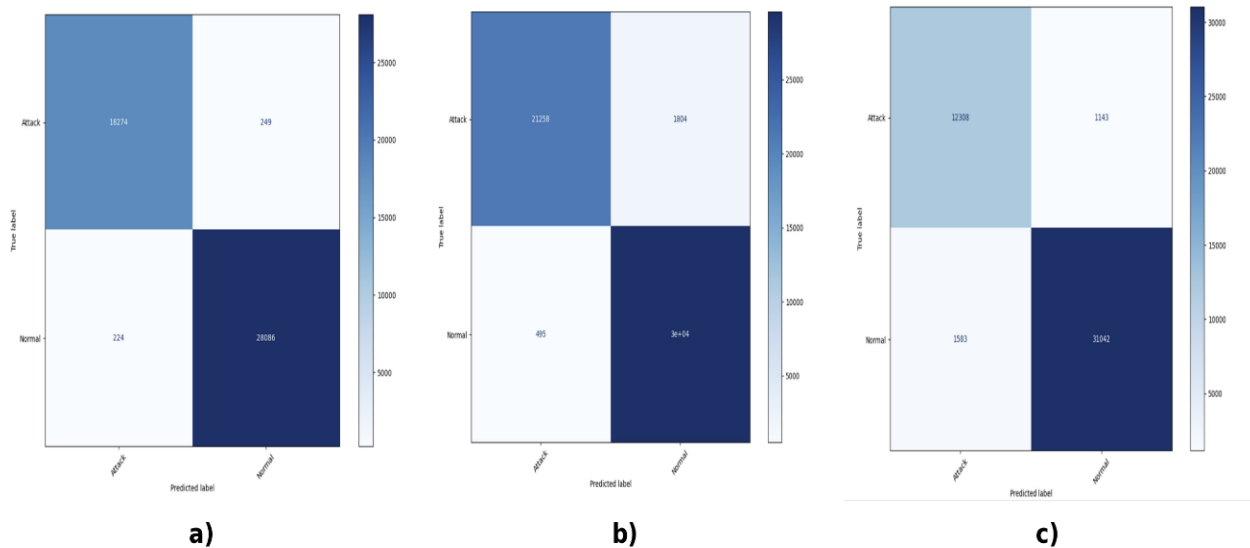
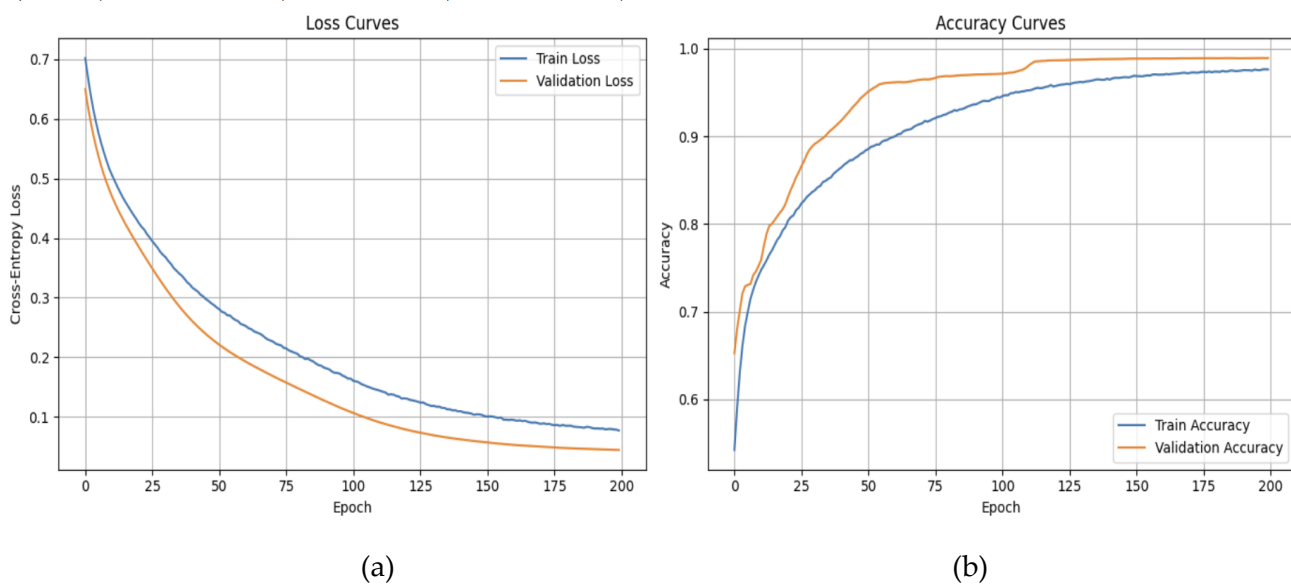


