Artificial Intelligence Review _####################_
https://doi.org/10.1007/s10462-025-11432-2

Evaluating large language models effectiveness for flow-based intrusion detection: a comparative study with ML and DL baselines

Lorena Mehavilla[1] · **María Rodríguez**[1] · **José García**[1] · **Álvaro Alesanco**[1]

Received: 19 May 2025 / Accepted: 19 October 2025

Abstract
This paper presents the first systematic benchmark evaluating Large Language Models (LLMs), specifically GPT-2, GPT-Neo-125M, and LLaMA-3.2-1B, as standalone classifiers for intrusion detection, covering both binary and multiclass classification tasks, using structured Zeek logs derived from the CIC IoT 2023 dataset. We compare their performance against established and widely used Machine Learning (XGBoost, Random Forest, Decision Tree) and Deep Learning models (MLP, GRU, LeNet-5) across key evaluation metrics: detection effectiveness (precision, recall and F1-score), inference speed, and resource consumption. All models are consistently trained and rigorously evaluated on the CIC IoT 2023 dataset, ensuring fair, reproducible, and transparent comparisons. Our findings indicate that while LLMs achieve strong F1-score exceeding 95%, and do not fully utilize available GPU resources, they still do not outperform top-performing ML models. Notably XGBoost achieves a higher F1-score of 96.96%, using only 4% of the available CPU. These results emphasize the practical trade-offs between detection capability, inference efficiency, and hardware requirements when applying LLMs in flow-based IDS contexts, particularly in resource-constrained environments such as IoT or edge deployments.

Keywords Intrusion detection · Large language models · Internet of things · Machine learning · Deep learning

Lorena Mehavilla, María Rodríguez, José García and Álvaro Alesanco have contributed equally to this work.

✉ Lorena Mehavilla
lmehavilla@unizar.es

María Rodríguez
maria.rodriguezg@unizar.es

José García
jogarmo@unizar.es

Álvaro Alesanco
alesanco@unizar.es

1 Aragón Institute of Engineering Research (I3A), University of Zaragoza, C/ Mariano Esquillor Gómez, 50018 Zaragoza, Spain

1 Introduction

The frequency and sophistication of cyberattacks have surged in recent years. According to Exploding Topics, a cyberattack occurs approximately every 11 seconds (Martin 2024), and Cisco estimates that information-stealing malware alone has cost victims over $3 billion in 2024 (Cisco 2024). With cybercrime projected to cost the global economy $10.5 trillion annually by 2025 (Fox 2023), there is an urgent need for robust and scalable solutions in threat detection and response. Among the most promising advancements in artificial intelligence is the emergence of Large Language Models (LLMs), which have demonstrated transformative capabilities in natural language processing, code generation, and reasoning. As such, researchers have begun exploring their applicability beyond text, including cybersecurity tasks such as intrusion detection.

While many studies have investigated traditional Machine Learning (ML) and Deep Learning (DL) techniques for intrusion detection systems (IDS) (Saran and Kesswani 2023; Liu et al. 2024), few have extended this analysis to modern Generative AI models, particularly Large Language Models (LLMs) (Azzam et al. 2023; Bhandari et al. 2022; Wangsa et al. 2024; Diaf et al. 2024). Existing research on LLMs in cybersecurity has largely emphasized prompt-based interaction or post-detection reasoning, with limited attention to their use as direct classifiers of structured flow data like Zeek logs. To our knowledge, this is the first study to rigorously evaluate fine-tuned LLMs as standalone classifiers on Zeek-based intrusion detection tasks, benchmarked against traditional ML and DL models. This addresses a crucial gap in the literature: the classification capabilities of LLMs remain underexplored, especially when compared to well-established ML baselines. In real-world IDS deployments, LLMs must first classify network flows accurately before offering human-readable explanations. If this foundational task fails, any downstream capabilities, such as summarization or interpretability, risk becoming irrelevant or even misleading. Therefore, ensuring strong foundational classification performance is a necessary step before leveraging their advanced interpretability features. Despite their impressive general language understanding and reasoning capabilities, LLMs may not be inherently optimized for structured classification tasks such as network flow analysis, where traditional models often achieve superior performance.

In this study, we aim to fill this gap by evaluating the capability of LLMs, specifically GPT-2, GPT-Neo-125M, and LLaMA-3.2-1B, as direct classifiers of encrypted network flows for intrusion detection. We compare their performance with several widely used ML and DL models, including XGBoost, Random Forest, Decision Tree, MLP, GRU, and LeNet-5. Our experiments are conducted on the CIC IoT 2023 dataset (Neto et al. 2023), a realistic and diverse IoT attack dataset that includes both benign and malicious network behaviours. We provide a fair and balanced comparison across multiple dimensions: detection effectiveness, inference speed, and resource consumption.

This work makes the following key contributions:

- First, we present a systematic comparison between LLMs, Transformers and traditional ML/DL models in the context of intrusion detection using structured Zeek flows.
- Second, we evaluate performance not only in terms of detection effectiveness but also through inference time and compute resource usage, making our findings more applicable to real-world deployment.

- Third, we conduct a set of supporting analyses using t-SNE and SHAP (Naif Alatawi 2025; Khediri et al. 2024) to enhance understanding of the dataset structure and the decision-making processes of baseline ML models. Although these techniques do not directly provide explainability for LLMs, they establish a foundation for future research on explainable LLM-based IDS systems.

The remainder of the paper is structured as follows: Sect. 2 reviews related work; Sect. 3 describes the dataset and modelling approach; Sect. 4 presents experimental results and performance comparisons; Sect. 5 provides explainability analysis for ML models; and Sect. 6 concludes with key takeaways and directions for future research.

2 Literature review

2.1 Large language models in cybersecurity

Generative AI, particularly Large Language Models (LLMs), has recently emerged as a tool for automating both offensive and defensive cybersecurity tasks, including attack simulation, threat intelligence extraction, and rule generation. For example, (Babaey and Ravindran 2025) developed a framework that leverages GPT-4o to generate realistic SQL injection (SQLi) payloads and create Web Application Firewall (WAF) rules. However, the LLM in this setup is used exclusively for content generation, while the classification and clustering are handled by a separate ML model. Similarly, (Fieblinger et al. 2024) explored the use of LLMs like LLaMA and Mistral for extracting Cyber Threat Intelligence (CTI) and building knowledge graphs. Yet, their work focused only on malicious CTI reports, omitting the analysis of benign behaviours necessary for whitelisting or trust modelling.

While these studies show the growing interest in LLMs for cybersecurity, they fall short in several key areas: they rarely assess classification performance in IDS tasks, often rely on handcrafted prompts or biased features (e.g., IPs, ports), and typically omit comparisons with ML or DL baselines, limiting both rigor and generalizability.

2.2 LLMs for intrusion detection systems (IDS)

Several recent studies have applied LLMs directly to intrusion detection tasks, yielding promising results but also exposing critical limitations. For example, (Hassanin et al. 2025) evaluated a pre-trained LLM for cyber threat detection in satellite networks using datasets like UNSW-NB15 and TON-IoT. Despite reporting near-perfect accuracy, the model relied heavily on IP addresses and timestamps, likely introducing bias and overfitting. (Gutiérrez-Galeano et al. 2025) presented a cyberattack detection approach using fine-tuned pre-trained T5 model. While their method achieved near-perfect detection performance across the CIC IDS 2017, CSE CIC IDS 2018 and BCCC CIC IDS 2017 datasets, it still relied on bias-inducing features like destination ports. Moreover, the study did not include comparisons with other LLMs that share similar model sizes, training procedures, or computational footprints, limiting the robustness of its evaluation.

Similarly, Zhang et al. (2023) integrated GPT-2 with cybersecurity tooling such as Suricata and the ELK stack for real-time threat detection. However, the fine-tuning prompts included timestamps, IPs and ports, again risking overfitting. Mahmoodi and Jameii (2024) proposed a fine-tuned LLaMA2 model for DDoS detection using the CIC IDS 2017 dataset. Despite high reported metrics (>98%), the study lacked comparison with ML baselines and did not specify the feature selection process. Zhang et al. (2024) evaluated two pre-trained LLMs (GPT-3.5 and GPT-4) in three distinct in-context learning methods. The results showed that GPT-4 could achieve an F1-score above 95% with only 10 in-context learning examples. Nonetheless, the study relied on the LLM to select the most important features, which included the IP addresses, previously mentioned for introducing bias. As with earlier works, this study did not include comparisons with ML and DL baselines, and the evaluation was limited to models from the GPT family.

Some comparative studies have begun to evaluate LLMs against traditional techniques. For instance, Guastalla et al. (2024) compared GPT-3.5 and GPT-4 against a simple MLP, finding that the LLMs significantly outperformed the baseline. Nonetheless, the MLP's poor performance ($\leq$ 75% F1) weakened the comparative value. A more rigorous comparison in Houssel et al. (2024) evaluated GPT-4 and LLaMA against Random Forests and Decision Trees using NetFlow datasets, yet classification accuracy hovered around 50%, suggesting suboptimal modelling or dataset issues. Nwafor et al. (2024) fine-tuned GPT-3.5 base model and developed an embedding similarity technique from GPT-3.5 using both balance and imbalance versions of the CIC IoT 2023 dataset. The study evaluated the performance on binary and multiclass classification tasks. While the embedding similarity technique achieved an F1-score of 98.8%, the baseline ML models (Random Forest and Gradient Boosting) outperformed it. Although the study correctly selected non-bias-inducing features, it did not report all the obtained results, limiting the transparency and reproducibility of the evaluation. Bui et al. (2024) investigated whether LLM can detect malicious pattern directly within raw packets. The study explored three approaches: (i) an end-to-end method where the LLM provides the full solution in a single step with few-shot prompting; (ii) a RAG strategy where the LLM has access to a database of exemplary ray payloads of previous attacks; and (iii) a task-specific fine-tuned LLM designed to classify attack types. This third approach achieves over a 95% accuracy, outperforming state-of-the-art ML baselines. Nevertheless, the accuracy was the only performance metric reported; more informative metrics such as F1-score, precision, recall and confusion matrices should have been included to enable a more robust evaluation and comparison.

In general, most existing LLM-based IDS studies suffer from one or more of the following limitations:

- Use of bias-prone features (e.g., IPs, ports, timestamps).
- Lack of standard ML or DL baselines for comparison.
- Overreliance on accuracy without deeper performance metrics.
- Limited reproducibility or insufficient model detail.

2.3 Transformers-based models for IDS

Transformers, a main component of LLMs, have also been explored as based models for IDS, leveraging their ability to capture complex dependencies in sequential data. For instance, Zhou and Wang (2024) developed a network intrusion system using a transformer model with a softmax classifier as its detection model. To evaluate its performance, NSL-

KDD and UNSW-NB15 datasets were employed. However, the method relied on bias-prone features and was only compared against OSVM, BiGRU-SL, and Res-TranBiLSTM. Similarly, Nguyen and Watabe (2022) employed a BERT-based model that treated network flows as sequences of words, achieving strong binary classification results on the CIDDS datasets. Nevertheless, this work also used potentially bias-inducing features and did not extend to multiclass classification.

