LLM-Powered Intrusion Detection Systems for Unmanned Aerial Vehicles

Cong Chi Nguyen, Tuan-Cuong Vuong, Mai Xuan Trang, Huan Vu, Vu-Duc Ngo and Thien Van Luong

***Abstract*—In recent years, many machine learning and deep learning based intrusion detection systems (IDS) for Unmanned Aerial Vehicle (UAV) networks have been proposed. However, most of these models require large labeled datasets to achieve high performance and struggle to adapt to unseen threats. To address these challenges, this work presents the first attempt to apply a large language model (LLM) to UAV intrusion detection systems. In particular, using the real-world UAV-ID dataset (including denial-of-service, replay, evil twin, and false data injection attacks), we transform tabular data of network traffic information into semantic text prompts. These prompts are used to instruction-tune LLMs, creating an LLM-IDS framework. The resulting system shows remarkable data efficiency: achieving a 91.06% F1-score with only 5,000 samples. This performance significantly outperforms machine learning-based baselines such as decision tree and random forest even when those baselines use twice the data. These findings establish a new state-of-the-art benchmark, highlighting the instruction-tuning paradigm for creating data-efficient security systems.**

***Index Terms*—UAV intrusion detection, Large Language Models, machine learning, cyber-physical systems, network security.**

I. Introduction

Unmanned Aerial Vehicles (UAV) are now widely deployed in many fields due to their fast setup and flexible use [1]. They support surveillance, public safety, agriculture, medical services, and military missions for both civil and defense needs [2]. Because the UAV relies on wireless links, they face cyber attacks such as replay, spoofing, and denial-of-service (DoS) attacks [3]. This risk calls for reliable intrusion detection systems (IDS) for UAV networks.

In recent years, numerous machine learning-based datasets and intrusion detection techniques tailored for UAVs have been introduced [4]. To achieve detection and classification of intrusions, such techniques utilize cyber or physical features derived from the available datasets [5]. Herein, Cyber features cover protocol, transmission time, and packet size for network traffic. Physical features include speed, direction, and angles from onboard sensors during flight operations. Most studies rely on simulated datasets or datasets unrelated to UAV, as summarized in [6]. As illustrated in [7], public real UAV datasets remain scarce because collection and labeling are expensive. In this context, an actual dataset for UAV intrusion detection named UAV-ID is the first real dataset combining both cyber and physical features [5]. On UAV-ID, cyber features outperform physical features, and fusion yields additional gains. Additionally, combining both data types would enhance detection performance more effectively than relying on just one of them. On the modeling side, many IDS use machine learning (ML) or deep learning (DL) but still have many limitations [4]. ML models such as decision tree (DT), random forest (RF) are fast but may fail to generalize to new attacks [8]. DL models can learn complex patterns but often need large labeled datasets and high compute, which limits real-time use [9]. Leveraging pre-training on massive corpora, large Language Models (LLM) based approaches can quickly transfer to new tasks and require only a small labeled dataset when using parameter-efficient fine-tuning (PEFT) [10]. To the best of our knowledge, none of the existing works has evaluated the effectiveness of fine-tuning LLM on the UAV-ID dataset. Moreover, a thorough comparison between LLM and traditional ML/DL baselines in the context of UAV intrusion detection has been overlooked in the literature.

Building upon this research background, we propose a more efficient intrusion detection approach for UAV based on the real-world cyber UAV-ID dataset [5]. Specifically, the main contributions of this study are outlined below.

- We propose a novel LLM-based IDS for UAV networks that, for the first time, operates effectively with limited labeled data. In particular, we serialize tabular records into a textual format using a two-part structure to align with the LLM processing capabilities. During the process of converting tabular records into textual format, we apply in-context learning to enhance contextual understanding for improving model performance. Thanks to this design, the proposed approach can improve detection performance compared to traditional methods.
- As mentioned in [6], most existing studies use simulated datasets or datasets unrelated to UAV. We evaluate the effectiveness of the proposed method using the recent real-world UAV-ID dataset. Experimental results show that our approach outperforms baseline methods, including traditional ML/DL models and state-of-the-art (SOTA) techniques such as feature selection and PCA combined with ensemble learning. More importantly, we find that even with a small amount of data, LLM delivers higher effectiveness compared to other methods.

This paper is structured as follows: after this introduction, we provide an overview of related works, followed by our

Chi Cong Nguyen, Tuan-Cuong Vuong, Mai Xuan Trang are with A2I Lab, School of Computing, Phenikaa University, Hanoi, Vietnam (e-mail: {cong.nguyenchi, cuong.vuongtuan, trang.maixuan}@phenikaa-uni.edu.vn).

Thien Van Luong and Huan Vu are with Business AI Lab, Faculty of DS&AI, College of Technology, National Economics University, Hanoi, Vietnam (e-mail: {huanv, thienlv}@neu.edu.vn).

Vu-Duc Ngo is with MobiFone HighTech Center, MobiFone Corp., Hanoi, Vietnam (e-mail: duc.ngo@mobifone.vn).

Corresponding author: Thien Van Luong (e-mail: thienlv@neu.edu.vn)

methodology, an overview of the dataset, experimental results, and finally, conclusions and future research directions.

II. Related Work

This section first surveys existing UAV intrusion datasets, followed by a discussion on intrusion detection methods including both ML and DL approaches. Then, we highlight recent works on LLM for intrusion detection as well as relevant research gaps in this topic.

A. UAV Intrusion Datasets

A range of existing datasets for UAV intrusion detection was summarized in [6]. In these datasets, cyber features typically include information related to Internet protocols such as MAC/IP addresses, packet sizes, while physical features capture flight telemetry such as speed, angles, and directions. Early research in this domain was often constrained by a reliance on general-purpose network security datasets, such as KDD-Cup-99 [11], NSL-KDD [12], UNSW-NB15 [13], InSDN [14], CICIDS-2017 [15], and CSE-CIC-IDS2018 [16]. However, these datasets suffer from a critical limitation: they were not designed for UAV environments and thus lack the specific attack scenarios and physical context inherent to drone operations. Furthermore, many studies utilized simulated data, like the InSDN dataset [14], [17], or were based on privately collected data that were not made public [18]–[20], which impeded reproducibility and comparative analysis across the research community. A primary shortcoming of these earlier datasets was the lack of cyber-physical feature fusion. By focusing on either cyber or physical features in isolation, they hindered the ability of detection models to identify sophisticated attacks that manifest across both domains, such as a cyber intrusion causing anomalous physical behavior.

To address these gaps, a new generation of datasets collected from actual UAV has emerged. Initial advancements included the ALFA and UA datasets [21], which provided real physical data but were limited to a binary "benign" versus "malicious" classification. Similarly, the UAVIDS dataset [22] offered valuable insights into UAV operational modes such as hover, forward but lacked explicit labels for cyber-attacks. The UAV-ID dataset [5], which is the focus of our study, represents a significant leap forward. It was specifically developed to overcome the limitations of its predecessors by providing a realistic, multi-faceted, and publicly available resource. Crucially, it is one of the few datasets that fuses both cyber and physical features and contains labeled data for multiple, distinct cyber-attack types executed on a real UAV. The original study demonstrated that the cyber features were particularly effective for high-performance intrusion detection [5]. Given its realism, labeling, and success in enabling more robust research, we select the cyber portion of the UAV-ID dataset for this work.