Figure 10. Confusion matrices for GATv2 on compound multi-vector attack scenarios: (a) Scenario 7 (Flooding+Fuzzy), (b) Scenario 8 (Fuzzy+Replay), (c) Scenario 9 (Flooding+Fuzzy+Replay).

Figure 11 shows the training and validation learning curves for three compound multi-vector attack scenarios (7, 8, and 9) over 200 epochs, presented as a 3×2 grid of loss and accuracy subplots. In all three cases, the Cross-Entropy Loss and Accuracy curves show a similar two-stage learning behavior: a steep, rapid improvement in the first 50 epochs, followed by a slow asymptotic convergence to terminal values, which is typical for a well-initialized model with a stable optimization regime. Scenario 7 achieves the best terminal performance, with validation accuracy ~ 0.99 and training loss ~ 0.08 . The feature space of its attack composition is highly discriminative and well separated. Scenarios 8 and 9 converge to competitive but relatively modest terminal accuracies of ~ 0.95 and ~ 0.93 - 0.94 , respectively, with a shared loss plateau near 0.15 indicative of an inherent generalization ceiling due to the greater inter-class ambiguity of their more complex combinations of attack vectors. Notably, in Scenario 9, there are mild transient oscillations in accuracy during epochs 25-75, consistent with a more challenging, noisily distributed decision boundary, but these oscillations settle into stable convergence prior to epoch 100. What is common to all subplots is that the validation metrics are always higher than the training metrics, most likely since regularization mechanisms (e.g. dropout) are switched off during inference.