In a more robust attempt, Adjewa et al. (2024) fine-tuned BERT for multiclass IDS tasks on the Edge IIoT dataset while avoiding timestamp-based features. They even demonstrated deployment on edge hardware like Raspberry Pi 4, showing accuracy above 96%. Nonetheless, their evaluation focused solely on accuracy, omitting precision, recall, and F1-score, which are more informative in imbalanced scenarios. Wang et al. (2023) proposed an intrusion detection model combining ResNet, Transformer and BiLSTM architectures to capture both spatial and temporal characteristics of network traffic. The model was evaluated on the NSL-KDD, CIC IDS 2017, and MQTT datasets. Nonetheless, the comparative studies lacked key performance metrics, making proper benchmarking difficult. Li et al. (2022) proposed for encrypted malicious traffic classification a hybrid deep learning model using a softmax classifier fed by two subnetworks: T-net (Transformer) and C-net (Convolutional Neural Network). This model tried to avoid statistic features that can introduce biases and for evaluation it used MTA, STRA and USTC datasets. However, the comparison was limited to models such as TCMal-WT, M1-CNN, M2-CNN, MalDIST, and PERT, with limited insight into broader baselines.

Beyond individual methods, systematics reviews have begun to systematize this research landscape. For instance, Kheddar (2025) conducted a comprehensive survey of Transformer and LLM-based approaches for cyber-threat detection, including IDS. Their analysis included architectures ranging from attention-based models and BERT/GPT variants to CNN/LSTM-Transformer hybrids and emerging designs. The survey also highlighted diverse application domains, while identifying persistent challenges such as interpretability, scalability, and adaptability to evolving threats. This broader perspective underscores both the versatility of Transformers in cybersecurity and the need for more standardized evaluation practices in IDS research.

2.4 Machine learning and deep learning for IDS

In contrast, traditional ML and DL approaches for intrusion detection are well established and extensively validated. For instance, Rodríguez et al. (2022) evaluated ML classifiers like Decision Trees, SVM, and XGBoost on the CIC IDS 2017 dataset, demonstrating strong performance with low computational requirements. García and Entrena (2024) studied feature selection and classifier combinations across several IoT-focused datasets, achieving F1-scores above 0.99 while reducing feature sets by over 60%.

More complex DL models have also been explored. Saran and Kesswani (2023) proposed an ML-based IDS for MQTT traffic using classifiers like KNN, Naïve Bayes, Random Forest, and SGD. Udurume et al. (2024) compared CNN and CNN-BiLSTM architectures against ML models on the UNSW-NB15 and NSL-KDD datasets. Although DL models achieved competitive results (up to 97% F1-score), details on feature selection and model configurations were often lacking, limiting reproducibility. A similar issue was observed in Ahmad et al. (2022), which analysed several datasets using both individual (MLP, CNN, LSTM and TCN) and hybrid (autoencoder + TCN/LSTM/BRNN/BLSTM and CNN + LSTM) Deep Learning classifiers, achieving impressive results but failing to specify the selected features for each dataset.

In addition to individual evaluations, some works have provided systematic benchmarks of ML and DL approaches for IDS. Ahmad et al. (2021) conducted a systematic review of journal articles focusing on ML and DL based network intrusion detection systems (NIDS). Each study was analysed in terms of proposed methodology, strength, weakness, evaluation metrics and the datasets used in both binary and multiclass scenarios. A et al. (2024) aimed to analyse an IA-based NIDS design and compare the performance achieved by different ML models (Decision Tree, Random Forest and Support Vector Machines) and DL models (CNN and RNN) across three merged datasets (KDD Cup 99, NLS-KDD and CIC IDS 2017) along with real-time traffic captures.

These studies reinforce that classical ML models often provide a strong balance between effectiveness, simplicity, and efficiency, especially in structured datasets like those derived from network traffic.

2.5 Summary and gap

While LLMs show emerging potential for intrusion detection, prior studies rarely provide rigorous, side-by-side comparisons with traditional ML or DL models under consistent experimental conditions. This paper fills that gap by evaluating LLMs not as post-hoc reasoning tools, but as direct classifiers, using standardized datasets and baselines for a fair and reproducible comparison. Many also rely on features that can bias results or use insufficient performance metrics. Meanwhile, ML and DL techniques have been thoroughly benchmarked, are lightweight, and often achieve excellent performance with minimal resources.

Transformer-based models, which serve as the architectural foundation for LLMs, have also been explored independently for IDS tasks. While they offer strong sequence modelling capabilities and have shown promise in capturing temporal and spatial patterns in network traffic, many studies suffer from similar limitations as LLM-based approaches. These include reliance on bias-prone features, lack of detailed baseline comparisons, and limited use of robust evaluation metrics.

To address this gap, our study conducts a systematic, side-by-side evaluation of LLMs, Machine Learning, and Deep Learning models for both binary and multiclass intrusion detection. While two of the evaluated LLMs are built on Transformer architectures, we do not separately assess standalone Transformer-based models. Instead, our focus is on evaluating LLMs as end-to-end classifiers in comparison with traditional techniques. Using the CIC IoT 2023 dataset, we assess not only detection effectiveness but also inference time, resource consumption, and explainability (on ML models) to provide a holistic analysis of these approaches.

Table 1 presents a summary of prior LLM-based IDS studies.

Table 1 Summary of prior LLM-based IDS studies

References	Study	LLM Used	Dataset(s)	Bias-prone features (e.g., IPs, Ports)	Evaluation type	ML/DL baseline
Hassanin et al. (2025)	Hassanin et al. (2025)	Custom LLM (PLLM-CS)	UNSW-NB15, TON-IoT	Yes (IPs, timestamps)	Binary class.	RF, ETC, XGB, CNN, LSTM, BiLSTM, FNN, GRU
Gutiérrez-Galeano et al. (2025)	Gutierrez at al. (2025)	T5	CIC IDS 2017, CSE CIC IDS 2018, BCC CIC IDS 2017	Yes (dst port)	Multiclass class.	Models from other studies
Zhang et al. (2023)	Zhang et al. (2023)	GPT-2	Custom dataset	Yes (IPs, ports, timestamps)	Threat detection	No
Mahmoodi and Jameii (2024)	Mahmoodi and Jameii (2024)	LLaMA 2	CIC-IDS-2017	Unclear	DDoS detection	DNN, CNN, LSTM
Zhang et al. (2024)	Zhang et al. (2024)	GPT-3.5, GPT-4	DTL-IDS 5G dataset 2023	Yes (IPs)	Binary class.	No
Guastalla et al. (2024)	Guastalla et al. (2024)	GPT-3.5, GPT-4, Ada	CIC-IDS-2017, Urban IoT	Yes (dst port)	Binary class.	MLP
Houssel et al. (2024)	Houssel et al. (2024)	GPT-4, LLaMA 3 8B Instruct	NF-UNSW-NB15-v2, NF-CSE-CIC-IDS2018-v2	Unclear	Binary class.	RF, DT, LSTM NN, DANN
Nwafor et al. (2024)	Nwafor et al. (2024)	GPT-3.5	CIC IoT 2023	No	Binary and Multiclass class.	RF, GB
Bui et al. (2024)	Bui et al. (2024)	LLaMA2-7/13B, Mistral-7B, GPT-3.5, GPT-4	Custom Dataset	Yes (IPs)	Multiclass class.	XGB, RF, CNB, SVM, Ridge Class., Logistic Regr.

3 Materials and methods

Figure 1 illustrates the benchmark workflow, which outlines the step-by-step application and evaluation of the selected AI methods described in Sect. 3.4, for both binary and multiclass intrusion detection.

3.1 Dataset description

This study employs the CIC IoT 2023 dataset (Neto et al. 2023), a realistic and publicly available dataset designed for intrusion detection in IoT and smart home environments. It contains traffic generated by 105 real IoT devices, including cameras, sensors, microcon-

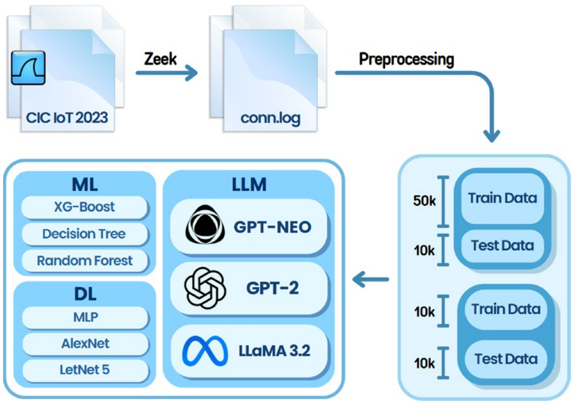


Fig. 1 Benchmark workflow

trollers, and smart speakers, operating under both normal and attack conditions. The data was collected using a port mirroring setup, with PCAP and CSV files preserving the traffic for offline processing and analysis. The dataset covers 33 distinct attack types grouped into seven categories: Distributed Denial-of-Service (DDoS), reconnaissance, web-based exploits, spoofing, brute-force, Denial-of-Service (DoS), and Mirai-based botnet attacks. Benign traffic was generated from natural device interactions, such as sensor data reporting, streaming, or issuing voice commands. Although the study is limited to this single dataset, its diversity of attack scenarios, and inclusion of real-world IoT device behaviour make it a strong representative benchmark for evaluating intrusion detection systems in resource-constrained environments. As such, it provides a focused and realistic context for analysing LLM-based approaches, with broader generalization left for future multi-dataset evaluations.

Given the size of the dataset, we applied a downsizing strategy to enable efficient model training while maintaining behavioural diversity. We selected a representative subset of PCAP files that included the main attack families along with benign traffic:

- **Benign** - regular IoT device behaviour without malicious activity;
- **DDoS** - large-scale flooding attacks, including HTTP floods and TCP PSH-ACK floods;
- **Mirai** - botnet-based attacks leveraging GRE or UDP payloads;
- **Scanning** - reconnaissance behaviour such as OS fingerprinting and port probing.

Table 2 summarizes the total number of PCAP files available in the dataset, their combined size, and the details of the subset chosen for this study. This subset preserves a balanced representation of traffic types while making the training process computationally feasible.

3.1.1 Exploratory analysis of traffic characteristics

A detailed exploratory analysis using Zeek (Zeek Network 2025) was performed to better understand the structural and behavioural patterns present in the dataset. These patterns played a key role in guiding both preprocessing and the evaluation of model performance.

Key observations include:

- Protocol usage varied significantly by class:

Table 2 Total number of PCAP files available in CIC IoT 2023 dataset, their combined sizes, and the details of the selected subset for this study

Label	Number of available file	Size of available files	Selected files	Size of selected files
Benign	4	6.51GB	BenignTraffic.pcap, BenignTraffic1.pcap, BenignTraffic2.pcap, BenignTraffic3.pcap	6.51GB
DDoS	176	322.74GB	DDoS-PSHACK_Flood.pcap, DDoS-PSHACK_Flood1.pcap, DDoS-HTTP_Flood.pcap	4.38GB
Mirai	76	142.07GB	Mirai-udpplain.pcap, Mirai-udpplain1.pcap, Mirai-greip_flood.pcap, Mirai-greip_flood1.pcap, Mirai-greeth_flood.pcap, Mirai-greeth_flood1.pcap	4.383GB
Scanning	76	142.07GB	Recon-OSScan.pcap, Recon-PortScan.pcap	11.4GB

- Mirai traffic was transmitted exclusively over UDP, with no TCP or ICMP present.
- DDoS traffic occurred primarily over TCP.
- Scanning and Benign traffic were the only types using ICMP.