B. Intrusion Detection Methodologies

The evolution of UAV datasets was paralleled by the development of diverse intrusion detection methodologies, primarily centered on traditional ML and DL. The literature included classical ML models such as Q-learning [23] and RF [24], as well as advanced DL architectures such as LSTM [25], autoencoders, and CNN [26]. However, a significant body of this research was validated on generic or simulated datasets, which raises questions about their real-world applicability. A stark example of this mismatch is found in [27], where methods designed to detect sophisticated, UAV-specific attacks such as GPS spoofing and jamming were evaluated on the KDD-CUP99 dataset. Even when applied to the UAV-ID dataset, research has so far focused on these traditional models. For instance, the original authors [5] employed SHAP for feature selection, a direction later enhanced by an autoencoder-based feature extraction pipeline [28] and a method combining PCA with ensemble learning [6]. Furthermore, researchers actively tried to overcome the inherent data limitations of these traditional models, employing techniques such as Generative Adversarial Networks (GANs) for data augmentation against specific threats [29] and Genetic Algorithms (GAs) for model optimization [30]. Despite their documented success, these traditional ML/DL approaches share fundamental limitations. They typically require extensive labeled datasets for effective training [31]. Critically, they encounter difficulties in generalizing to novel, zero-day attack patterns without costly and time-consuming retraining. This rigidity is particularly problematic in the dynamic threat landscape of UAV, where new adversarial techniques can emerge rapidly, a challenge highlighted by the performance degradation of TinyML models over time [32].

To overcome these challenges of data dependency and adaptability, LLM emerged as a transformative model in the security field. By conceptualizing sequential data such as network traffic logs or API call sequences, as a language modality, LLM can learn complex patterns, context, and semantic relationships that traditional models often miss [10]. Their efficacy has been demonstrated in various security domains, including SOTA performance in malware detection [33], software vulnerability analysis [34], and anomaly detection in telecom networks [35]. However, the application of LLM to the specific domain of UAV security remains nascent. While some work explored using LLM to analyze UAV network protocols [36], their potential for building a complete, data-driven IDS on real-world UAV traffic has not been fully investigated.

This review reveals a critical and timely research gap: while realistic UAV datasets now exist and LLM have proven powerful for sequential security data, the application of LLM to detect intrusions within a real-world, UAV dataset such as UAV-ID [5] remains an underexplored frontier. This study aims to bridge that gap. We propose a novel framework that transforms structured network traffic from the UAV-ID cyber dataset into a textual format suitable for LLM. By fine-tuning an LLM on this data, we investigate its ability to create a more generalizable and data-efficient intrusion detection system for UAV, as detailed in Section III.

III. Proposed Method

This section details our novel LLM-based intrusion detection method for UAV. The approach fine-tunes the LLM by first

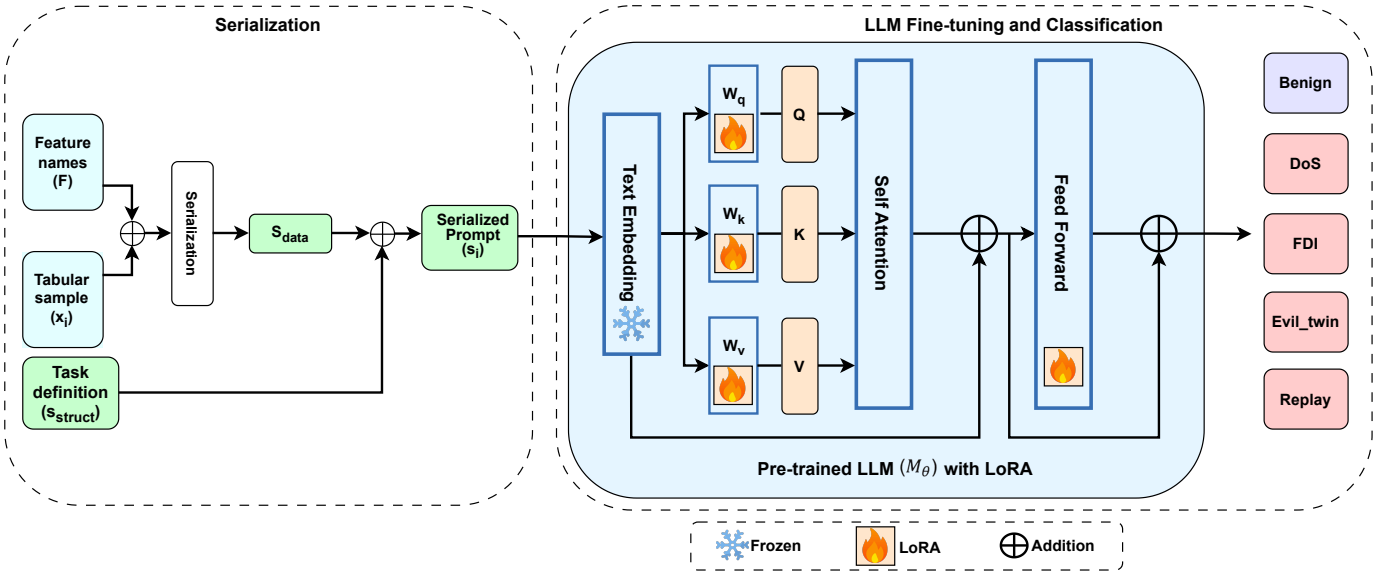

Fig. 1. Overview of the proposed LLM-IDS framework, which consists of two phases: (1) Serialization; (2) LLM fine-tuning and Classification.

converting tabular network data into a template instruction-based format. The complete workflow is illustrated in Fig. 1.

A. Problem Formalization

Suppose we have a tabular intrusion detection dataset $D = \{(\mathbf{x}_i, y_i)\}_{i=1}^{n}$, where n is the total number of network traffic samples. Each $\mathbf{x}_i$ is a d-dimensional feature vector, and $y_i \in C$ is its corresponding class label, where C is the set of predefined classes (e.g., Benign and various attack types). We define the column names, or feature names, as $F = \{f_1, ..., f_d\}$, which are in the form of natural language strings such as "Protocol" and "Packet Size". In this work, we employ an LLM as the classifier, which cannot process raw feature vectors. Consequently, the classification task must be reformulated as a text generation problem. Our objective is to define a serialization function that transforms each tabular pair $(\mathbf{x}_i, F)$ into a natural language prompt s_i, which is then used to fine-tune the LLM so that the fine-tuned LLM can provide the prediction $\hat{y}_i$ as close to the label y_i as possible.

B. Serialization of Tabular Data

To apply an LLM to this tabular classification task, the data must be transformed from its structured format into a natural-language prompt s. This process is known as the serialization of tabular data, as shown in Fig. 1. Our methodology starts with a data preprocessing pipeline to ensure the quality and consistency of the input data. The specifics of this preprocessing pipeline will be detailed for an actual UAV-ID dataset in Section IV. Based on such preprocessed data, we now define a serialization function which converts a feature vector $\mathbf{x}_i$ and its feature names F into a structured instructional prompt.

As illustrated in Fig. 2, for each sample $\mathbf{x}_i$, the complete prompt s_i is formed by concatenating a general instruction header ($s_{instruct}$) with its serialized data content (s_{data}). Formally, $s_i = s_{instruct} \oplus s_{data}$, where an example for this is provided in Fig. 2 that we will explain in the following.

- `text` ($s_{instruct}$): A contextual header serving as the prompt $s_{instruct}$. It contains a clear task definition, specifies the valid output classes C, and optionally includes k in-context learning examples $\{(\mathbf{x}_j, y_j)\}_{j=1}^{k}$ randomly sampled from the training set. The value of k determines the prompting strategy (zero-shot, one-shot, or few-shot).
- `input` (s_{data}): A serialized representation of the specific sample $\mathbf{x}_i$ is generated by concatenating the full descriptive name $f_j \in F$ with its corresponding value x_{ij}, i.e., the j-th element of $\mathbf{x}_i$, for $j = 1, 2, ..., d$. This string concludes with a marker of (`\nAnswer:`) to signal the model to begin its prediction.