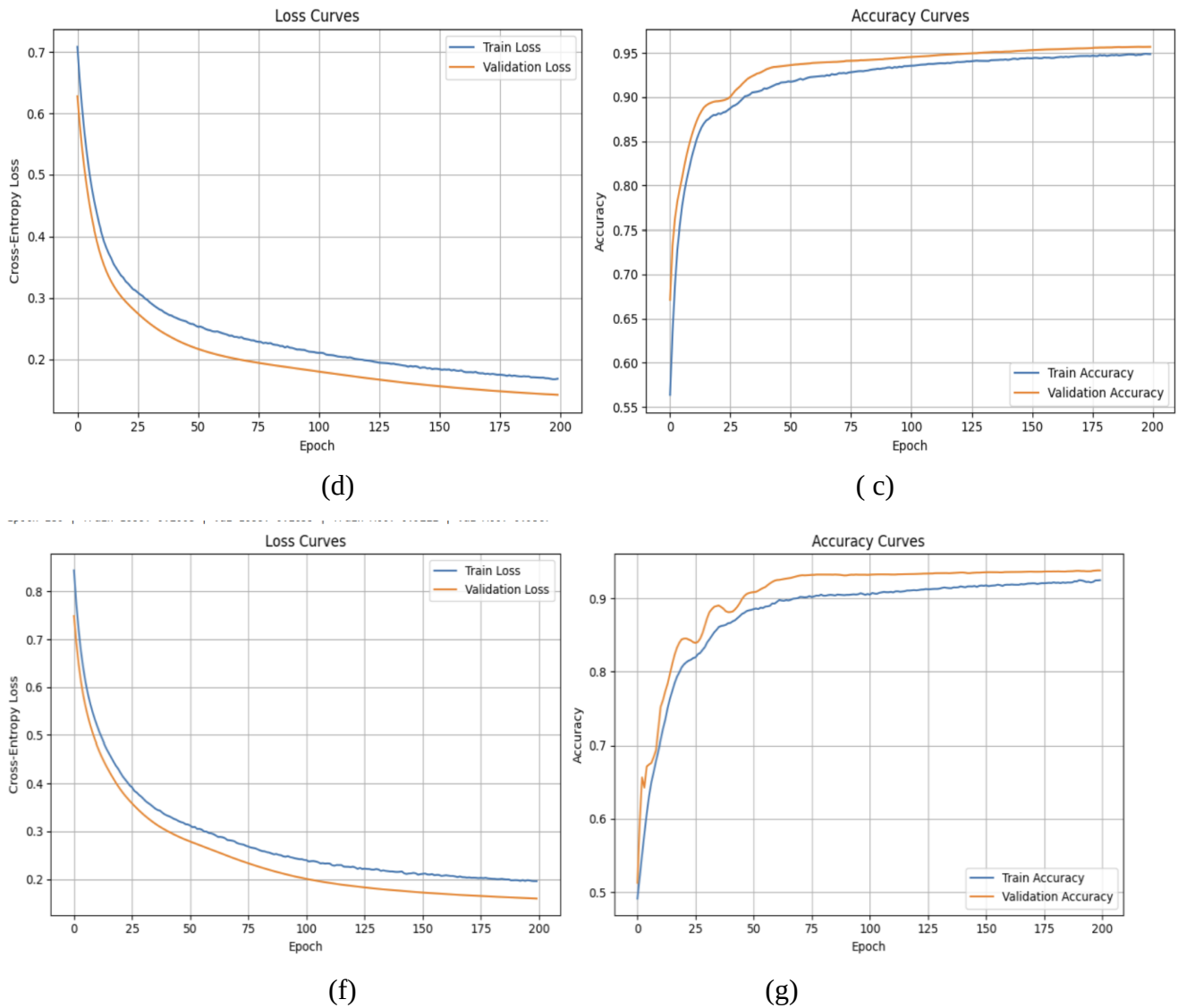


Figure 11. Training and validation learning curves for compound multi-vector attack scenarios: (a) Scenario 7 loss, (b) Scenario 7 accuracy, (c) Scenario 8 loss, (d) Scenario 8 accuracy, (f) Scenario 9 loss, (g) Scenario 9 accuracy.

5 Discussion

The experimental results collectively establish GATv2 as a high-performing baseline for UAVCAN intrusion detection, with weighted F1-scores exceeding 98% across six of the nine reported scenarios. Table 7 consolidates the classification report metrics across all reported scenarios, enabling a systematic comparison of GATv2 performance as a function of attack type, injection intensity, and compositional complexity. Results are ordered by scenario number and annotated with the attack category.

Table 7. Cross-scenario GATv2 classification performance summary

Scenario	Attack Type	Attack Precision (%)	Attack Recall (%)	Attack F1 (%)	Normal F1 (%)	Accuracy (%)	Weighted F1 (%)
1	Flooding	99.65	100.00	99.82	99.77	99.66	99.80
2	Flooding	94.42	100.00	97.14	99.26	98.56	98.59

3	Fuzzy	99.02	98.86	98.94	99.00	98.97	98.97
4	Fuzzy	94.88	97.60	96.22	98.92	98.32	98.33
5	Fuzzy	78.55	91.26	84.43	93.23	90.57	—
6	Replay	90.29	85.14	87.64	94.12	91.93	91.87
7	Flood+Fuzzy	98.78	98.65	98.72	99.16	98.99	98.98
8	Fuzzy+Replay	97.72	92.17	94.87	96.26	95.67	95.65
9	Flooding+Fuzzy+Replay	88.60	91.50	90.02	95.79	94.08	94.11

We attribute the approach's success in Flooding detection to the clear graph-structural footprint of high-frequency, repetitive frames, resulting in dense, homogeneous neighborhoods that the attention mechanism can distinguish from Normal, heterogeneous traffic. This result is consistent with the more general finding in the network intrusion detection literature that volume-based attacks are the most amenable to machine-learning classification.