This clear separation suggests that protocol (proto) can be a strong feature for classification, particularly in distinguishing Mirai from the rest.

- Service fields from Zeek logs revealed useful insights:
  - In Benign traffic, DNS was the most common service (over 40%), followed by unknown/empty services (30%), and smaller amounts of NTP, SSL, and HTTP.
  - DDoS flows lacked identified services in 88% of the cases, while the remaining 10% used HTTP.
  - Mirai traffic was even more anonymized: over 99.8% of flows had no associated service label.
  - Scanning flows were mostly undefined (84%) but still showed some usage of DNS (10%) and HTTP (2%).

These distributions indicate that benign traffic tends to be more service-rich and structured, while attack traffic often appears protocol-only or ambiguous.

- Connection states (conn_state) also revealed clear differences:
  - S0 (connection attempt seen, no reply) was dominant in Mirai (99.8%) and frequent in other classes (34% in DDoS, 26% in Benign, 24% in Scanning).
  - Other common states such as SF (normal TCP session) were more prevalent in benign flows, reflecting stable, completed connections.
  - REJ (rejected connections), RSTR, and OTH states were more frequently observed in Scanning and DDoS traffic, likely indicating probing or aggressive attack behaviour.

- History field patterns (Zeek's encoding of TCP flag sequences) also offered interesting indicators:

  - Common benign patterns included regular handshakes and data exchange (Dd, D, ^dD).
  - Attack traffic, especially DDoS and Mirai, showed erratic patterns such as repeated SYN or RST packets (S, Sr, ^hadf).
  - Scanning traffic was more varied but included aggressive probe signatures such as R, Dd, and ^dD.

- Data volume and packet size observations:

  - DDoS traffic showed high variability in the number of origin bytes, but generally low compared to benign flows.
  - Mirai exhibited consistently small numbers of response bytes.
  - Benign and Scanning flows included significantly more packets and bytes in both directions.

- Connection duration:

  - Although exact averages were not quantified in the original version, it was qualitatively noted that Mirai flows had the shortest durations, followed by DDoS. In contrast, Benign and Scanning flows typically lasted longer.

This suggests that duration may serve as a useful feature in identifying high-speed or bursty attacks like Mirai.

Together, these observations underscore that the dataset contains rich structural distinctions between traffic classes. These distinctions motivated the selection and normalization of features, helped design the input encoding strategies for different models, and informed our interpretation of model errors, particularly when benign traffic was misclassified as scanning, or vice versa.

3.2 Zeek-based preprocessing and feature selection

The original PCAP files selected for this study were processed using Zeek, an open-source network traffic analyser, to extract structured flow-level information (Zeek Network 2025). Specifically, we focused on Zeek's conn.log file, which summarizes each network connection with a set of descriptive features.

The following key preprocessing steps were applied to prepare the flows in conn.log for modelling:

- Removal of Potentially Biasing Identifiers: To ensure models focus on behavioural patterns rather than memorizing specific endpoints or artifacts, the following fields were removed:

  - ts (timestamp),
  - uid (unique connection ID),
  - id fields (IP addresses and ports).

- Handling Missing Values: Missing numerical values were replaced with zeros, and missing categorical values were filled with appropriate placeholders.
- Filtering Flows with Packet Loss: A portion of the dataset showed packet loss, identified via the missed_bytes field in Zeek logs. To reduce noise and improve data quality, all flows with over 1%-byte loss were removed, resulting in the exclusion of 63833 flows.

After preprocessing, a curated set of features was retained for training Machine Learning, Deep Learning, and Large Language Models. Table 3 summarizes the fields extracted from Zeek's conn.log, indicating which features were selected for modelling.

We retained only the fields marked with a checkmark, which were selected based on their behavioural relevance. Fields excluded (marked with a cross) were removed to prevent potential bias or information leakage from static or environment-specific attributes. This curated and normalized feature set provided a solid foundation for training all the models evaluated in this study, balancing behavioural richness with generalizability.

Table 3 Features extracted from Zeek conn.log and their usage

Feature	Description	Used for modelling
ts	Timestamp of the first packet in the connection	✗
uid	Unique identifier assigned to each connection	✗
id	4-tuple of IP addresses and ports	✗
proto	Transport layer protocol (e.g., TCP, UDP)	✓
service	Application-layer service detected	✓
duration	Duration of the connection (in seconds)	✓
orig_bytes	Payload bytes sent by the originator	✓
resp_bytes	Payload bytes sent by the responder	✓
conn_state	Final connection state (e.g., SF, REJ)	✓
local_orig	Whether the connection was initiated locally (T/F)	✓
local_resp	Whether the responder is local (T/F)	✓
missed_bytes	Number of bytes missed during connection capture	✓
history	Encoded TCP flag transitions observed during connection	✓
orig_pkts	Number of packets sent by the originator	✓
orig_ip_bytes	IP-layer bytes sent by the originator	✓
resp_pkts	Number of packets sent by the responder	✓
resp_ip_bytes	IP-layer bytes sent by the responder	✓
tunnel_parents	UID(s) of encapsulating tunnel connections (if any)	✓

Table 4 Example input formats for ML/DL and LLM models

Model type	Input example
ML/DL	[1, 18, 0.398803, 0, 0, 2, 1, 1, 0, 6817, 1, 60, 1, 40, 0]
LLM	tcp - 0.398803 0 0 REJ T T 0 Sr 1 60 1 40 (empty)

```
{"proto": "tcp", "service": "-", "duration": "0.398803",
"orig_bytes": "0", "resp_bytes": "0", "conn_state": "REJ",
"local_orig": "T", "local_resp": "T", "missed_bytes": "0",
"history": "Sr", "orig_pkts": "1", "orig_ip_bytes": "60",
"resp_pkts": "1", "resp_ip_bytes": "40", "tunnel_parents":
"(empty)"}
```

Listing 1 Example of alternative prompt format in JSON.

3.3 Input representation strategies

Following preprocessing and feature selection, two input formats were prepared to accommodate the architectural differences between traditional models and Large Language Models (LLMs). While both formats preserved the same semantic information about each network flow, their structure was adapted to suit each model family (see Table 4).

For Machine Learning (ML) and Deep Learning (DL) models, flows were encoded as fixed-length numerical vectors. Categorical features (e.g., proto, service, conn_state, local_orig, local_resp) were mapped to integers, and numerical features were normalized where needed.

The choice of label encoding was a pragmatic decision to mitigate the issue of dimensional explosion. Several features, particularly proto, service, conn_state and history, have a high number of unique values. Using one-hot encoding would have created a very high number of new, highly sparse features.

For LLMs, each flow was formatted as a tokenized text prompt, combining stringified categorical and numerical feature values in a space-separated sequence. This allowed the model to process structured flows similarly to natural language inputs, enabling contextual pattern recognition through attention mechanisms. This textual representation allowed LLMs to process structured flows as sequential inputs, capturing contextual dependencies through attention mechanisms.

This dual-representation strategy ensured that all models, regardless of architecture, received consistent and semantically rich input tailored to their learning mechanisms.

Additionally, an alternative prompt format was tested in which each flow was encoded as a JSON object prior to tokenization. As it is displayed in Listing 1 categorical and numerical fields were formatted as key-value pairs. However, while this approach preserved semantic structure, it consistently led to a performance degradation of approximately 1% across recall, precision, and F1-score compared to the concatenated format. Given this drawback, the JSON-based representation was excluded from the final evaluation benchmark.

3.4 Model architectures

In this study, three model families were evaluated for the task of intrusion detection: Machine Learning models, Deep Learning models, and Large Language Models. This section describes the architectures and selection rationale for each group.

3.4.1 Machine learning and deep learning baselines

Traditional ML and DL models served as strong performance baselines, providing a reference point for evaluating the effectiveness of Large Language Models.

- Decision Tree (DT): A simple, interpretable model that recursively partitions the feature space based on feature values. It provides fast inference and easy visualization but it is prone to overfitting on complex datasets.
- Random Forest (RF): An ensemble of Decision Trees trained via bootstrap aggregation (bagging). Random Forests improve generalization and reduce variance. They are particularly effective in noisy or high-dimensional spaces.
- XGBoost (XGB): A powerful gradient boosting algorithm that builds additive models in a forward stage-wise fashion. XGBoost is known for its efficiency, ability to handle missing data, and strong performance in structured data tasks like intrusion detection.
- Multilayer Perceptron (MLP): A fully connected feedforward neural network with three hidden layers of decreasing size. MLPs are capable of modelling nonlinear relationships between features but they may struggle with temporal or sequential dependencies.
- Gated Recurrent Unit (GRU): A recurrent neural network variant designed to model sequential patterns efficiently. GRUs introduce gating mechanisms to manage long-term dependencies and they are less computationally intensive than LSTMs. Although GRUs are typically designed for sequential data, we included this model for comparative consistency with prior IDS research and to explore whether recurrent architectures could still extract implicit temporal-like patterns (e.g., packet timing statistics or burstiness) from flow-level feature vectors, even though flows represent aggregated snapshots rather than raw packet sequences.
- LeNet-5: A classical Convolutional Neural Network (CNN) architecture, adapted here for intrusion detection by reshaping flow features into 2D matrices. While originally designed for image recognition, LeNet-5 can capture local feature patterns even in structured flow data.

All ML models were implemented using the scikit-learn and XGBoost libraries, and DL models were implemented using TensorFlow/Keras.

3.4.2 Large language models (LLMs)

The core innovation of this study lies in applying Large Language Models (LLMs) to structured flow-based intrusion detection tasks. Three LLMs were selected based on their open availability, architecture diversity, and computational feasibility, with the IoT-oriented nature of the dataset also guiding this selection. While traditional NIDS typically operate at centralized points within the network, recent efforts have explored flow-based IDS deployments on edge devices such as Raspberry Pi (Bhandari et al. 2023; Huda et al. 202; Mehavilla et al. 2022). These scenarios benefit from lightweight, self-contained models capable of local inference without cloud reliance, thereby enabling low-latency detection, enhancing data privacy, and supporting operation in bandwidth-constrained environments. In this context, our study examines whether compact LLMs can satisfy these constraints while still delivering competitive performance.

- GPT-2 (Radford et al. 2019): A 117M parameter decoder-only Transformer model developed by OpenAI. GPT-2 introduces strong language generation capabilities through large-scale unsupervised pretraining. Its moderate size makes it suitable for fine-tuning on structured prompts without excessive computational requirements.
- GPT-Neo-125M (Black et al. 2021): An open-source Transformer model developed by EleutherAI, intended as an alternative to GPT-3. GPT-Neo uses architectural enhancements to improve efficiency and flexibility. Its relatively small footprint (125 million parameters) enables efficient experimentation across training strategies.
- LLaMA-3.2-1B (LLaMA 2025): A lightweight version of Meta AI's LLaMA family, consisting of approximately 1.3 billion parameters. Designed for efficiency and broader accessibility, LLaMA models combine strong performance with manageable resource demands, making them attractive for intrusion detection tasks on structured data.