Each complete prompt s_i and its ground-truth label y_i are packaged into a conversational JSON format suitable for fine-tuning. This format consists of a two-turn dialogue: a `human` turn containing s_i and an `assistant` turn containing only the correct label y_i.

C. LLM Fine-tuning and Classification

Let M represent the foundation LLM with its pre-trained parameters θ. We employ a Low-Rank Adaptation (LoRA) technique [37] to fine-tune the LLM using the instructional prompt s_i designed in the previous subsection. Particularly, LoRA freezes the original parameters θ and introduces a small set of trainable, low-rank parameters, denoted as ϕ.

As depicted in the LLM fine-tuning phase of Fig. 1, LoRA freezes the pre-trained model weights and injects trainable rank decomposition matrices into each layer of the Transformer architecture. Formally, consider a pre-trained weight matrix $W_0 \in \mathbb{R}^{d\times k}$ within the model. LoRA constrains the weight update ΔW by representing it as the product of two low-rank matrices $B \in \mathbb{R}^{d\times r}$ and $A \in \mathbb{R}^{r\times k}$, where the rank $r \ll \min(d, k)$. Consequently, the forward pass for an input vector $x \in \mathbb{R}^k$ is modified from $h = W_0 x$ to:

$$h = W_0 x + \Delta W x = W_0 x + \frac{\alpha}{r} BAx, \tag{1}$$

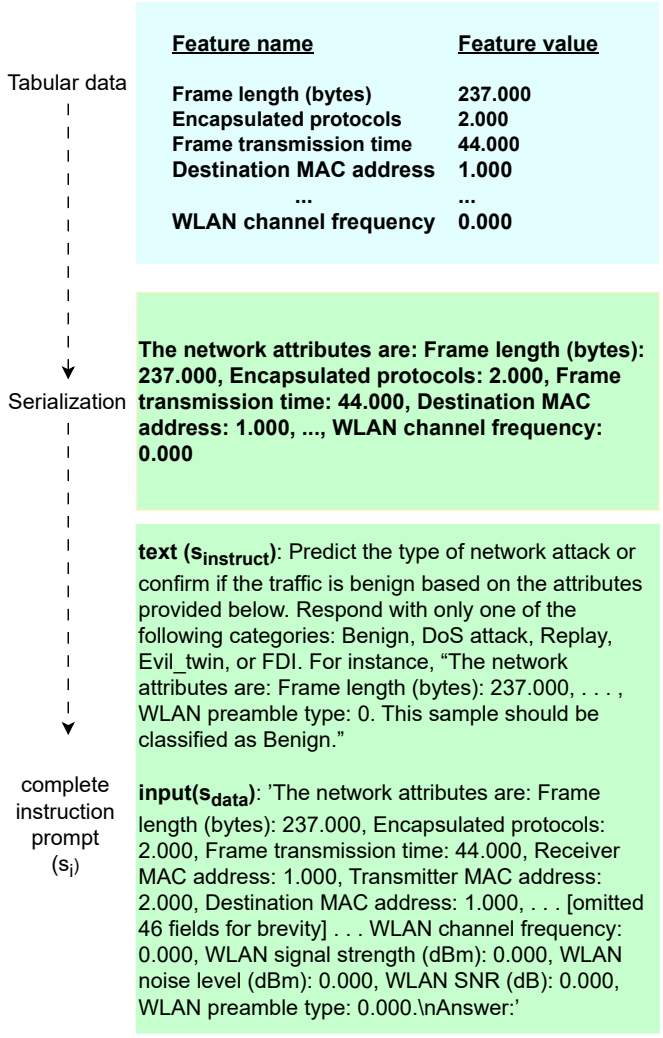

Fig. 2. Illustration of the process for creating a prompt from the preprocessed tabular data. A complete instruction prompt has two primary components: **text** and **input**. The **text** states the task; the **input** holds the serialized features for a sample. This example shows a zero-shot prompt; for one/few-shot, prepend completed examples before `For instance`.

where α is a constant scaling factor used to stabilize training. Regarding initialization, A is initialized with a random Gaussian distribution, while B is initialized to zero, ensuring that $\Delta W = 0$ at the start of training. This guarantees that the fine-tuning process begins exactly with the pre-trained model's behavior. In this framework, W_0 remains frozen. The learnable parameters are exclusively contained within the set $\phi = \{A, B\}$ across all adapted modules. This reduces the number of trainable parameters by several orders of magnitude compared to full fine-tuning.

Let $D_{train} = \{(s_i, y_i)\}_{i=1}^{|D_{train}|}$ be our instruction-formatted training set, where s_i is the complete serialized prompt (derived from $\mathbf{x}_i$, F, and $s_{instruct}$ as defined in Section III-B) and y_i is the corresponding ground-truth label token sequence.

The fine-tuning objective is to find the optimal set of parameters ϕ^* that minimizes the auto-regressive cross-entropy loss across the training set [38]. This objective function, $L(\phi)$, is defined as:

$$L(\phi) = \sum_{(s_i, y_i) \in D_{train}} -\log P(y_i | s_i; \theta, \phi), \tag{2}$$

where $P(y_i|s_i; \theta, \phi)$ is the conditional probability of the model generating the correct token sequence y_i given the input prompt s_i, using the frozen pre-trained parameters θ and the trainable LoRA parameters ϕ [37]. The optimization process seeks to find:

$$\phi^* = \arg\min_{\phi} L(\phi), \tag{3}$$

which is then used for making the prediction in the test set.

During the classification phase in Fig. 1, the fine-tuned model, M_{ϕ^*}, hereafter LLM-IDS framework, acts as a classifier. For an unseen test sample $\mathbf{x}_{test}$, we first generate its corresponding prompt s_{test} using the same serialization function. The prediction is obtained by finding the most probable token sequence:

$$\hat{y} = \arg\max_{y \in C} P(y | s_{test}; \theta, \phi^*), \tag{4}$$

which is then compared against the ground-truth label y_{test} to evaluate the model's performance, as detailed in Section V.

IV. UAV-ID Dataset Description

This section details the UAV dataset used in our study, covering the description of the original dataset and the data preparation process for our experiments.

TABLE 1
THE STATISTICS OF THE ORIGINAL CYBER UAV DATASET

Class	Samples	Original features	Rate (%)
Benign	9425	37	22.3%
DoS	11671	37	27.6%
Replay	12006	37	28.4%
Evil twin	5683	34	13.5%
FDI	3473	34	8.2%

A. Original Dataset

The dataset utilized in this study is the UAV-ID dataset [5]. Unlike most existing datasets which often rely on simulations, it is a real-world dataset. This resource was compiled from experiments under both benign operational conditions and four distinct attack scenarios. These attacks include DoS, Replay, Evil Twin, and FDI. Specifically, the DoS attack is executed by sending a spoofed de-authentication frame to disrupt the UAV's connection with its controller. The Replay attack involves re-transmitting previously captured commands to seize unauthorized control of the UAV. In the Evil Twin scenario, an attacker clones the legitimate Service Set Identifier to establish a rogue access point for data interception and manipulation. Finally, the FDI attack compromises the UAV operational integrity by introducing manipulated vectors that distort control signals and sensor measurements, leading to erratic flight patterns and, consequently, potential mission failure. The original authors of [5] captured data from both

cyber and physical domains. The cyber data consists of network features such as communication protocols, MAC/IP addresses, and port numbers. The physical data includes UAV status indicators such as speed, temperature, and orientation angles. In alignment with findings from the original study that demonstrated the higher efficacy of cyber data for intrusion detection, this work exclusively utilizes the cyber dataset.