The drop in performance seen in Scenario 5 (Fuzzy, 90.57%) and Scenario 6 (Replay, 91.93%) demonstrates two different failure modes. In Scenario 5, the confusion matrix shows a higher number of false positives (2,522 Normal frames classified as Attack), indicating that the model expands its Attack decision region when it faces long-duration Fuzzy injection. This is because the normal class distribution is statistically contaminated over long collection windows, smearing out the inter-class boundary in the learned embedding space. The dominant error mode flips in Scenario 6, with 2,409 Attack frames misclassified as Normal (false negatives). This is consistent with the structural stealth of replay traffic. Taken together, these results suggest that GATv2's attention mechanism is more effective at identifying structural anomalies.

The results from the compound scenario add another dimension for analysis. Scenario 7 (Flooding & Fuzzy) performs almost the best. Thus, if a strong-signal attack type emerges concurrently with a weak-signal type, the strong-signal one would dominate the classification. This finding has important consequences for the design of adversarial attacks. A sophisticated attacker who deliberately mixes high- and low-signature attack types can aid detection by introducing high-signature traffic alongside stealthy payloads. We do not observe this effect in scenarios 8 and 9, which are combinations of two covert attack types (Fuzzy and Replay), confirming that compounding low-signal attack types is more detrimental to model performance than compounding high-signal attack types. The operational costs of the asymmetric error distributions across scenarios differ from a methodological point of view. False negatives are more severe than false positives (false alarms) in safety-critical UAV systems. An undetected attack on the flight-control bus can result in loss of the vehicle or mission failure. In contrast, too many false positives can reduce the operator's trust in the IDS, and lead to alarm fatigue. These trade-offs imply that for production deployment, GATv2 should use class-weighted loss functions or asymmetric decision thresholds, calibrated to the threat model of the deployment environment.

The results are compared with existing works using the same dataset, HCRL/OCSLAB UAVCAN Attack Dataset, in Table 8 [30]. Studies based on other UAV, CAN-bus, IoT or synthetic datasets were excluded. The retained studies use different class definitions, preprocessing procedures, scenario groupings, data splits and centralized or federated learning settings, so the reported values should be interpreted as descriptive context rather than a direct ranking.

Table 8. Comparison with existing intrusion-detection studies using the same HCRL/OCSLAB UAVCAN Attack Dataset.

Study	Method	Task/dataset coverage	Reported performance	Comparison note
Hong and Yoo (2024) [31]	Feature selection, heterogeneous	Multi-attack classification using the HCRL	Approximately 97% accuracy and F1-score	Strong interpretable ensemble result, but the class

	ensemble, and SHAP explanation	UAVCAN Attack Dataset		formulation and evaluation protocol differ from the present scenario-wise binary tests.
Majumder et al. (2024) [32]	GCN, GAT, GraphSAGE, graph transformer, and LSTM baseline	Scenario-level graph intrusion detection across the ten UAVCAN scenarios	Reported scenario/model accuracies range from about 81% to nearly 100%; some highly imbalanced cases have substantially lower F1-scores	Most closely related graph-based comparison; graph construction, label definitions, and split design are not identical to the present GATv2 study.
Wei et al. (2026) [33]	UpsFed-IDS using U-shaped split federated learning and a 1D-CNN	Six UAVCAN data types under IID and non-IID federated partitions	IID: 90.88% accuracy and 90.80% F1; non-IID: 95.40% accuracy and 93.59% F1	Evaluates privacy-preserving distributed learning rather than a centralized scenario-wise GNN, so the metrics answer a different deployment question.
Present study	GATv2 on partition-isolated K-NN feature-similarity graphs	Binary Normal-versus-Attack classification in nine reported UAVCAN scenarios	Accuracy: 90.57–99.66%; weighted F1: 90.77–99.80%	Provides detailed scenario-specific error analysis, including false-positive and false-negative behavior; recording-aware, repeated-seed, and controlled same-split baseline experiments remain required.