These LLMs were either fine-tuned from existing pretrained models, trained from scratch, or domain-adapted using unlabeled network traffic before fine-tuning, as detailed in Sect. 3.5. All LLMs were obtained from Hugging Face (Hugging Face 2025).

While all models in this study, including ML, DL, and LLMs, operate on structured flow records composed of a fixed set of fields, the representation and processing approach differ. Traditional ML and DL models ingest fixed-length feature vectors directly, whereas LLMs process the same information as a serialized sequence of tokens. This token-based formulation enables the use of attention mechanisms, which may allow LLMs to model interactions between flow attributes in a more flexible manner. While this does not change the underlying structure of the data, it introduces a different inductive bias that may support richer feature interactions or patterns not easily captured by traditional models.

3.5 LLM training strategies

To fully explore the potential of Large Language Models (LLMs) for flow-based intrusion detection, three distinct training strategies were employed. These strategies aimed to assess how different levels of prior knowledge, pretraining, and adaptation affect model performance on the classification task.

3.5.1 Fine-tuning pretrained LLMs (Experiment 1)

In the first strategy, publicly available pretrained LLMs were fine-tuned directly on the structured flow data. Each LLM was initialized with weights learned during large-scale, general-domain text pretraining and then adapted to the intrusion detection task.

- The LLMs were trained to classify each flow as either benign or one of the attack class-

es, based on the tokenized input representation described in Sect. 3.3.
- Fine-tuning was performed using parameter-efficient adaptation techniques to optimize resource usage while preserving general linguistic knowledge.
- Models fine-tuned under this approach included GPT-2, GPT-Neo-125M, and LLaMA-3.2-1B.

This strategy allowed the models to leverage their pretrained contextual reasoning abilities and adapt them to the specific semantics of network traffic flows.

3.5.2 Training LLMs from scratch (Experiment 2)

The second strategy involved training LLMs from randomly initialized weights, without relying on any prior general-domain knowledge.

- The model architecture remained the same as in the fine-tuning scenario, but all weights were randomly initialized at the start of training.
- Only GPT-2 and GPT-Neo-125M were used for this experiment, due to the larger computational demands associated with training bigger models like LLaMA-3.2-1B from scratch.
- The goal was to evaluate whether the inductive biases of Transformer architectures alone, without external pretraining, could support effective flow classification.

Training from scratch allowed a direct assessment of the intrinsic suitability of LLM architectures for structured cybersecurity tasks.

3.5.3 Domain-specific pretraining and fine-tuning (Experiment 3)

The third strategy introduced an intermediate pretraining phase on domain-specific, unlabelled flow data before fine-tuning for classification.

- A corpus of Zeek conn.log entries, accumulated from years of network monitoring, was used for unsupervised next-token prediction pretraining.
- Models first learned flow-specific token patterns (e.g., relationships between protocol, byte counts, and connection states) without access to any labels.
- After this domain-specific pretraining, the models were fine-tuned on the labelled intrusion detection dataset for binary and multiclass classification tasks.
- Only GPT-2 and GPT-Neo-125M were employed in this strategy, as domain-specific pretraining followed by fine-tuning was computationally intensive.

This approach tested whether exposing the models to unlabelled but domain-relevant traffic prior to supervised learning could enhance classification accuracy and generalization.

Table 5 presents the summary of the explained LLM Training Strategies.

Each of these training strategies provided complementary insights into the role of pretraining, domain adaptation, and architectural biases in the performance of LLMs for intrusion detection. Details regarding training hyperparameters, optimization procedures, and computational setup are provided in Sect. 3.6.

Table 5 Summary of LLM training strategies

Experiment	Description	Models used
Experiment 1	Fine-tuning pretrained LLMs	GPT-2, GPT-Neo-125M, LLaMA-3.2-1B
Experiment 2	Training LLMs from scratch	GPT-2, GPT-Neo-125M
Experiment 3	Domain-specific pretraining + fine-tuning	GPT-2, GPT-Neo-125M

Table 6 Number of flow instances available in the selected PCAP files detailed in Table 2

Label	Instances
Begin	294952
DDoS	4495277
Mirai	13362549
Scanning	390072

3.6 Experimental setup

This section describes the dataset partitioning, hardware and software environment, and the evaluation metrics used in the experiments. Consistency across models and experiments was prioritized to ensure fair comparisons.

3.6.1 Dataset splitting and class balance

To enable both binary and multiclass classification experiments, the dataset was carefully partitioned:

- A balanced subset was constructed by selecting an equal number of flows from each traffic class (Benign, DDoS, Mirai, Scanning) after preprocessing.
- Two different training scenarios were considered:
  - 10,000 flows per class for initial experiments.
  - 50,000 flows per class to evaluate performance scaling with more data.
- For each scenario, 10000 flows per class were held out as a test set, ensuring no overlap with the training datasets.

This sampling ensured class balance across both binary and multiclass tasks and allowed evaluation of model robustness with varying training set sizes. All models, including ML, DL, and LLMs, were trained and tested on the same splits to ensure comparability.

Table 6 summarizes the total number of available flow instances extracted from the selected PCAP files (detailed in Table 2). As shown, the dataset contained a sufficient number of flows in each category, and therefore no oversampling or data augmentation was required to achieve class balance in either the training or test sets.

3.6.2 Model training configurations

To ensure fairness and comparability across model families, specific training configurations and hyperparameters were used for each group of models.

For Machine Learning and Deep Learning models, widely accepted standard settings were used, as summarized in Table 7.

For the Deep Learning baseline, the scikit-learn MLPClassifier was configured with three hidden layers containing 128, 64, and 32 neurons, respectively. A Rectified Linear Unit (ReLU) activation function was applied to all hidden layers to introduce non-linearity. The output layer, implicitly managed by the MLPClassifier, consisted of one neuron per class and employed a softmax function to generate a probability distribution across the classes. Model training was performed using the multinomial cross-entropy loss function, which is a standard choice for multi-class classification tasks.

The LeNet-5 architecture was adapted to handle one-dimensional tabular data. In this configuration, the network input was a single-dimensional feature vector. Each data record, after completing the preprocessing pipeline, was represented as a vector of shape $(F,)$, where F denotes the total number of features.

All LLMs were trained for a maximum of 10 epochs using QLoRA (Dettmers et al. 2023) due to computational constraints and to ensure fairness across model families. Our main objective was not to maximize performance per model, but to evaluate relative performance under equal training budgets. This constraint is realistic in edge settings and highlights each model's efficiency in low-data, low-compute regimes. The configuration is summarized in Table 8.

These settings were selected to balance training efficiency with model performance across the different architectures.

3.6.3 Hardware and software environment

All experiments were conducted using a controlled virtualized infrastructure to guarantee consistency:

- CPU: Intel®Xeon®Silver 4210 (40 cores).
- RAM: 62.5 GB.
- GPU: NVIDIA RTX 2060 with 6 GB VRAM.
- Operating System: Ubuntu Server 22.04 LTS.

The software stack included:

Table 7 Hyperparameters used for ML and DL baseline models

Model	Hyperparameters
XGBoost	Objective: multi:softmax, Estimator: Decision Tree
Decision Tree	Default parameters (scikit-learn) max_depth=None, min_samples_split=2, min_samples_leaf=1
Random Forest	Number of estimators: 100, Random state: 0
MLP	Max iterations: 1000, Random state: 42, Hidden layers: (128, 64, 32)
GRU	Epochs: 50, Validation split: 20%
LeNet-5	Epochs: 50, Validation split: 20%

Table 8 LLM training and QLoRA configuration

Category	Parameter	Value
Training	Epochs	10
	Train batch size	8
	Evaluation batch size	16
	Learning rate	2e−4
	Optimizer	paged_adamw_32bit
	Gradient accumulation steps	2
	Scheduler	Constant
LoRA	Rank	4
	Alpha	32
	Dropout	0.01
LoRA Targets	GPT-2	c_attn, c_proj, c_fc
	GPT-Neo-125M	q_proj, k_proj, v_proj, out_proj, c_fc, c_proj
	LLaMA-3.2-1B	q_proj, k_proj, v_proj, o_proj, gate_proj, up_proj

- Python 3.10 .
- scikit-learn 1.3.0 (for ML models).
- XGBoost 1.7.6 (for gradient boosting models).
- TensorFlow 2.12 / Keras 2.12 (for DL models).
- Hugging Face Transformers 4.36, PEFT 0.6.0, and BitsAndBytes 0.41 (for LLM training and fine-tuning).

Random seeds were fixed where applicable to promote reproducibility. All model training and inference were executed on the same hardware.

3.6.4 Evaluation metrics

Model performance was evaluated using a combination of classification, efficiency, and resource usage metrics:

- Precision: The proportion of true positive predictions among all positive predictions made by the model.
- Recall: The proportion of true positives identified among all actual positives.
- F1-Score: The harmonic mean of precision and recall, balancing false positives and false negatives.
- Inference Time: Measured in flows per second processed, to assess real-time suitability.
- Resource Consumption:
  - RAM usage,
  - CPU utilization,
  - GPU utilization and memory usage (for DL and LLMs),
  - Disk size of the trained model.

In addition to overall metrics, confusion matrices were analysed to better understand per-class behaviour, particularly to assess misclassification tendencies between similar classes like Benign and Scanning.

All metrics were computed separately for:

- Binary classification (Benign vs. Malicious),
- Multiclass classification (Benign, DDoS, Mirai, Scanning).

Macro-averaged scores were reported to ensure that each class contributed equally to the overall evaluation, providing a balanced view of per-class performance even when class sizes were equal in the sampled subsets.

4 Results and discussion

4.1 Classification performance

The classification performance of all evaluated models is reported for both binary (Benign vs. Malicious) and multiclass (Benign, DDoS, Mirai, Scanning) intrusion detection tasks. Performance metrics include raw precision (Prec.), recall (Rec.), and F1-score, consistent with prior IDS benchmarking studies. Results are presented in Table 9 (training set: 10,000 samples per class) and Table 10 (training set: 50,000 samples per class). The best-performing model in each group (ML, DL, and LLMs), determined by the highest F1-score, is shown in italics, with the top ML/DL model and the top LLM experiment additionally shown in bold.

4.1.1 Analysis

Across both training sizes and classification models, LLMs outperform DL models and approach the performance of ML baselines. The best LLM setup, for multiclass classification, GPT-Neo under Experiment 1 (fine-tuning pretrained), achieves an F1-score above 0.96. Meanwhile for binary classification, in training with 50000 samples per class, GPT-2 under Experiment 1 is preferred, but in training with 10000 samples per class GPT-Neo under Experiment 1 is still the best option. Notably, LLMs exhibit low sensitivity to task complexity. Unlike DL models, their performance remains stable across both binary and multiclass classification, even with moderated-sized training data (the original 612 GB dataset was reduced by more than 450 GB). This suggests a robust ability to address specific classification challenges, particularly the partial overlap between benign and scanning traffic, as further illustrated in the t-SNE visualizations (Sect. 5.1).

In contrast, Deep Learning models, particularly GRU and LeNet-5, show a notable drop in multiclass performance, with F1-scores falling below 0.85, even when trained on larger datasets. This suggests difficulty in modelling nuanced distinctions between similar traffic types like benign and scanning. This suggests that while they are effective in binary classification, they struggle with the increased complexity of distinguishing similar attacks.