B. Data Pre-processing

As noted in Section IV-A, a key challenge of the original dataset [5] is its use of different feature sets for various attack types. As documented in [6], the benign, DoS, and replay scenarios are represented by 37 numerical features, while the evil twin and FDI attacks are described by a set of 34 features. To address this and prepare the data for serialization, we employ a structured pre-processing pipeline comprising three main stages. First, in the data aggregation phase, we consolidate the disparate data files corresponding to the five distinct classes into a unified framework. Subsequently, we apply column standardization to harmonize the feature space by constructing a superset of all attributes found across the subsets. This process results in a unified feature space of 57 dimensions, ensuring that all samples possess a consistent column structure. Several features that do not affect the detection performance can be omitted such as timestamp_c, frame.number, and wlan.bssid. As a consequence, there are 54 remaining helpful features. Finally, the missing value handling stage addresses the gaps created by merging features present in one class but absent in others; these missing values are filled with zeros to ensure input consistency without introducing artificial noise. A complete statistical breakdown of the original dataset, confirming a total of 42,258 samples before cleaning, is provided in Table 1. Prior to its use in our experiments, we conduct a final data cleaning and validation process. This involves identifying and removing any duplicate samples to prevent data leakage and model bias. Furthermore, we address instances of label conflict, where identical feature vectors are associated with different attack labels. These ambiguous samples are also removed. As a consequence, the final dataset used for training and evaluation is ensured to be of high quality and consistency.

To meet the input requirements of each model category, we prepare the data in two formats:

- **Tabular Format**: The data is maintained in its original tabular structure, intended for training and evaluating the traditional ML/DL models.
- **Textual Format**: The tabular data is converted into text sequences using a predefined instruction-following template. This format is specifically designed for fine-tuning and evaluating the LLM.

From these formats, we construct the following subsets for our experiments:

1) For ML/DL Models (Tabular Format):
   - Training Set 1 (Train-5K): A subset of 5,000 samples.
   - Training Set 2 (Train-10K): A larger set of 10,000 samples, which includes all samples from the Train-5K set.
   - Testing Set (Test-10K): A dedicated set of 10,000 samples for evaluation.
   - Full set (Train-Full): We divide the unified dataset into train and test sets with a ratio of 7:3, the train set has 29,580 samples, while the test set has 12,678 samples.
2) For the LLM (Textual Format):
   - Training Set: A set of 5,000 instruction-formatted samples for fine-tuning.
   - Testing Set: A set of 10,000 samples for evaluation.

This experimental setup allows us to investigate several key research questions. We aim to assess the LLM's ability to achieve high performance with only a small fine-tuning dataset (5,000 samples), an amount often considered insufficient for traditional ML/DL models. This premise leads to two direct comparisons: 1) Does the LLM-IDS framework outperform ML/DL models when trained on an identical amount of data? 2) Can traditional ML/DL models, when trained on double the data (10,000 samples), outperform the LLM-IDS framework?

To ensure a fair and direct comparison, the samples within the corresponding training and testing sets across both formats are identical. For instance, the 5,000 samples in the Textual Format training set are the same as those in the Tabular Format Train-5K set. We maintain a consistent label distribution across all subsets, preserving the original dataset's proportions. As a result, our experimental setup enables a balanced and robust evaluation across all models.

V. Experiment and Result

This section validates the effectiveness of our proposed LLM-IDS framework in comparison with baselines. We present the experimental setup, which includes the fine-tuning parameters for our LLM-IDS framework, the selection of relevant baseline methods, and the performance metrics used for evaluation. Subsequently, we conduct a thorough performance analysis, comparing our approach against these baselines. We note that the highest performance for each column of the comparison table is highlighted in **bold**, while the second-highest is underlined. The section concludes with a discussion of the key findings and the implications of applying LLM to the security domain.

A. Implementation Settings

1) Performance Metrics: To evaluate the effectiveness of our proposed model and the baselines, we employ four standard classification metrics: Accuracy, Precision, Recall, and F1-score. These metrics are derived from the number of True Positives (TP), True Negatives (TN), False Positives (FP), and False Negatives (FN) generated from the test set predictions. As illustrated in Fig. A.3, a prediction is True only if the generated output exactly matches the ground-truth label textually. Accuracy measures the proportion of all instances that are correctly classified, given by:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

Precision indicates the ability of the classifier not to label a negative sample as positive. It is the ratio of true positives to the total number of samples predicted as positive:

$$\text{Precision} = \frac{TP}{TP + FP}$$

Recall (or Sensitivity) reflects the classifier's ability to identify all positive samples. It is the ratio of true positives to the total number of actual positive samples:

$$\text{Recall} = \frac{TP}{TP + FN}$$

We select the F1-score as the primary metric for comparison, as it is the harmonic mean of Precision and Recall, providing a single score that balances both concerns, which is particularly useful for imbalanced datasets:

$$\text{F1-score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

For our multi-class classification task, all metrics are reported as macro-averaged percentages over classes.

TABLE 2
OPTIMAL HYPERPARAMETERS SELECTED FOR BASELINE MODELS

Model	Optimized hyperparameters
DT [39]	`max_depth=20` `class_weight='balanced'`

RF [40]	`n_estimators=500`
	`max_depth=20`
	`min_samples_split=5`
	`max_features=0.7`

KNN [41]	`n_neighbors=15` `distance='Manhattan'`
Logistic Reg. [42]	`C=0.1` `solver='saga'`
1D-CNN [43]	`Conv1D(32, kernel_size=1)` `AdamW(lr=1e-4, w_decay=1e-6)`

2) LLM and Fine-tuning Details: Before selecting the base model for fine-tuning, we conduct a preliminary evaluation of several LLM models to establish a performance baseline. Table 3 presents the comparative results of DeepSeek-R1 7B [44], Llama-2-7b-chat-hf [45], Mistral-7B-Instruct [46], GPT-4o-mini [47], gemma-7b-it [48], and Qwen2.5-7B-Instruct [49]. The training is implemented using PyTorch, Transformers, and the PEFT library on a single NVIDIA Quadro RTX 8000 GPU. We employed DeepSpeed[1] with ZeRO Stage 2 and trained with mixed precision (FP16) without quantization.

For fine-tuning, we target all linear modules in the attention and MLP layers, including query, key, value, output, gate, up, and down projections, which correspond to the pre-trained weights W_0 depicted in phase 2 of Fig. 1. We set the LoRA rank to $r = 16$, alpha to $\alpha = 32$, and dropout to 0.05. The input prompts were structured using a custom template as shown in Fig. 2. The model was trained to generate the specific class labels by minimizing the cross-entropy loss [38]. We fix the training duration to 5 epochs with a learning rate of 3e-4 and a cosine scheduler (warmup ratio of 0.01). We use the AdamW optimizer with a weight decay of 1e-5 [50]. The effective batch size is set to 6.

[1]https://www.deepspeed.ai

3) Selected Baselines: Our evaluation strategy is designed to prove the effectiveness of the proposed LLM-IDS framework against a robust baseline suite consisting of ML, DL, and SOTA algorithms, specifically the PCA-based ensemble method (PCA + Ensemble) [6], feature selection-based ensemble approaches (FS + Ensemble) [5], and the Autoencoder-based Feature Extraction method (Autoencode) [28]. To establish a robust set of baselines, we select a diverse range of traditional ML and DL classifiers, implemented via the Scikit-learn[2] and Keras[3] libraries. The resulting optimal hyperparameter configurations for all baseline models are detailed in Table 2. We proceed through three phases:

- First, we conduct a direct comparison where all models are trained on the Train-5K subset and evaluated on the common Test-10K set.
- Next, we benchmark our LLM-IDS framework still trained on Train-5K against baselines trained on the larger Train-10K dataset, with all evaluations consistently performed on the Test-10K set.
- In the final phase, we challenge our 5K-trained model against baselines trained on the complete Train-Full dataset, with all models evaluated on the full test set to demonstrate superior generalization.