6 Conclusion

In this study, we performed a comprehensive evaluation of the GATv2 graph attention network for binary intrusion detection on UAV CAN communication networks. We tested the model across 9 of 10 attack scenarios, including Flooding, Fuzzy, Replay, and compound attacks. In the most tractable Flooding scenario, the proposed model achieved a weighted F1-score of 99.80% under the original frame-level protocol and exceeded 98% in most single-vector scenarios, providing a robust baseline for graph-based UAV IDS research. Performance degraded to 90.57% and 91.93% under long Fuzzy and Replay conditions, respectively, revealing scenario-specific failure modes arising from statistical boundary blur and structural stealth. The confusion matrix analysis offered a more detailed picture than the quantitative summary, indicating asymmetric error patterns with different operational consequences: for the covert attack scenarios, the false negatives prevail, risking undetected intrusion, while for the fuzzy scenarios, the false positives prevail, risking alert reliability.

Cross-scenario comparison showed an observed ordering in the supplied scenarios, but attack signal strength was not quantitatively defined and therefore a formal monotonic relationship is not claimed: Flooding attacks are most detectable, Replay attacks are least detectable, and compound scenarios depend on whether a high-signal attack type is present among co-occurring threats. These findings collectively advance the understanding of graph-attention-based anomaly detection in resource-

constrained, time-critical IoT communication environments. Future work will address the identified limitations through replay-specific feature engineering, asymmetric loss calibration, and extensions to multi-head attention, aiming to achieve robust detection across all 9 UAVCAN attack scenarios.

Data availability: The authoritative dataset is the HCRL/OCSLAB UAVCAN Attack Dataset:

<https://ocslab.hksecurity.net/Datasets/uavcan-attack-dataset> . The official page states that the dataset was collected using UAVCAN v0/DroneCAN and contains Flooding, Fuzzy, and Replay attacks _