ML models, particularly XGBoost, retain the best overall performance and stability across both classification tasks, achieving over 0.96 F1-score in all cases.

Table 9 Performance metrics obtained by all experiments when using the training dataset with 10,000 samples per class. Best models per group are italicized; top ML/DL and LLM models are bolded

Train 10k		Machine learning			Deep learning			Experiment 1			Experiment 2		Experiment 3	
		DT	RF	XGB	MLP	GRU	LeNet	GPT2	GPT-Neo	LLaMA	GPT2	GPT-Neo	GPT2	GPT-Neo
Bin	Rec	0.9565	0.9601	***0.9660***	*0.9552*	0.9514	0.9543	0.9612	***0.9643***	0.9350	*0.9576*	0.9461	*0.9636*	0.9459
	Prec	0.9571	0.9611	***0.9680***	*0.9573*	0.9570	0.9568	0.9636	***0.9666***	0.9327	*0.9614*	0.9503	*0.9655*	0.9493
	F1	0.9552	0.9580	***0.9666***	*0.9557*	0.9525	0.9549	0.9617	***0.9648***	0.9457	*0.9584*	0.9471	*0.9641*	0.9468
Multi	Rec	0.9515	0.9563	***0.9630***	*0.7955*	0.7840	0.7847	0.9598	***0.9597***	0.9418	*0.9508*	0.9347	*0.9603*	0.9475
	Prec	0.9516	0.9563	***0.9648***	*0.7922*	0.7856	0.7864	0.9621	***0.9608***	0.9491	*0.9552*	0.9441	*0.9621*	0.9511
	F1	0.9511	0.9539	***0.9632***	*0.7902*	0.7804	0.7813	0.9598	***0.9598***	0.9415	*0.9509*	0.9344	*0.9604*	0.9478

Finally, as an illustrative example, Fig. 2 presents the evolution of training and evaluation loss across epochs of model GPT-Neo during its fine-tuning process (under Experiment 1) with 10000 samples per class for multiclass classification. The figure also displays the progression of evaluation metrics across epochs. It can be observed that the model effectively adapts to the new data, with the highest F1-score achieved at epoch 7.

4.1.2 Key insight

Pretrained LLMs, especially GPT-Neo under Experiment 1 (Pretrained Fine-Tuning) achieving the highest F1-score among all LLM configurations, provide robust and highly accurate performance across both binary and multiclass classification tasks. They consistently outperform DL models and closely trail ML baselines. Their stable performance across both binary and multiclass tasks indicates that LLMs are well suited for structured classification, even with limited data size. This suggests promising potential in future research for leveraging LLMs in downstream tasks, such as generating natural language explanations or summarizing threats within attack detection pipelines.

4.2 Data size sensitivity

To assess how training dataset size affects model performance, we compared results from two training regimes: 10,000 samples per class (Table 9) vs 50,000 samples per class (Table 10).

4.2.1 Analysis

As expected, most models benefited from increased training data, with improvements more visible in multiclass classification. However, the magnitude of gain varied based on the model architecture and, in the case of LLMs, the training strategy employed.

- ML models showed marginal improvements. For instance, XGBoost F1-score in multiclass classification increased from 0.9632 to 0.9696.
- DL models showed mixed results. GRU achieved a notable but less stable gain, with its F1-score rising from 0.7804 to 0.8430 in multiclass classification. In contrast, MLP and LeNet-5 remained relatively unchanged, suggesting limited capacity to benefit from larger datasets.
- LLMs responded differently depending on the training strategy:

  – Experiment 1 (fine-tuning pretrained) showed marginal improvement, with GPT-Neo in multiclass classification increasing from 0.9598 to 0.9618, demonstrating solid consistency across dataset sizes.
  – Experiment 2 (trained from scratch) showed minimal gains, suggesting that trained from scratch models can achieve near-optimal performance with fewer training samples. For instance, GPT-2 in binary classification increase from a 0.9584 to a 0.9614 F1-score. Displaying that empty models get better results with more data. Figures 3 and 4 illustrate the training and evaluation loss, as well as the evaluation metrics, across epochs for GPT-2 under Experiment 2 with 10000 samples per class

Table 10 Performance metrics obtained by all experiments when using the training dataset with 50,000 samples per class. Best models per group are italicized; top ML/DL and LLM models are bolded

Train 50k		Machine learning			Deep learning			Experiment 1			Experiment 2		Experiment 3	
		DT	RF	XGB	MLP	GRU	LeNet	GPT2	GPT-Neo	LLaMA	GPT2	GPT-Neo	GPT-2	GPT-Neo
Bin	Rec	0.9608	0.9640	***0.9691***	*0.9574*	0.9518	0.9557	***0.9629***	0.9608	0.9435	*0.9614*	0.9581	*0.9584*	0.9493
	Prec	0.9614	0.9650	***0.9711***	*0.9597*	0.9569	0.9578	***0.9665***	0.9653	0.9503	*0.9637*	0.9620	*0.9621*	0.9548
	F1	0.9578	0.9599	***0.9696***	*0.9580*	0.9528	0.9563	***0.9636***	0.9616	0.9449	*0.9619*	0.9588	*0.9589*	0.9504
Multi	Rec	0.9553	0.9610	***0.9667***	0.7929	*0.8526*	0.7964	0.9555	***0.9619***	0.9449	*0.9544*	0.9538	*0.9612*	0.9596
	Prec	0.9555	0.9616	***0.9683***	0.7923	*0.8588*	0.7932	0.9587	***0.9642***	0.9508	*0.9569*	0.9565	*0.9640*	0.9615
	F1	0.9537	0.9563	***0.9696***	0.7902	*0.8430*	0.7912	0.9556	***0.9618***	0.9449	*0.9544*	0.9539	*0.9610*	0.9596

and 50000 samples per class, respectively. In both cases the training and evaluation loss decrease consistently with each epoch, while the evaluation metrics show corresponding improvements.
– Experiment 3 (domain-pretrained then fine-tuned) showed different evolution between binary and multiclass classification. As multiclass classification increases its performance with the amount of data, binary classification of GPT-2 decreases.

4.2.2 Key insight

These results indicate that pretrained LLMs generalize effectively, even with limited training data. Their architecture, particularly the use of token-level contextual modelling, allows them to extract meaningful representations even when trained on moderate-size datasets (10000 flows per class), while maintaining stable performance at larger scales (50000 flows per class). In contrast, non-pretrained DL models require larger volumes of data to approach comparable performance, and even then, they continue to struggle in multiclass classification tasks.

4.3 Confusion matrix analysis

While aggregate metrics such as F1-score provide an overview of model performance, confusion matrices allow for deeper inspection of misclassification patterns, especially in multiclass settings. Figure 5 presents the confusion matrices for the best performing among ML/DL models (XGBoost) in binary and multiclass classification when trained with 10000 samples per class. Figure 6 presents the confusion matrices for the best performing LLM experiments (GPT-Neo, Experiment 1), under the same conditions. Both models were selected based on their highest F1-scores, as reported in Table 9.

4.3.1 Analysis

- XGBoost demonstrates near-perfect classification in multiclass classification, with all four classes sharply separated with very minimal confusion, particularly between Mirai and other attack types. Nonetheless, a small number of Scanning flows are misclassified as Benign, representing the primary source of error. Although scanning attacks are less damaging in isolation, their misclassification is significant because it can allow adversaries to probe a network without detection, creating opportunities for subsequent exploitation. In binary classification the 3.3% of the Malicious traces are misclassified as Benign, while just over 1.7% of the Benign traces are misclassified as Malicious.
- GPT-Neo under Experiment 1 exhibits the cleanest diagonal among the LLM-based confusion matrices. Like the ML models, it achieves high accuracy across attack classes; nevertheless, it tends to confuse Benign traffic with Scanning and vice versa. In binary classification the amount of error is higher than with XGBoost as the metrics have previously shown, the 3.8% of the Malicious traces and the 2.6% Benign traces are misclassified. This highlights a specific vulnerability in LLM classification of malicious flows.

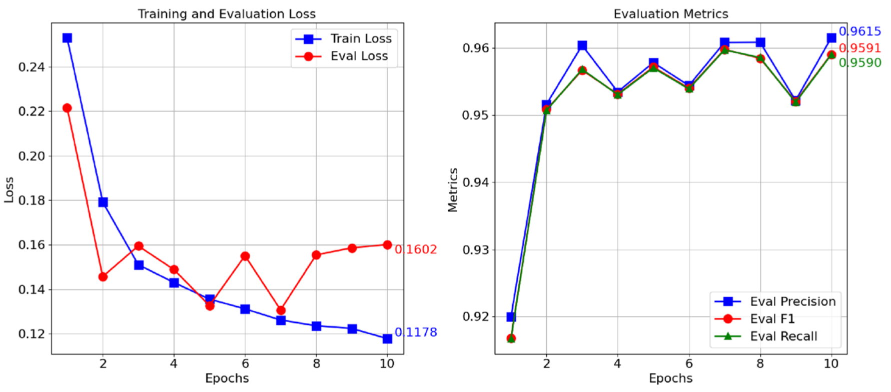


Fig. 2 Training and evaluation loss and evaluation metrics during the epochs when fine-tuning GPT-Neo with 10,000 classes per label in multiclass classification (Experiment 1)

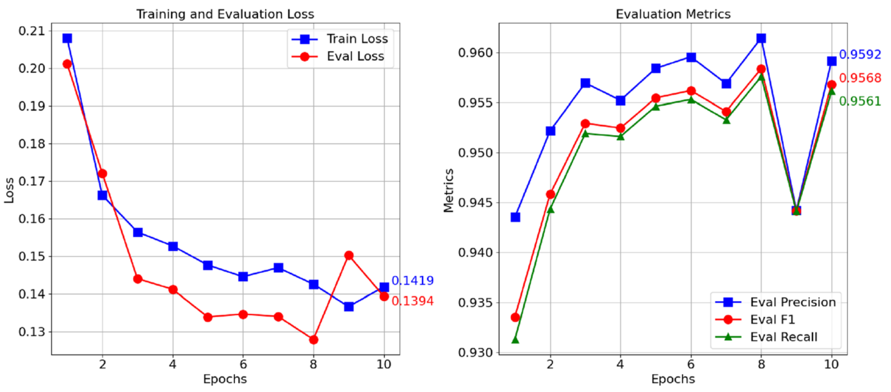


Fig. 3 Training and evaluation loss and evaluation metrics during the epochs when training from scratch GPT-2 in binary classification with 10,000 samples per class (Experiment 2)