B. Experiment Results

TABLE 3
PERFORMANCE COMPARISON OF PRETRAINED MODELS

Model	Precision	Recall	F1-score	Accuracy
DeepSeek-R1 7B	12.14	18.57	9.18	21.17
Llama-2-7b-chat-hf	<u>20.93</u>	<u>21.58</u>	<u>15.94</u>	21.31
Mistral-7B-Instruct	4.47	19.73	7.29	22.02
GPT-4o-mini	**28.51**	**42.99**	**31.68**	**41.00**
Gemma-7b-it	4.59	20.04	7.36	<u>22.35</u>
Qwen2.5-7B-Instruct	8.17	16.24	10.63	19.84

First, to determine the most suitable base model for our fine-tuning framework, we evaluate the performance of various LLMs. The comparative results are presented in Table 3. We maintain a consistent parameter scale of approximately 7B for most open-source models. As shown, GPT-4o-mini achieves the highest performance across all metrics, with an F1-score of 31.68% and Accuracy of 41.00%. However, despite its superior performance, GPT-4o-mini is a closed-source model, limiting its deployment in resource-constrained UAV environments, privacy concerns regarding sensitive network data, and high cost. Among the open-source alternatives, Llama is the most promising candidate. It secures the best performance in open-source metrics, achieving a Precision of 20.93% and an F1-score of 15.94%, outperforming other models such as Mistral, Gemma, and Qwen, which yield around an F1-score of 7-10%. One possible explanation is that Llama's pre-training corpus may contain a richer distribution of data related to network security or technical protocols, enabling it to recognize UAV attack patterns better than its counterparts.

[2]https://scikit-learn.org
[3]https://keras.io

TABLE 4
ABLATION ON THE NUMBER OF DEMONSTRATIONS USED IN INSTRUCTION-DATA GENERATION

Prompt Strategy	Precision	Recall	F1-score	Accuracy
Zero-shot	91.04	90.92	90.98	90.67
One-shot	**91.25**	**90.99**	**91.06**	**90.80**
Two-shot	72.87	71.64	71.93	70.76
Three-shot	72.58	71.19	71.92	70.75

Secondly, we determine the optimal prompt configuration for our fine-tuning process. We ablate the number of demonstration examples used in the prompt template for instruction-tuning data generation: zero-shot, one-shot, two-shot and three-shot. The results of this ablation study are presented in Table 4. As shown in the table, the one-shot strategy yielded the highest performance across all metrics. Additionally, increasing the number of examples to two or three shots results in a degradation in performance. This decline is likely attributed to the context window constraints of the underlying models, such as 4096 tokens for Llama 2 [45]. Based on this empirical evidence, we select the one-shot strategy as our final model for all subsequent comparisons against baseline methods.

As evidenced in Tables 5, 6, and 7, the proposed LLM-IDS consistently yields the highest Precision, Recall, F1-score, and Accuracy across all three experimental scenarios.

TABLE 5
PERFORMANCE COMPARISON BETWEEN OUR ONE-SHOT LLM-IDS AND BASELINES TRAINED ON TRAIN-5K

Models	Precision	Recall	F1-score	Accuracy
CNN	77.69	79.43	77.73	71.34
Logistic Regression	78.76	79.99	79.16	72.13
KNN	80.09	80.87	80.34	73.83
DT	81.91	82.46	81.93	75.99
RF	83.63	84.61	83.76	78.60
FS + Ensemble [5]	81.08	82.04	80.56	75.05
Autoencoder [28]	71.34	71.39	71.37	71.39
PCA + Ensemble [6]	83.30	83.01	80.85	76.22
Our LLM-IDS	**91.25**	**90.99**	**91.06**	**90.80**

Table 5 shows the direct performance benchmark comparison between the proposed LLM-IDS framework and various baselines utilizing the same 5,000-sample training set. It is shown in Table 5 that the proposed instruction-tuned method outperforms all baselines in all performance metrics with a large margin. For instance, our proposed scheme achieves an F1-score of 91.06% and an Accuracy of 90.80%, which are much higher than the corresponding second-best performance metrics recorded by the RF model (83.76% F1-score and 78.60% Accuracy). Consequently, the LLM-IDS framework significantly outperforms the strongest baseline, resulting in a performance gap of 7.30% in F1-score and 12.20% in Accuracy. More interestingly, even established SOTA methods such as PCA + Ensemble and FS + Ensemble are still much worse than the proposed LLM-IDS framework on this dataset. This observation indicates that the proposed LLM-IDS framework is much more effective in detecting intrusions than traditional and ensemble learning models, specifically when training data is limited.

TABLE 6
PERFORMANCE COMPARISON BETWEEN OUR ONE-SHOT LLM-IDS AND BASELINES TRAINED ON TRAIN-10K

Models	Precision	Recall	F1-score	Accuracy
CNN	80.95	81.22	81.13	73.53
Logistic Regression	79.18	80.27	80.26	72.48
KNN	82.72	83.66	82.86	77.39
DT	83.44	84.12	83.65	78.15
RF	84.37	85.33	84.43	79.50
FS + Ensemble [5]	83.11	84.92	83.04	77.77
Autoencoder [28]	71.54	71.55	71.55	71.55
PCA + Ensemble [6]	83.90	84.48	83.44	78.46
Our LLM-IDS	**91.25**	**90.99**	**91.06**	**90.80**

Next, Table 6 compares the LLM-IDS framework (trained on 5,000 samples) against baselines trained on a larger dataset of 10,000 samples. Despite the baselines benefiting from a 100% increase in training data, the LLM-IDS framework maintains higher detection metrics. Specifically, while the RF model improves its F1-score to 84.43% with the additional data, it remains 6.63% lower than the LLM-IDS framework. This comparison indicates that the proposed instruction-tuned framework achieves higher classification performance while requiring fewer labeled training samples than standard supervised learning approaches.

TABLE 7
PERFORMANCE COMPARISON BETWEEN OUR ONE-SHOT LLM-IDS AND BASELINES TRAINED ON TRAIN-FULL

Method	Precision	Recall	F1-score	Accuracy
CNN	77.69	79.43	77.73	71.34
Logistic Regression	71.65	79.28	79.60	76.43
KNN	76.09	81.73	82.55	82.04
DT	77.49	83.30	83.23	83.27
RF	77.05	82.71	83.14	82.78
FS + Ensemble [5]	83.61	84.33	83.97	78.35
Autoencoder [28]	85.99	82.26	84.09	75.34
PCA + Ensemble [6]	86.29	86.94	86.61	82.16
Our LLM-IDS	**91.25**	**90.99**	**91.06**	**90.80**

In the final experimental phase detailed in Table 7, we observe that the LLM-IDS framework achieves metrics that are much higher than the corresponding outcomes of all baselines, even when the baselines benefit from the significantly larger Train-Full dataset. While SOTA methods such as PCA + Ensemble [6] benefit from the full data to reach an F1-score of 86.61%, they still fall short of the LLM-IDS framework, which is 91.06% F1-score. This represents a performance gap of 4.45% in F1-score and 8.64% in accuracy in favor of our method. Additionally, the LLM-IDS framework proves superior to the Autoencoder method [28], exceeding its F1-score of 84.09% by a margin of 6.97%. Likewise, our method surpasses the FS + Ensemble approach [5] by 7.09%.

C. Discussion

Our experimental results demonstrated that the proposed instruction-tuning mechanism allows the LLM-IDS framework to achieve the best intrusion detection performance compared to the SOTA method and traditional ML/DL models. More interestingly, our framework demonstrates exceptional data efficiency, outperforming baselines that utilize 100% of the training data while relying on a 10% subset, as illustrated in Table 6 and Table 7. However, we also acknowledge that using compact models such as 7B parameters introduces a constraint regarding the context window limit, which restricts the number of examples effectively processed, as indicated in Table 4.