REFERENCES

- [1] Diez, Y.; Kentsch, S.; Fukuda, M.; Caceres, M.L.L.; Moritake, K.; Cabezas, M. Deep Learning in Forestry Using UAV-Acquired RGB Data: A Practical Review. *Remote Sensing* 2021, 13, 2837. DOI: 10.3390/rs13142837.
- [2] Mogili, U.R.; Deepak, B.B.V.L. Review on Application of Drone Systems in Precision Agriculture. *Procedia Computer Science* 2018, 133, 502–509. DOI: 10.1016/j.procs.2018.07.063.
- [3] Tsouros, D.C.; Bibi, S.; Sarigiannidis, P.G. A Review on UAV-Based Applications for Precision Agriculture. *Information* 2019, 10, 349. DOI: 10.3390/info10110349.
- [4] Arafat, M.Y.; Alam, M.M.; Moh, S. Vision-Based Navigation Techniques for Unmanned Aerial Vehicles: Review and Challenges. *Drones* 2023, 7, 89. DOI: 10.3390/drones7020089.
- [5] Minaeian, S.; Liu, J.; Son, Y.J. Vision-Based Target Detection and Localization via a Team of Cooperative UAV and UGVs. *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 2016, 46, 1005–1016. DOI: 10.1109/TSMC.2015.2491878.
- [6] Herisse, B.; Russotto, F.X.; Hamel, T.; Mahony, R. Hovering Flight and Vertical Landing Control of a VTOL Unmanned Aerial Vehicle Using Optical Flow. In *Proceedings of IROS 2008*, pp. 801–806. DOI: 10.1109/IROS.2008.4650731.
- [7] Gupta, L.; Jain, R.; Vaszkun, G. Survey of Important Issues in UAV Communication Networks. *IEEE Communications Surveys & Tutorials* 2016, 18, 1123–1152. DOI: 10.1109/COMST.2015.2495297.
- [8] Shakhathreh, H.; et al. Unmanned Aerial Vehicles: A Survey on Civil Applications and Key Research Challenges. *IEEE Access* 2019, 7, 48572–48634. DOI: 10.1109/ACCESS.2019.2909530.
- [9] Bekmezci, I.; Sahingoz, O.K.; Temel, Ş. Flying Ad-Hoc Networks (FANETs): A Survey. *Ad Hoc Networks* 2013, 11, 1254–1270. DOI: 10.1016/j.adhoc.2012.12.004.
- [10] Kirienko, A. UAVCAN Specification (v1.0). OpenCyphal Organization, 2020. No DOI available; official specification.
- [11] Checkoway, S.; et al. Comprehensive Experimental Analyses of Automotive Attack Surfaces. In *Proceedings of the USENIX Security Symposium*, 2011. No DOI available.
- [12] Stallings, W. *Cryptography and Network Security: Principles and Practice*. Pearson, 2017. No DOI available (book).
- [13] Liao, H.-J.; Lin, C.-H.R.; Lin, Y.-C.; Tung, K.-Y. Intrusion Detection System: A Comprehensive Review. *Journal of Network and Computer Applications* 2013, 36, 16–24. DOI: 10.1016/j.jnca.2012.09.004.
- [14] Kim, G.; Lee, S.; Kim, S. A Novel Hybrid Intrusion Detection Method Integrating Anomaly Detection with Misuse Detection. *Expert Systems with Applications* 2014, 41, 1690–1700. DOI: 10.1016/j.eswa.2013.08.066.
- [15] Yin, C.; Zhu, Y.; Fei, J.; He, X. A Deep Learning Approach for Intrusion Detection Using Recurrent Neural Networks. *IEEE Access* 2017, 5, 21954–21961. DOI: 10.1109/ACCESS.2017.2762418.
- [16] Javaid, A.; Niyaz, Q.; Sun, W.; Alam, M. A Deep Learning Approach for Network Intrusion Detection System. In *Proceedings of the EAI International Conference*, 2016. DOI: 10.4108/eai.3-12-2015.2262516.
- [17] Shone, N.; Ngoc, T.N.; Phai, V.D.; Shi, Q. A Deep Learning Approach to Network Intrusion Detection. *IEEE Transactions on Emerging Topics in Computational Intelligence* 2018. DOI: 10.1109/TETCI.2017.2772792.
- [18] Yang, Z.; Zhang, Y.; Zeng, J.; Yang, Y.; Jia, Y.; Song, H.; Lv, T.; Sun, Q.; An, J. AI-Driven Safety and Security for UAVs: From Machine Learning to Large Language Models. *Drones* 2025, 9, 392. DOI: 10.3390/drones9060392.