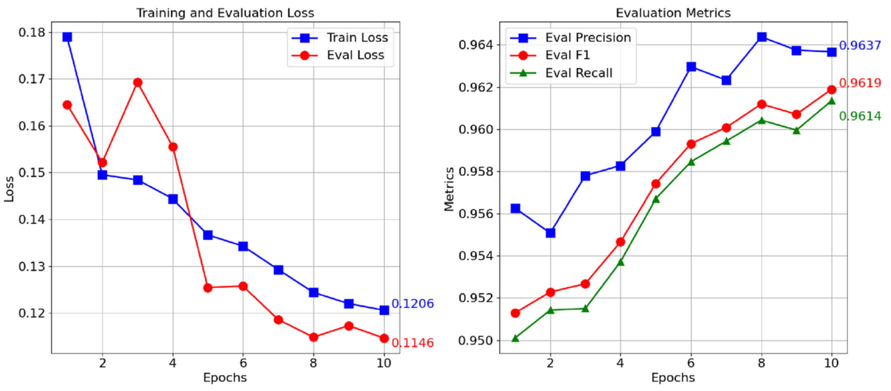


Fig. 4 Training and evaluation loss and evaluation metrics during the epochs when training from scratch GPT-2 in binary classification with 50,000 samples per class (Experiment 2)

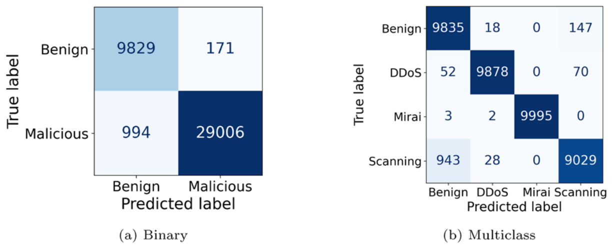


Fig. 5 Confusion matrices for the best-performing among ML/DL models (XGBoost) in both binary and multiclass classification, trained with 10,000 samples per class

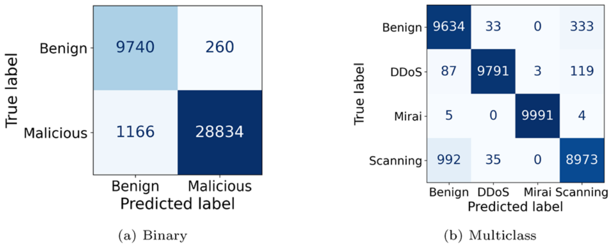


Fig. 6 Confusion matrices for the best-performing LLM experiment (GPT-Neo under Experiment 1) in both binary and multiclass classification, trained with 10,000 samples per class

The confusion matrices reveal a clear pattern: both ML and LLM models produce false negatives where malicious traffic is labeled as Benign, but the rate is lower for XGBoost than for GPT-Neo. This distinction underscores a critical trade-off between sensitivity and specificity across model families.

4.3.2 Key insight

While XGBoost achieves the highest overall F1-scores, GPT-Neo performs competitively, particularly in binary classification, underscoring the effectiveness of token-sequence modelling in LLMs.

4.4 Inference speed and real-time feasibility

To evaluate whether the tested models are viable for real-time or near-real-time intrusion detection, we analysed their inference throughput, measured in flows per second (fps). The results are presented in Table 11 (training set: 10,000 samples per class) and Table 12 (training set: 50,000 samples per class).

4.4.1 Analysis

- ML models demonstrated exceptionally high throughput, with Decision Tree reaching up to 6.25 million fps and XGBoost exceeding 1.5 million fps in binary classification with 10000 training samples per class. These rates significantly surpass those of all other model families, reinforcing the suitability of ML for high-throughput environments.
- DL models exhibited mid-level throughput, with MLP reaching over 200000 fps and GRU operating at approximately 12000 fps.
- LLMs were significantly slower but remain viable for inference in constrained use cases.

Table 11 Flows per second processed across all experiments using the training dataset with 10,000 samples per class

Train 10k	Machine learning			Deep learning			Experiment 1			Experiment 2		Experiment 3	
	DT	RF	XGB	MLP	GRU	LeNet	GPT2	GPT-Neo	LLaMA	GPT2	GPT-Neo	GPT2	GPT-Neo
Bin	6250000	99255	1581027	201005	11834	13468	229	158	105	517	446	244	171
Multi	5633802	89786	456621	161877	12269	12779	247	168	104	447	443	234	173

– The largest model in terms of parameter count (LLaMA) achieved between 104 and 105 fps.
– The most efficient LLM configuration, GPT-Neo under Experiment 2, reached up to 517 fps.
– Experiment 1 and 3 resulted in higher latency, with inference times increasing by approximately 50% with respect Experiment 2, indicating a performance trade-off introduced by pretraining.

Although LLMs are 10,000 times slower than ML models, their inference speeds can still be adequate for applications that process only tens to hundreds of flows per second. Such settings include post-processing detection, triage systems for suspicious traffic, or deployments in low-traffic and edge computing environments. Nonetheless, for high-volume production networks, ML models remain the optimal choice, as combine superior accuracy with the ability to process millions of flows per second.

These results highlight a clear trade-off: ML models remain the most cost-effective solution for large-scale real-time IDS, while LLMs may play a complementary role in specialized settings where computational overhead is acceptable and extended capabilities are desirable.

4.4.2 Key insight

Despite their complexity, LLMs can achieve inference speeds of up 517 to flows per second. Although they are 4 orders of magnitude slower than ML baselines, in future studies their semantic richness may justify their potential use in layered or filtered deployment strategies, where a rationale for classification decisions could be provided.

4.5 Resource usage

While detection performance is essential, practical deployment requires careful attention to resource consumption. In the experimental environment presented in Sect. 3.6.3 we evaluated each model's disk size (MB), RAM usage (GB), CPU and GPU utilization (%), and GPU memory (MB) demand during both loading and inference. These measurements are summarized in Table 13. For Experiments 1 and 3, the reported disk space includes both the original pre-trained model size and the additional storage required by the QLoRA fine-tuned adapters, separated by a plus symbol, reflecting the complete storage footprint needed for deployment.

4.5.1 Analysis

- ML models exhibited the smallest resource footprint, with disk usage of less than 1MB and RAM consumption under 0.6 GB. Additionally, they demonstrated minimal CPU usage (less than 2%) and did not require GPU memory.
- DL models required up to 0.5 MB of disk space and used up to 1.75 GB of RAM during model upload. During inference, DL models maintained steady RAM usage and required up to 25% of CPU usage (in case of MLP) and 4.8 GB of GPU memory (for

Table 12 Flows per second processed across all experiments using the training dataset with 50,000 samples per class

Train 50k	Machine learning			Deep learning			Experiment 1			Experiment 2		Experiment 3	
	DT	RF	XGB	MLP	GRU	LeNet	GPT2	GPT-Neo	LLaMA	GPT2	GPT-Neo	GPT2	GPT-Neo
Bin	4878048	76997	1556420	153846	11111	12461	221	161	104	510	431	236	173
Multi	4494382	69881	419727	137362	11799	12578	241	167	105	508	445	228	162

GRU and LeNet-5).

- LLMs exhibited varying resource usage based on the training strategy employed:

  - Experiment 1 consumed the most disk space due to fine-tuning the pretrained model with additional data.
  - Experiment 3 proved to be the most efficient, with GPT-2 and GPT-Neo requiring less than 2 GB of disk space.
  - RAM usage for LLMs ranged between 1.74 to 2.49 GB during model upload, increasing to 2.74 GB during inference.
  - GPU memory consumption varied between 667 MB and 3.8 GB, with LLaMA under Experiment 1 requiring the most. Notably, GPT-2 under Experiment 1 maintained the lowest GPU usage among LLMs, with a GPU load ranging from 12% to 33% during inference. In contrast, LLaMA under Experiment 1 reached up to 96% GPU usage.

4.5.2 Key insight

As shown in Table 13, most LLMs in the experimental environment (see Sect. 3.6.3) did not fully utilize available resources during inference. While they are not as lightweight as ML models, their GPU usage is lower than expected and, in some cases, even lower than that of DL models. This suggests that edge devices such as NVIDIA Jetson Orin Nano can be used to deploy these models locally, without relying on external communication, ideal for IoT deployments. Moreover, this supports the relevance of the dataset used, which contains information pertinent to IoT environments and scenarios where resource-constrained edge computing is critical.

4.6 Summary

1. LLMs show promising classification capabilities for flow-based intrusion detection, although their deployment may be constrained by performance and resource demands. Pretrained LLMs, particularly GPT-Neo under Experiment 1, demonstrated F1-scores exceeding 0.95, outperforming DL baselines and approaching the performance of robust ML models such as XGBoost. Their ability to generalize across both binary and multiclass tasks, coupled with their strong performance on moderated-size data (the original 612 GB dataset was reduced by more than 450 GB), positions LLMs as competitive alternatives for flow-based intrusion detection systems (IDS).
2. ML models still dominate in performance and efficiency, achieving F1-scores of up to 0.9696. They excel in real-time inference and are highly suitable for lightweight deployment.
3. LLMs introduce a unique trade-off between throughput and resource usage. While they are less efficient than ML models, their semantic depth could in the future offer advantages in more complex tasks requiring contextual understanding. In contrast, ML models such as XGBoost remain the most efficient choice for large-scale deployments, capable of handling millions of flows per second. LLMs are therefore less suitable for real-time, high-volume scenarios due to higher inference costs, but their potential

Table 13 Resource consumption when executing the experiments

		Machine learning			Deep learning			Experiment 1			Experiment 2		Experiment 3	
		DT	RF	XGB	MLP	GRU	LeNet	GPT2	GPT-Neo	LLaMA	GPT2	GPT-Neo	GPT2	GPT-Neo
	Disk	0.25	18.51	0.25	0.29	0.11	0.45	10739.31 +926.76	3969.63 +935.82	9447 +5933.87	14681.23	14853.16	1894.23 +116.94	1916.40 +126.00
Model	RAM	0.47	0.47	0.47	0.49	1.6	1.47	1.87	1.74	2.28	2.38	2.49	1.14	1.35
Upload	CPU	1.1	0.4	0.9	0.9	14.8	12.7	0.40	0.70	0.40	1	0.10	0.60	0.60
	GPU	X	X	X	X	0	0	12	12	24	8	8	9	4
	GPU mem	X	X	X	X	4885	4873	667	715	3879	615	675	677	741
Test	RAM	0.56	0.58	0.56	0.56	1.72	1.60	2.04	2.07	2.37	2.72	2.74	1.43	1.60
	CPU	2.7	3.3	4.0	25.9	13.7	8.9	3-7	3-6	3-12	3-5	3-7	3-5	3-6
	GPU	X	X	X	X	12	1	12-33	23-26	94-96	21-31	22-32	14-30	27-28
	GPU mem	X	X	X	X	4885	4885	669	717	3881	807	859	679	743

(Disk and GPU mem in MB, CPU and GPU utilization in % and RAM in GB) strengths lie in cross-dataset generalization and zero-shot transfer to unseen environments, highlighting a key direction for future research.

4.6.1 Practical implications

Building on the performance and efficiency comparisons summarized above, Table 14 highlights not only the performance of different model families but also their practical trade-offs in real-world deployment scenarios. By considering factors such as inference speed, resource requirements, and scalability, practitioners can make informed decisions when selecting a model for specific intrusion detection scenarios. Deep learning (DL) models are generally suboptimal, as they neither achieve the highest performance nor efficient resource usage, requiring substantial CPU and GPU memory. In contrast, traditional ML models offer the best trade-off, delivering high performance and fast inference without the need for GPU acceleration, making them suitable for high-throughput scenarios. LLMs occupy an intermediate position between ML and DL models. They achieve competitive performance, and although they require GPU and disk resources, their overall resource consumption remains low. However, their inference time is approximately four orders of magnitude slower than ML models, limiting their applicability to scenarios where the network flow rate does not exceed the LLM's processing capacity.