VI. Conclusions

We proposed a novel framework for UAV intrusion detection that adapts LLM for tabular security data. Specifically, our LLM-IDS framework is constructed by transforming network traffic into instruction templates to fine-tune the Llama model using the LoRA technique. Our extensive experimental results demonstrate that our proposed LLM-IDS not only outperforms a suite of traditional ML/DL models but also exhibits remarkable data efficiency, achieving SOTA performance with a fraction of the training data. This confirms that the proposed framework overcomes the data dependency limitations of standard ML/DL models, successfully adapting pre-trained capabilities to tabular security tasks even with limited samples. For future work, we plan to investigate efficient context management strategies and compression techniques to mitigate this bottleneck, allowing for more robust few-shot learning on compact models. In light of the context window constraints inherent to smaller architectures, we also plan to extend this research by exploring long-context LLM variants or efficient context compression techniques.

VII. Declarations

Competing interests The authors declare no competing interests.

Ethics approval Not Applicable.

Consent to participate Not Applicable.

Consent to publish Not Applicable.

Funding The work of Thien Van Luong was supported by National Economics University under grant number NEU1-2025.02. The work of Thien Van Luong was also supported by Vietnam National Foundation for Science and Technology Development (NAFOSTED) under grant number 102.05-2025.57.

References

[1] F. Al-Turjman, M. Abujubbeh, A. Malekloo, and L. Mostarda, "UAVs assessment in software-defined IoT networks: An overview," *Computer Communications*, vol. 150, pp. 519–536, 2020.

[2] G. Pajares, "Overview and current status of remote sensing applications based on UAVs," *PE&RS*, vol. 81, pp. 281–330, 04 2015.

[3] J.-P. Condomines, R. Zhang, and N. Larrieu, "Network intrusion detection system for UAV ad-hoc communication: From methodology design to real test validation," *Ad Hoc Networks*, vol. 90, p. 101759, 2019.

[4] V.-D. Ngo, C. Vuong, T. Luong, and H. Tran, "Machine learning-based intrusion detection: feature selection versus feature extraction," *Cluster Computing*, vol. 27, pp. 1–15, 07 2023.

[5] S. Hassler, U. Mughal, and M. Ismail, "Cyber-physical intrusion detection system for unmanned aerial vehicles," *IEEE Trans. Intell. Transp. Syst.*, vol. PP, pp. 1–12, 01 2023.

[6] T. Luong, V.-C. Pham, T. Le, and X.-N. Tran, "Robust intrusion detection for unmanned aerial vehicles: a PCA-based feature extraction and ensemble learning approach," *Cluster Computing*, vol. 28, 05 2025.

[7] H. Naik, J. Yang, D. Das, and M. C. Crofoot, "BuckTales: A multi-UAV dataset for multi-object tracking and re-identification of wild antelopes," in *Adv. Neural Inf. Process. Syst.*, vol. 37, 2024, pp. 81 992–82 009.

[8] E. Biermann, E. Cloete, and L. Venter, "A comparison of intrusion detection systems," *Comput. Secur.*, vol. 20, no. 8, pp. 676–683, 2001.

[9] M. Sharma and C. R. S. Kumar, "Machine learning-based smart surveillance and intrusion detection system for national geographic borders," in *Artificial Intelligence and Technologies*, 2022, pp. 165–176.

[10] H. Xu, S. Wang, N. Li, K. Wang, and Zhao, "Large language models for cyber security: A systematic literature review," *ACM Trans. Softw. Eng. Methodol.*, Sep. 2025.

[11] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the KDD CUP 99 data set," in *CISDA 2009*, 2009, pp. 1–6.

[12] L.Dhanabal and D. S. P. Shantharajah, "A study on NSL-KDD dataset for intrusion detection system based on classification algorithms," *Int. J. Adv. Res. Comput. Commun. Eng.*, 2015.

[13] N. Moustafa and J. Slay, "UNSW-NB15: a comprehensive data set for network intrusion detection systems," in *MilCIS 2015*, 2015, pp. 1–6.

[14] M. S. Elsayed, N.-A. Le-Khac, and A. D. Jurcut, "InSDN: A novel SDN intrusion dataset," *IEEE Access*, vol. 8, pp. 165 263–165 284, 2020.

[15] R. Panigrahi and S. Borah, "A detailed analysis of CICIDS2017 dataset for designing intrusion detection systems," *Int. j. eng. technol.*, vol. 7, pp. 479–482, 01 2018.

[16] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," in *ICISSP*, 2018.

[17] H. Sedjelmaci, S. Senouci, and N. Ansari, "Intrusion detection and ejection framework against lethal attacks in UAV-aided networks: A bayesian game-theoretic methodology," *IEEE T-ITS*, vol. 18, no. 5, pp. 1143–1153, 2017.

[18] S. Khan, C. F. Liew, T. Yairi, and R. McWilliam, "Unsupervised anomaly detection in unmanned aerial vehicles," *Appl. Soft Comput.*, vol. 83, p. 105650, 2019.

[19] R. A. Agyapong, M. Nabil, A.-R. Nuhu, and Rasul, "Efficient detection of GPS spoofing attacks on unmanned aerial vehicles using deep learning," in *IEEE SSCI 2021*, 2021, pp. 01–08.

[20] J. Whelan, A. Almehmadi, and K. El-Khatib, "Artificial intelligence for intrusion detection systems in Unmanned Aerial Vehicles," *Comput. Electr. Eng.*, vol. 99, p. 107784, 2022.

[21] J. Whelan, T. Sangarapillai, O. Minawi, and Almehmadi, "Novelty-based intrusion detection of sensor attacks on Unmanned Aerial Vehicles," in *Q2SWinet 2020*, 2020, pp. 23–28.

[22] A. Alipour-Fanid, M. Dabaghchian, N. Wang, P. Wang, L. Zhao, and K. Zeng, "Machine learning-based delay-aware UAV detection and operation mode identification over encrypted Wi-Fi traffic," *IEEE Trans. Inf. Forensics Secur.*, vol. 15, pp. 2346–2360, 2020.

[23] O. Bouhamed, O. Bouachir, M. Aloqaily, and I. A. Ridhawi, "Lightweight IDS for UAV networks: A periodic deep reinforcement learning-based approach," in *IFIP/IEEE IM*, 2021, pp. 1032–1037.

[24] V. U. Ihekoronye, S. O. Ajakwe, and Kim, "Cyber edge intelligent intrusion detection framework for UAV network based on random forest algorithm," in *13th ICTC 2022*, pp. 1242–1247.

[25] J. Han and W. Pak, "Hierarchical LSTM-based network intrusion detection system using hybrid classification," *Applied Sciences*, vol. 13, no. 5, p. 3089, 2023.

[26] L. Kou, S. Ding, T. Wu, W. Dong, and Y. Yin, "An intrusion detection model for drone communication network in SDN environment," *Drones*, vol. 6, no. 11, p. 342, 2022.

[27] H. Sedjelmaci, S. M. Senouci, and N. Ansari, "A hierarchical detection and response system to enhance security against lethal Cyber-Attacks in UAV networks," *IEEE Trans. Syst. Man. Cybern.*, vol. 48, no. 9, pp. 1594–1606, 2018.

[28] T.-C. Vuong, C. C. Nguyen, V.-C. Pham, X.-N. Tran, and T. V. Luong, "Effective intrusion detection for UAV communications using autoencoder-based feature extraction and machine learning approach," in *NOLTA 2024*.