- [19] Mahdi, Z.; Abdalhussien, N.; Mahmood, N.; Zaki, R. Detection of Real-Time Distributed Denial-of-Service (DDoS) Attacks on Internet of Things (IoT) Networks Using Machine Learning Algorithms. *Computers, Materials & Continua* 2024, 80, 2139–2159. DOI: 10.32604/cmc.2024.053542.
- [20] Chukwukelu, G.; Essien, A.; Salami, A.I.; Utuk, E. Comparative Analysis of Machine Learning Techniques for DDoS Intrusion Detection in IoT Environments. In *Proceedings of the ICSBT International Conference on Smart Business Technologies*, 2024, pp. 19–27. No DOI available.
- [21] Adewuyi, A.; Oladele, A.A.; Enyiorji, P.U.; Ajayi, O.O.; Tsambatare, T.E.; Oloke, K.; Abijo, I. The Convergence of Cybersecurity, Internet of Things (IoT), and Data Analytics: Safeguarding Smart Ecosystems. *World Journal of Advanced Research and Reviews* 2024, 23, 379–394. DOI: 10.30574/wjarr.2024.23.1.1993.
- [22] Yan, M.; Yang, R.; Zhang, Y.; Yue, L.; Hu, D. A Hierarchical Reinforcement Learning Method for Missile Evasion and Guidance. *Scientific Reports* 2022, 12, 18888. DOI: 10.1038/s41598-022-21756-6.
- [23] Liu, T.; Wang, F.; Li, P. Intelligent Penetration Guidance for Hypersonic Missiles via Reinforcement Learning and Optimal Control. *Preprints* 2025. DOI: 10.20944/preprints202508.0144.v2.
- [24] Vuong, T.C.; Nguyen, C.C.; Pham, V.C.; Le, T.T.H.; Tran, X.N.; Van Luong, T. Effective Intrusion Detection for UAV Communications Using Autoencoder-Based Feature Extraction and Machine Learning Approach. *arXiv* 2024, arXiv:2410.02827. No DOI available (arXiv preprint).
- [25] Ceviz, O.; Sadioglu, P.; Sen, S.; Vassilakis, V.G. A Novel Federated Learning-Based IDS for Enhancing UAVs Privacy and Security. *Internet of Things* 2025, 31, 101592. DOI: 10.1016/j.iot.2025.101592.
- [26] Aldossary, M.; Alzamil, I.; Almutairi, J. Enhanced Intrusion Detection in Drone Networks: A Cross-Layer Convolutional Attention Approach for Drone-to-Drone and Drone-to-Base Station Communications. *Drones* 2025, 9, 46. DOI: 10.3390/drones9010046.
- [27] Guerber, C.; Larrieu, N.; Royer, M. Software Defined Network Based Architecture to Improve Security in a Swarm of Drones. In *Proceedings of ICUAS 2019, Atlanta, GA, USA, 11–14 June 2019*, pp. 51–60. DOI: 10.1109/ICUAS.2019.8797834.
- [28] Altawy, R.; Youssef, A.M. Security, Privacy, and Safety Aspects of Civilian Drones: A Survey. *ACM Transactions on Cyber-Physical Systems* 2016, 1, 1–25. DOI: 10.1145/3001836.
- [29] Sharma, N.; Magarini, M.; Jayakody, D.N.K.; Sharma, V.; Li, J. On-Demand Ultra-Dense Cloud Drone Networks: Opportunities, Challenges and Benefits. *IEEE Communications Magazine* 2018, 56, 85–91. DOI: 10.1109/MCOM.2018.1701001.
- [30] Kim, D.; Song, Y.; Kwon, S.; Kim, H.; Yoo, J.D.; Kim, H.K. UAVCAN Dataset Description. *arXiv* 2022, arXiv:2212.09268. DOI: 10.48550/arXiv.2212.09268.
- [31] Hong, Y.-W.; Yoo, D.-Y. Multiple Intrusion Detection Using Shapley Additive Explanations and a Heterogeneous Ensemble Model in an Unmanned Aerial Vehicle's Controller Area Network. *Applied Sciences* 2024, 14, 5487. DOI: 10.3390/app14135487.
- [32] Majumder, R.; Comert, G.; Werth, D.W.; Gale, A.; Chowdhury, M.; Salek, M.S. Graph-Powered Defense: Controller Area Network Intrusion Detection for Unmanned Aerial Vehicles. *arXiv* 2024, arXiv:2412.02539. DOI: 10.48550/arXiv.2412.02539.
- [33] Wei, Z.; Wang, J.; Zhao, Z.; Zhao, Z.; Shi, K. UpsFed-IDS: U-Shaped Split Federated Intrusion Detection System for Securing UAV Communication in Dynamic Networks. *Ad Hoc Networks* 2026, 183, 104133. DOI: 10.1016/j.adhoc.2025.104133.