5 Analysis of feature influences

As has been shown in the previous section, traditional Machine Learning methods outperform more complex Deep Learning and LLM-based approaches in the classification task. The simplicity and transparency of ML not only yields excellent results but also makes it easier to trace decision boundaries and interpret outcomes. To better understand why traditional ML models outperform more complex architectures in this task, we apply two explainability techniques: t-distributed stochastic neighbour embedding (t-SNE) for feature space visualization and SHapley Additive exPlanations (SHAP) for feature-level attribution analysis. These techniques help us visualize feature space separability and understand the specific input attributes that drive predictions, further justifying the effectiveness of ML for this task.

5.1 t-SNE analysis of feature space

To gain deeper insight into the discriminative patterns that make Machine Learning so effective in this context, we conducted a t-SNE analysis of the feature space. This non-linear dimensionality reduction technique helps visualize how well-separated the classes are, even before modelling.

Figures 7 and 8 show the t-SNE projections of the traffic flows, derived from the same features used by both ML and DL models, and based on the test dataset comprising 10000 samples per class. This ensures that the projections directly reflect the data used in the evaluation and are consistent with the experimental setup. The 2D t-SNE projection (Fig. 7) reveals distinct clusters for DDoS, Mirai, and Benign traffic, while Scanning flows are heav-

Table 14 Comparative table of the three model families (ML, DL and LLM)

	ML	DL	LLM
Performance	High	Medium	High
Inference time	Fast	Medium	Slow
Disk	Low	Low	High
RAM usage (inference)	Over 0.5GB	Over 1.5GB	Over 2GB
CPU usage (%)	Order of units	Order of tens	Order of units
GPU requirement	No needed	High memory level	Partial usage

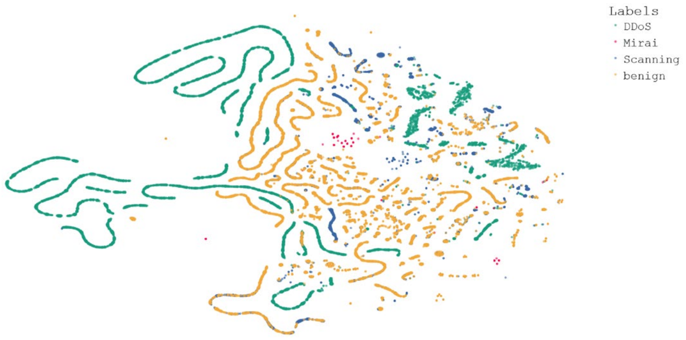


Fig. 7 2D Plot T-SNE

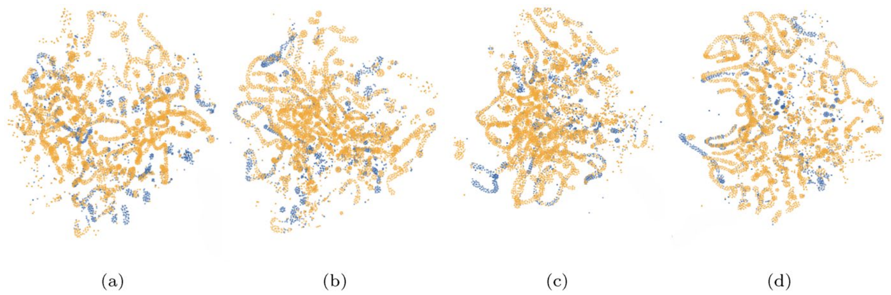


Fig. 8 3D Plot T-SNE of Scanning and Benign

ily overlapped with Benign. This explains the frequent misclassifications observed between these two classes in confusion matrices (see Figs. 5b and 6b).

Interestingly, the 3D t-SNE projection (Fig. 8) reveals a more nuanced separation: Scanning flows occupy a distinct region that is not discernible in two dimensions. This suggests that models capable of learning high-dimensional, non-linear relationships can offer advantages when feature overlap is more subtle or complex.

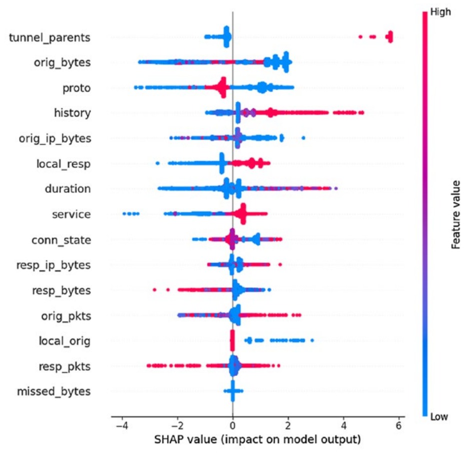
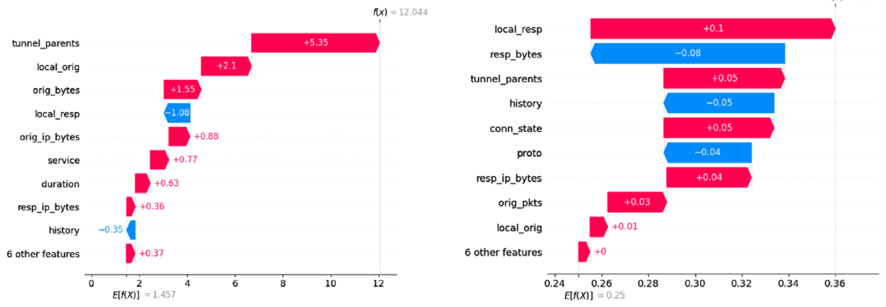


Fig. 9 SHAP summary plot for XGBoost trained for binary intrusion detection



(a) SHAP waterfall plot for a Mirai sample using XGBoost trained for binary detection.

(b) SHAP waterfall plot for benign class of benign sample using Random Forest trained for multiclass detection.

Fig. 10 SHAP waterfalls

5.2 SHAP-based explainability and feature importance

To further explain why Machine Learning excels in this classification task, we applied SHAP (SHapley Additive exPlanations) to one of our tree-based models, XGBoost. SHAP provides both global and local interpretability, showing which features most influence predictions and how individual samples are classified.

Figure 9 presents a global SHAP summary plot for the XGBoost model. Features such as tunnel_parents, orig_bytes, and proto show high impact on model outputs. For example, tunnel_parents is strongly associated with Mirai traffic, which includes tunnelling behaviour in some variants. These results confirm that a small subset of features carries most of the predictive signal, enabling lightweight models to perform robustly without requiring deeper architectures.

To illustrate how individual predictions are influenced by specific features, we generated SHAP waterfall plots for correctly and incorrectly classified samples across different models. Figure 10a illustrates a correctly classified Mirai instance, where the presence of tunnel_parents pushed the prediction toward the correct class. Meanwhile Fig. 10b highlights a benign sample misclassified as Scanning due to an atypical combination of features like proto being equal to UDP, low resp_bytes, and unusual history value. These insights confirm that while models can generally differentiate well, edge cases remain.

SHAP results across the multiclass setup reveal consistent trends: orig_ip_bytes and history are critical for DDoS detection, tunnel_parents identifies Mirai traffic, and a mix of proto, conn_state, and service helps distinguish Benign from Scanning. These insights align with earlier feature ranking results, reinforcing confidence in the ML pipeline.

5.3 Implications

These SHAP and t-SNE results reinforce that tree-based models like XGBoost not only deliver strong predictive performance but also provide high interpretability through a small set of consistently influential features. While scanning traffic remains the most ambiguous class, this highlights an opportunity to test hybrid pipelines, combining ML for efficient classification with LLMs for deeper semantic interpretation when ambiguity arises. Nevertheless, this is outside the scope of this paper.

6 Conclusions

This study explored the application of Large Language Models (LLMs) to the task of intrusion detection in smart home and IoT network environments, using structured flow-level data. By framing network flows as tokenized prompts, we enabled LLMs, traditionally designed for natural language understanding, to process and classify structured security data.

Our findings show that fine-tuned LLMs achieve strong classification accuracy, outperforming DL models and approaching, but not surpassing, the performance of top ML baselines such as XGBoost. Notably, LLMs proved to be robust to both task complexity and training data volume, making them suitable for use cases where labelled data is limited or attack diversity is high.

While LLMs remain less efficient than lightweight ML models in terms of inference speed and resource usage, in future research they may offer valuable advantages in scenarios where interpretability, adaptability, or semantic reasoning are important. Their architectural capacity for modelling feature interactions suggests they could be useful components in hybrid or layered IDS frameworks, particularly where interpretability is valued.

Limitations and Future Work While this study presents one of the first systematic comparisons between LLMs and traditional models on structured Zeek flows, it has some limitations:

- Only three relatively small LLMs (GPT-2, GPT-Neo-125M, and LLaMA-3.2-1B) were evaluated. These models were chosen because they are openly available and reproducible, and their modest resource requirements make them feasible for evaluation in IoT and edge scenarios. However, recent high-performing models such as GPT-4, Mistral 7B, and LLaMA-2-7B may yield better performance.
- This study reported single-run results for each model, in line with standard practice in IDS benchmarks. As such, statistical visualizations (such as critical difference diagrams,

Wilcoxon or Friedman Test) could not be generated reliably.

- Due to architectural constraints, SHAP-based interpretation was limited to tree-based models, leaving LLM decision pathways opaque and preventing analysing of whether their predictions align with the most salient ground-truth features.
- The CIC IoT 2023 dataset focuses on known attack types, limiting generalizability to novel threats.
- This study relied on a stratified train/test split with a held-out test set, which is consistent with common practice in IDS benchmarking. Nonetheless, k-fold or repeated cross-validation was not applied due to the computational cost of training all models across multiple folds.

Future research should investigate:

- Advanced prompt engineering techniques to improve LLM precision, particularly for ambiguous traffic classes like Scanning.
- Continual and online learning frameworks that allow LLMs to adapt to evolving threat landscapes.
- Larger LLM architectures to better understand the trade-offs between classification performance, computational efficiency, and deployment feasibility across the full spectrum of LLMs.
- Hybrid detection pipelines where ML performs fast triage and LLMs serve as secondary classifiers or explainability engines.
- Cross-dataset generalization and zero-shot transfer to assess LLM robustness in unseen environments.
- Integration of attention visualization or token-level attribution methods to bring LLM explainability closer to SHAP-level transparency in ML.

As LLM architectures and fine-tuning techniques continue to evolve, their integration into the cybersecurity domain holds promise for more intelligent, interpretable, and adaptable network defence systems. By rigorously benchmarking LLMs alongside traditional techniques, this work provides an initial step toward evaluating how generative AI models may be integrated into intrusion detection pipelines.

Author contributions All authors contributed equally to this work.

Funding Open Access funding provided thanks to the CRUE-CSIC agreement with Springer Nature. This research was supported by the Government of Aragón (group reference Cenit T31_20R) and the project PID2022-136476OB-I00 funded by MCIN/AEI/10.13039/501100011033/FEDER, EU. L. Mehavilla was supported by a grant for predoctoral research from the Government of Aragón (2023-2027).