[29] X. He, Q. Chen, L. Tang, W. Wang, and T. Liu, "CGAN-Based collaborative intrusion detection for uav networks: A blockchain-empowered distributed federated learning approach," *IEEE Internet Things J.*, vol. 10, no. 1, pp. 120–132, 2023.

[30] K. Cengiz, S. Lipsa, R. K. Dash, N. Ivkovic, and M. Konecki, "A novel intrusion detection system based on Artificial Neural Network and Genetic Algorithm with a new dimensionality reduction technique for UAV communication," *IEEE Access*, vol. 12, pp. 4925–4937, 2024.

[31] E. Amer and T. Elboghdadly, "Evaluating machine learning techniques for ICS security: Insights from dataset limitations and classifier performance," in *MIUCC 2024*, pp. 368–373.

[32] T. Zheng, Y. Qiu, Y. Zheng, Q. Wang, and X. Chen, "Enhancing TinyML-Based container escape detectors with systemcall semantic association in UAVs networks," *IEEE Internet Things J.*, vol. 11, no. 12, pp. 21 158–21 169, 2024.

[33] J. Liu, Y. Zhao, Y. Feng, Y. Hu, and X. Ma, "SeMalBERT: Semantic-based malware detection with bidirectional encoder representations from transformers," *J. Inf. Secur. Appl.*, vol. 80, p. 103690, 2024.

[34] M. Omar and S. Shiaeles, "VulDetect: a novel technique for detecting software vulnerabilities using language models," in *IEEE CSR 2023*, pp. 105–110.

[35] A. Maatouk, N. Piovesan, F. Ayed, A. D. Domenico, and M. Debbah, "Large language models for telecom: Forthcoming impact on the industry," *IEEE Commun. Mag.*, vol. 63, pp. 62–68, 2023.

[36] B. Piggott, S. Patil, G. Feng, I. Odat, and Mukherjee, "Net-GPT: A LLM-empowered man-in-the-middle chatbot for unmanned aerial vehicle," in *SEC 2024*, pp. 287–293.

[37] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, W. Chen *et al.*, "Lora: Low-rank adaptation of large language models." *ICLR*, vol. 1, no. 2, p. 3, 2022.

[38] C. E. Shannon, "A mathematical theory of communication," *BSTJ*, vol. 27, no. 3, pp. 379–423, 1948.

[39] W.-Y. Loh, "Classification and regression trees," *Wiley Interdiscip. Rev.: Data Min. Knowl. Discov.*, vol. 1, pp. 14 – 23, 01 2011.

[40] L. Breiman, "Random forests," *Mach. Learn.*, vol. 45, no. 1, 2001.

[41] T. Cover and P. Hart, "Nearest neighbor pattern classification," *IEEE Trans. Inf. Theory.*, vol. 13, no. 1, pp. 21–27, 1967.

[42] D. R. Cox, "The regression analysis of binary sequences," *J. R. Stat. Soc., B: Stat. Methodol.*, vol. 20, no. 2, pp. 215–232, 1958.

[43] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE.*, vol. 86, no. 11, pp. 2278–2324, 1998.

[44] DeepSeek-AI, D. Guo, D. Yang, H. Zhang, and J. Song, "Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning," 2025. [Online]. Available: https://arxiv.org/abs/2501.12948

[45] H. Touvron, L. Martin, K. Stone, P. Albert, and A. Almahairi, "Llama 2: Open foundation and fine-tuned chat models," 2023. [Online]. Available: https://arxiv.org/abs/2307.09288

[46] A. Q. Jiang, A. Sablayrolles, A. Mensch, and C. Bamford, "Mistral 7b," 2023. [Online]. Available: https://arxiv.org/abs/2310.06825

[47] OpenAI, J. Achiam, S. Adler, S. Agarwal, and et.al, "GPT-4 technical report," 2024.

[48] G. Team, C. Riquelme, J. Puigcerver, B. Mustafa, M. Neumann *et al.*, "Gemma 2: Improving open language models at a practical size," *arXiv:2408.00118*, 2024.

[49] A. Yang, B. Yang, B. Hui, B. Zheng, B. Yu, C. Zhou *et al.*, "Qwen2 technical report," *arXiv preprint arXiv:2407.10671*, 2024.

[50] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," 2019. [Online]. Available: https://arxiv.org/abs/1711.05101

APPENDIX A
SAMPLE INSTRUCTION TEMPLATES

text: Predict the type of network attack or confirm if the traffic is benign based on the attributes provided below. Respond with only one of the following categories: 'benign', 'DoS attack', 'Replay', 'evil_twin', or 'FDI'.

For instance: "The network attributes are: Captured frame raw data: 36.000000, Frame data length: 164.000000, Frame length (bytes): 226.000000, Encapsulated protocols: 3.000000, Destination IP address: 1.000000, IP header flags: 0.000000, IP header length (bytes): 20.000000, IP header ID field: 287.000000, IP packet total length: 192.000000, IP data protocol: 17.000000, Source IP address: 1.000000, Packet TTL: 255.000000, Logical Link Control (LLC) type: 2048.000000, Antenna signal strength: 0.000000, Radiotap CCK flag: 0.000000, Radiotap OFDM flag: 0.000000, Radiotap header length: 0.000000, Signal quality: 0.000000, TCP acknowledgment number: 0.000000, TCP destination port: 0.000000, TCP flags: 0.000000, TCP header length (bytes): 0.000000, TCP options: 0.000000, TCP sequence number: 0.000000, TCP source port: 0.000000, TCP window size (bytes): 0.000000, Time since last packet: 3.416639, UDP destination port: 8890.000000, UDP packet length: 172.000000, UDP source port: 8889.000000, CCMP extended IV: 0.000000, Destination MAC address: 0.000000, wlan.duration: 44.000000, Frame control subtype: 8.000000, Frame control type: 2.000000, FCS value: 0.000000, FCS status: 0.000000, WLAN frame flags: 0.000000, WLAN fragment number: 0.000000, QoS parameters: 0.000000, QoS acknowledgment type: 0.000000, QoS priority level: 0.000000, Receiver MAC address: 0.000000, Source MAC address: 1.000000, Frame sequence number: 292.000000, Transmitter MAC address: 1.000000, WEP key: 0.000000, WLAN noise level (dBm): 0.000000, WLAN SNR (dB): 0.000000, WLAN channel: 0.000000, WLAN data rate: 0.000000, WLAN channel frequency: 0.000000, WLAN preamble type: 0.000000, WLAN signal strength (dBm): 0.000000. This sample should be classified as 'benign'."

input: 'The network attributes are: Captured frame raw data: 0.000000, Frame data length: 0.000000, Frame length (bytes): 26.000000, Encapsulated protocols: 0.000000, Destination IP address: 0.000000, IP header flags: 0.000000, IP header length (bytes): 0.000000, IP header ID field: 0.000000, IP packet total length: 0.000000, IP data protocol: 0.000000, Source IP address: 0.000000, Packet TTL: 0.000000, Logical Link Control (LLC) type: 0.000000, Antenna signal strength: 0.000000, Radiotap CCK flag: 0.000000, Radiotap OFDM flag: 0.000000, Radiotap header length: 0.000000, Signal quality: 0.000000, TCP acknowledgment number: 0.000000, TCP destination port: 0.000000, TCP flags: 0.000000, TCP header length (bytes): 0.000000, TCP options: 0.000000, TCP sequence number: 0.000000, TCP source port: 0.000000, TCP window size (bytes): 0.000000, Time since last packet: 0.002212, UDP destination port: 0.000000, UDP packet length: 0.000000, UDP source port: 0.000000, CCMP extended IV: 0.000000, Destination MAC address: 1.000000, wlan.duration: 314.000000, Frame control subtype: 12.000000, Frame control type: 0.000000, FCS value: 0.000000, FCS status: 0.000000, WLAN frame flags: 0.000000, WLAN fragment number: 0.000000, QoS parameters: 0.000000, QoS acknowledgment type: 0.000000, QoS priority level: 0.000000, Receiver MAC address: 1.000000, Source MAC address: 0.000000, Frame sequence number: 2565.000000, Transmitter MAC address: 0.000000, WEP key: 0.000000, WLAN noise level (dBm): 0.000000, WLAN SNR (dB): 0.000000, WLAN channel: 0.000000, WLAN data rate: 0.000000, WLAN channel frequency: 0.000000, WLAN preamble type: 0.000000, WLAN signal strength (dBm): 0.000000.'\nAnswer:'

Fig. A.1. Illustration of the one-shot prompt structure. The text block contains the task instruction and one labeled example (class: *benign*) to guide the model.

text: Predict the type of network attack or confirm if the traffic is benign based on the attributes provided below. Respond with only one of the following categories: 'benign', 'DoS attack', 'Replay', 'evil_twin', or 'FDI'.

For instance 1: "The network attributes are: Captured frame raw data: 36.000000, Frame data length: 164.000000, Frame length (bytes): 226.000000, Encapsulated protocols: 3.000000, Destination IP address: 1.000000, IP header flags: 0.000000, IP header length (bytes): 20.000000, IP header ID field: 287.000000, IP packet total length: 192.000000, IP data protocol: 17.000000, Source IP address: 1.000000, Packet TTL: 255.000000, Logical Link Control (LLC) type: 2048.000000, Antenna signal strength: 0.000000, Radiotap CCK flag: 0.000000, Radiotap OFDM flag: 0.000000, Radiotap header length: 0.000000, Signal quality: 0.000000, TCP acknowledgment number: 0.000000, TCP destination port: 0.000000, TCP flags: 0.000000, TCP header length (bytes): 0.000000, TCP options: 0.000000, TCP sequence number: 0.000000, TCP source port: 0.000000, TCP window size (bytes): 0.000000, Time since last packet: 3.416639, UDP destination port: 8890.000000, UDP packet length: 172.000000, UDP source port: 8889.000000, CCMP extended IV: 0.000000, Destination MAC address: 0.000000, wlan.duration: 44.000000, Frame control subtype: 8.000000, Frame control type: 2.000000, FCS value: 0.000000, FCS status: 0.000000, WLAN frame flags: 0.000000, WLAN fragment number: 0.000000, QoS parameters: 0.000000, QoS acknowledgment type: 0.000000, QoS priority level: 0.000000, Receiver MAC address: 0.000000, Source MAC address: 1.000000, Frame sequence number: 292.000000, Transmitter MAC address: 1.000000, WEP key: 0.000000, WLAN noise level (dBm): 0.000000, WLAN SNR (dB): 0.000000, WLAN channel: 0.000000, WLAN data rate: 0.000000, WLAN channel frequency: 0.000000, WLAN preamble type: 0.000000, WLAN signal strength (dBm): 0.000000. This sample should be classified as 'benign'."

For instance 2: "The network attributes are: Captured frame raw data: 0.000000, Frame data length: 0.000000, Frame length (bytes): 26.000000, Encapsulated protocols: 0.000000, Destination IP address: 0.000000, IP header flags: 0.000000, IP header length (bytes): 0.000000, IP header ID field: 0.000000, IP packet total length: 0.000000, IP data protocol: 0.000000, Source IP address: 0.000000, Packet TTL: 0.000000, Logical Link Control (LLC) type: 0.000000, Antenna signal strength: 0.000000, Radiotap CCK flag: 0.000000, Radiotap OFDM flag: 0.000000, Radiotap header length: 0.000000, Signal quality: 0.000000, TCP acknowledgment number: 0.000000, TCP destination port: 0.000000, TCP flags: 0.000000, TCP header length (bytes): 0.000000, TCP options: 0.000000, TCP sequence number: 0.000000, TCP source port: 0.000000, TCP window size (bytes): 0.000000, Time since last packet: 0.007366, UDP destination port: 0.000000, UDP packet length: 0.000000, UDP source port: 0.000000, CCMP extended IV: 0.000000, Destination MAC address: 0.000000, wlan.duration: 314.000000, Frame control subtype: 12.000000, Frame control type: 0.000000, FCS value: 0.000000, FCS status: 0.000000, WLAN frame flags: 0.000000, WLAN fragment number: 0.000000, QoS parameters: 0.000000, QoS acknowledgment type: 0.000000, QoS priority level: 0.000000, Receiver MAC address: 0.000000, Source MAC address: 1.000000, Frame sequence number: 3844.000000, Transmitter MAC address: 1.000000, WEP key: 0.000000, WLAN noise level (dBm): 0.000000, WLAN SNR (dB): 0.000000, WLAN channel: 0.000000, WLAN data rate: 0.000000, WLAN channel frequency: 0.000000, WLAN preamble type: 0.000000, WLAN signal strength (dBm): 0.000000. This sample should be classified as 'DoS attack'."

input: 'The network attributes are: Captured frame raw data: 0.000000, Frame data length: 0.000000, Frame length (bytes): 26.000000, Encapsulated protocols: 0.000000, Destination IP address: 0.000000, IP header flags: 0.000000, IP header length (bytes): 0.000000, IP header ID field: 0.000000, IP packet total length: 0.000000, IP data protocol: 0.000000, Source IP address: 0.000000, Packet TTL: 0.000000, Logical Link Control (LLC) type: 0.000000, Antenna signal strength: 0.000000, Radiotap CCK flag: 0.000000, Radiotap OFDM flag: 0.000000, Radiotap header length: 0.000000, Signal quality: 0.000000, TCP acknowledgment number: 0.000000, TCP destination port: 0.000000, TCP flags: 0.000000, TCP header length (bytes): 0.000000, TCP options: 0.000000, TCP sequence number: 0.000000, TCP source port: 0.000000, TCP window size (bytes): 0.000000, Time since last packet: 0.002212, UDP destination port: 0.000000, UDP packet length: 0.000000, UDP source port: 0.000000, CCMP extended IV: 0.000000, Destination MAC address: 1.000000, wlan.duration: 314.000000, Frame control subtype: 12.000000, Frame control type: 0.000000, FCS value: 0.000000, FCS status: 0.000000, WLAN frame flags: 0.000000, WLAN fragment number: 0.000000, QoS parameters: 0.000000, QoS acknowledgment type: 0.000000, QoS priority level: 0.000000, Receiver MAC address: 1.000000, Source MAC address: 0.000000, Frame sequence number: 2565.000000, Transmitter MAC address: 0.000000, WEP key: 0.000000, WLAN noise level (dBm): 0.000000, WLAN SNR (dB): 0.000000, WLAN channel: 0.000000, WLAN data rate: 0.000000, WLAN channel frequency: 0.000000, WLAN preamble type: 0.000000, WLAN signal strength (dBm): 0.000000.'\nAnswer:'

Fig. A.2. Illustration of the two-shot prompt structure. The text block contains the task instruction and two labeled examples (classes: *benign* and *DoS attack*) to guide the model.

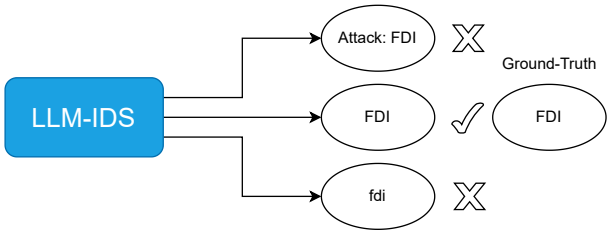

Fig. A.3. A prediction is considered correct only if it is textually identical to the ground-truth label.