Data availability The dataset used for this paper is available on https://www.unb.ca/cic/datasets/iotdataset-2023.html The authors declare that the rest of the data supporting the findings of this study are available on demand due to the research project is on going.

Declarations

Conflict of interest The authors declare no competing interests.

References

Adjewa F, Esseghir M, Merghem-Boulahia L (2024) Efficient Federated Intrusion Detection in 5G ecosystem using optimized BERT-based model. Preprint at arxiv:2409.19390

Ahmad R, Alsmadi I, Alhamdani W, Tawalbeh L (2022) A comprehensive deep learning benchmark for iot ids. Comput Secur 114:102588. https://doi.org/10.1016/j.cose.2021.102588

Ahmad Z, Shahid Khan A, Wai Shiang C, Abdullah J, Ahmad F (2021) Network intrusion detection system: a systematic study of machine learning and deep learning approaches. Trans Emerg Telecommun Technol 32(1):4150. https://doi.org/10.1002/ett.4150

Azzam M, Pasquale L, Provan G, Nuseibeh B (2023) Efficient predictive monitoring of linear time-invariant systems under stealthy attacks. IEEE Trans Control Syst Technol 31(2):735–747. https://doi.org/10.1109/TCST.2022.3196809

Babaey V, Ravindran A (2025) Gensqli: a generative artificial intelligence framework for automatically securing web application firewalls against structured query language injection attacks. Future Internet **17**(1) https://doi.org/10.3390/fi17010008

Bhandari G, Lyth A, Shalaginov A, Grønli T.-M (2023) Distributed deep neural-network-based middleware for cyber-attacks detection in smart iot ecosystem: a novel framework and performance evaluation approach. Electronics **12**(2) https://doi.org/10.3390/electronics12020298

Bhandari G.P, Lyth A, Shalaginov A, Grønli T.-M (2022) Artificial intelligence enabled middleware for distributed cyberattacks detection in IoT-based smart environments. Paper presented at IEEE International Conference on Big Data (Big Data), Osaka, Japan, pp. 3023-3032, https://doi.org/10.1109/BigData55660.2022.10020531

Black S, Leo G, Wang P, Leahy C, Biderman S (2021) GPT-Neo: Large scale autoregressive language modeling with mesh-tensorflow. https://doi.org/10.5281/zenodo.5297715

Bui M.-T, Boffa M, Valentim R.V, Navarro J.M, Chen F, Bao X, Houidi Z.B, Rossi D (2024) A systematic comparison of large language models performance for intrusion detection. Proc. ACM Netw. **2**(CoNEXT4) https://doi.org/10.1145/3696379

Cisco: cyber threat trends report: From Trojan Takeovers to Ransomware Roulette. Cisco. [Online]. Available: https://www.cisco.com/c/en/us/products/security/cyber-threat-trends-report.html [Accessed: May 25, 2025] (2024)

Dettmers T, Pagnoni A, Holtzman A, Zettlemoyer L (2023) LoRA: efficient finetuning of quantized LLMs. Preprint at arxiv:2305.14314

Diaf A, Korba A.A, Karabadji N.E, Ghamri-Doudane Y (2024) Beyond detection: leveraging large language models for cyber attack prediction in IoT networks. Paper presented at 20th International Conference on Distributed Computing in Smart Systems and the Internet of Things (DCOSS-IoT), Abu Dhabi, United Arab Emirates, pp. 117-123, https://doi.org/10.1109/DCOSS-IoT61029.2024.00026.

Fieblinger R, Alam M.T, Rastogi N (2024) Actionable cyber threat intelligence using knowledge graphs and large language models. Paper presented at 2024 IEEE European Symposium on Security and Privacy Workshops (EuroS &PW), Vienna, Austria, pp. 100-111, https://doi.org/10.1109/EuroSPW61312.2024.00018

Fox J (2023) Top cybersecurity statics for 2024. Cobalt. [Online]. Available: https://www.cobalt.io/blog/cybersecurity-statistics-2024 [Accessed: May 25, 2025] (December 8)

García J, Entrena J (2024) Alesanco: empirical evaluation of feature selection methods for machine learning based intrusion detection in iot scenarios. Internet Things 28:101367. https://doi.org/10.1016/j.iot.2024.101367

Guastalla M, Li Y, Hekmati A, Krishnamachari B (2024) Application of large language models to ddos attack detection, 83–99 https://doi.org/10.1007/978-3-031-51630-6_6

Gutiérrez-Galeano L, Domínguez-Jiménez J.-J, Schäfer J, Medina-Bulo I (2025) Llm-based cyberattack detection using network flow statistics. Appl Sci **15**(12) https://doi.org/10.3390/app15126529

Hassanin M, Keshk M, Salim S, Alsubaie M, Sharma D (2025) Pllm-cs: Pre-trained large language model (llm) for cyber threat detection in satellite networks. Ad Hoc Netw 166:103645. https://doi.org/10.1016/j.adhoc.2024.103645

Houssel P.R.B, Singh P, Layeghy S, Portmann M (2024) Towards explainable network intrusion detection using large language models. Preprint at arxiv:2408.04342

Huda S, Musthafa M.B, Nogami Y (2024) Zeek intrusion detection on raspberry pi for iot-based agriculture monitoring systems: Preliminary investigation. In: 2024 IEEE International Symposium on Consumer Technology (ISCT), pp. 372–377 . https://doi.org/10.1109/ISCT62336.2024.10791229

Hugging face: The AI community building the future. [Online]. Available: https://huggingface.co/ [Accessed: May 25, 2025]

Kheddar H (2025) Transformers and large language models for efficient intrusion detection systems: a comprehensive survey. Inf Fusion 124:103347. https://doi.org/10.1016/j.inffus.2025.103347

Khediri A, Slimi H, Yahiaoui A, Derdour M, Bendjenna H, Ghenai C.E (2024) Enhancing machine learning model interpretability in intrusion detection systems through SHAP explanations and LLM-generated descriptions. Paper presented at 6th International Conference on Pattern Analysis and Intelligent Systems (PAIS), EL OUED, Algeria, pp. 1-6, https://doi.org/10.1109/PAIS62114.2024.10541168

LLaMA-3.2-1B: large language model meta AI. [Online]. Available: https://www.llama.com/ [Accessed: May 25, 2025]

Li M, Song X, Zhao J, Cui B (2022) Tcmal: a hybrid deep learning model for encrypted malicious traffic classification. In: 2022 IEEE 8th international conference on computer and communications (ICCC), pp. 1634–1640. https://doi.org/10.1109/ICCC56324.2022.10065869

Liu Y, Wang X, Qu B, Zhao F (2024) Atvitsc: a novel encrypted traffic classification method based on deep learning. IEEE Trans Inf Forensics Secur 19:9374–9389. https://doi.org/10.1109/TIFS.2024.3433446

Mahmoodi M, Jameii S.M (2024) Utilizing large language models for DDoS attack detection. Paper presented at OPJU International Technology Conference (OTCON) on Smart Computing for Innovation and Advancement in Industry 4.0, Raigarh, India, pp. 1-6, https://doi.org/10.1109/OTCON60325.2024.10688345

Martin J (2024)How many cyber attacks occur each day? (2024). Exploding Topics, [Online]. Available: https://explodingtopics.com/blog/cybersecurity-stats [Accessed: May 25, 2025] (September 30)

Mehavilla L, Garcia J, Alesanco A (2022) Hunting@home: plug and play setup for intrusion detection in home networks. In: VI Jornadas Nacionales de Investigacion en Ciberseguridad (JNIC), Bilbao, España, pp. 26–29

Naif Alatawi M (2025) Enhancing intrusion detection systems with advanced machine learning techniques: An ensemble and explainable artificial intelligence (AI) approach. Secur Privacy 8(1):496. https://doi.org/10.1002/spy2.496

Neto E, Dadkhah S, Ferreira R, Zohourian A, Lu R, Ghorbani A (2023) Ciciot2023: a real-time dataset and benchmark for large-scale attacks in iot environment. Sensors **23**(13) https://doi.org/10.3390/s23135941

Nguyen L.G, Watabe K (2022) Flow-based network intrusion detection based on BERT masked language model. Paper presented at 3rd International CoNEXT Student Workshop (CoNEXT-SW '22), New York, NY, USA, pp. 7-8. https://doi.org/10.1145/3565477.3569152

Nwafor E, Baskota U, Parwez M.S, Blackstone J, Olufowobi H (2024) Evaluating large language models for enhanced intrusion detection in internet of things networks. In: GLOBECOM 2024-2024 IEEE Global Communications Conference, pp. 3358–3363. https://doi.org/10.1109/GLOBECOM52923.2024.10901300

Radford A, Wu J, Child R, Luan D, Amodei D, Sutskever I (2019) Language models are unsupervised multitask learners . https://api.semanticscholar.org/CorpusID:160025533

Rodríguez M, Alesanco Mehavilla L, García J (2022) Evaluation of machine learning techniques for traffic flow-based intrusion detection. Sensors 22 (23:9326) https://doi.org/10.3390/s22239326

Saran N, Kesswani N (2023) A comparative study of supervised machine learning classifiers for intrusion detection in internet of things. Proc Comput Sci 218:2049–2057. https://doi.org/10.1016/j.procs.2023.01.181

Udurume M, Shakhov V, Koo I (2024) Comparative evaluation of network-based intrusion detection: deep learning vs traditional machine learning approach. Paper presented at Fifteenth International Conference on Ubiquitous and Future Networks (ICUFN), Budapest, Hungary, pp. 520-525, https://doi.org/10.1109/ICUFN61752.2024.10625037

Vikram A, Shnain A.H, Jeet R, Vennila C, Sahu P, Krishnakumar K (2024) Ai-powered network intrusion detection systems. In: 2024 IEEE international conference on communication, computing and signal processing (IICCCS), pp. 1–6 . https://doi.org/10.1109/IICCCS61609.2024.10763627

Wang S, Xu W, Liu Y (2023) Res-tranbilstm: an intelligent approach for intrusion detection in the internet of things. Comput Netw 235:109982. https://doi.org/10.1016/j.comnet.2023.109982

Wangsa K, Karim S, Gide E, Elkhodr M (2024) A systematic review and comprehensive analysis of pioneering ai chatbot models from education to healthcare: Chatgpt, bard, llama, ernie and grok. Future Internet **16**(7) https://doi.org/10.3390/fi16070219

Zeek Network security monitor. [Online]. Available: https://zeek.org/ [Accessed: May 25, 2025]

Zhang X, Chen T, Wu J, Yu Q (2023) Intelligent network threat detection engine based on open source GPT-2 Model. Paper presented at 2023 International Conference on Computer Science and Automation Technology (CSAT), Shanghai, China, pp. 392-397, https://doi.org/10.1109/CSAT61646.2023.00107

Zhang H, Bin Sediq A, Afana A, Erol-Kantarci M (2024) Large language models in wireless application design: In-context learning-enhanced automatic network intrusion detection. In: GLOBECOM 2024 - 2024 IEEE Global Communications Conference, pp. 2479–2484 . https://doi.org/10.1109/GLOBECOM52923.2024.10901312

Zhou Q, Wang Z (2024) A network intrusion detection method for information systems using federated learning and improved transformer. Int J Semant Web Inf Syst 20:1–20. https://doi.org/10.4018/IJSWIS.334845