# Evaluating Large Language Models for Enhanced Intrusion Detection in Internet of Things Networks

Ebelechukwu Nwafor*, Ujjwal Baskota*, Md Salik Parwez†, Jeremy Blackstone‡, Habeeb Olufowobi†
*Department of Computing Sciences, Villanova University, PA, USA
†Department of Computer Science & Engineering, University of Texas at Arlington, TX, USA
‡Department of Electrical Engineering & Computer Science, Howard University, Washington, D.C, USA
Email: enwafor@villanova.edu

***Abstract*—The Internet of Things (IoT) landscape has grown exponentially in recent years, making robust and efficient intrusion detection systems (IDS) even more critical. While Large Language Models (LLMs) have gained significant traction, their effectiveness in network intrusion detection remains largely unexplored. This paper proposes an LLM-based framework for enhanced threat detection and analysis in IoT networks. We explore using advanced LLMs like OpenAI's Generative Pre-trained Transformer (GPT) model, focusing on techniques such as fine-tuning and embedding similarity. Using real-world intrusion datasets, we evaluate the proposed LLM's performance in detecting common network attacks and compare it with ensemble-based IDS solutions. We assess the efficiency of the LLM in binary class and multiclass classification task using standard metrics, such as accuracy, recall, precision, and F1 scores. While the fine-tuning approach does not produce comparable results to the current baseline ensemble-based IDS models, the embedding approach, however, yields comparable results. This analysis represents a starting point for exploring the utilization of advanced large language models for intrusion detection within an IoT ecosystem.**

***Index Terms*—GPT, IoT, Intrusion detection, Large Language Model**

## I. Introduction

The Internet of Things (IoT) has revolutionized our lives, connecting everyday objects and creating a vast network of heterogeneous devices. This proliferation, while beneficial, has introduced significant security vulnerabilities, especially with the evolving threat landscape of zero-day attacks. IoT devices often lack robust security measures, making them prime targets for cyber attacks. Prior research has investigated intrusion detection systems (IDS) specifically for sensor networks [1] and mobile computing systems [2], but current IDS struggle to keep pace with these rapidly changing threats. The growing number of heterogeneous devices within the IoT ecosystem creates significant challenges for adaptation. Effective IDS are more critical than ever as these devices are increasingly integrated into essential sectors like healthcare, smart homes, and industrial control systems.

Large Language Models (LLMs) such as GPT-4 [3] are the state-of-the-art for natural language processing (NLP). These powerful models excel at complex tasks such as sentiment analysis [4], sentence summarization [5], and machine translation [6] by utilizing complex neural networks with large parameters. Their ability to learn intricate patterns from vast amounts of data suggests a potential application beyond traditional language tasks – intrusion detection in IoT environments. Therefore, to address the limitations of the current IDS and enhance real-time intrusion detection in IoT networks, we explore the potential of these LLMs.

In this paper, we propose a novel approach that leverages the predictive capabilities of LLMs for real-time intrusion detection in IoT networks. We hypothesize that by fine-tuning LLMs on IoT network traffic data, these models can detect and identify sophisticated cyber threats, such as advanced persistent threats (APTs). The proposed approach aims to address the limitations of existing IDS methodologies, providing a more adaptable and efficient real-time security solution for the ever-evolving IoT networks.

The contributions of this paper are as follows:

- **Evaluate LLM Effectiveness for IoT Intrusion Detection:** We assess LLMs' ability to detect network intrusions specifically within IoT environments. This includes evaluating their proficiency in identifying threats using real-world IoT network traffic data.
- **Investigate LLM Adaptability in Dynamic IoT Networks:** We investigate how fine-tuning and embedding similarity of LLMs on IoT network traffic data allows them to adapt to IoT networks' unique and dynamic nature, including examining their ability to handle unexpected variations in network traffic data.
- **Comparative Analysis with Existing IDS Solutions:** We compare the performance of the proposed LLM-based IDS with current ensemble-based IDS solutions using standard metrics (accuracy, precision, recall, F1-score). This evaluation aims to assess LLMs' potential to revolutionize real-time intrusion detection in IoT networks.

## II. Related Work

LLMs have emerged as a promising approach for cyber intrusion detection. Li et al. [7] and Ferrag et al. [8] demonstrate the potential of LLMs in identifying anomalies and threats. However, their application in IoT intrusion detection remains largely unexplored.

Almodovar et al. [9] propose LogFiT, a model that utilize BERT [10] to detect anomalies in the system logs. Nguyen et al. [11] proposed a domain specific Network Intrusion Detection System (NIDS) using BERT on dataset from three different domains. They treated a network flow as a word and

a sequence of flows as a sentence. These studies highlight the effectiveness of LLMs in anomaly detection tasks. Ehsan et. al. [12] address the need for LLMs in cybersecurity by proposing SecureBERT, a fine-tuned model trained on a large cybersecurity dataset. This work emphasizes the ability of LLMs to understand and automate cybersecurity tasks.

While n-gram based approaches [13], [14] and machine learning methods [15] have been successful in intrusion detection, they may not effectively capture the intricate patterns present in IoT network traffic data.

## III. Background

This section provides key concepts essential to understanding our research on intrusion detection in IoT networks.

**Large Language Models (LLMs):** LLMs are advanced AI models trained on massive amount of data to understand, generate, and process complex natural language task [16]. These models excel in a variety of NLP tasks, such as machine translation, sentence completion, text summarization, and enhancing the capabilities of search engines to process extensive data efficiently. They have been shown to produce human-level performance on standardized academic benchmarks, including passing the medical and bar exam at the top percentile [17]. These models are developed using large amounts of data that are processed on AI algorithms such as deep learning transformer models and are further trained using reinforcement learning with human feedback (RLHF) [18]. This approach provides the models the ability to predict subsequent words in a sentence with a level of precision and understanding that closely mimics the human cognitive process.

There are various implementations of LLMs, such as GPT-4, and LLaMA-2 [19], which illustrates the diversity in this field. Although these models share a common foundational approach, they differ in their development and application. This diversity not only outlines the versatility of LLMs in handling complex language tasks but also highlights the rapid evolution and potential of AI in mimicking and enhancing human language capabilities. Their ability to analyze data and identify patterns makes them suitable for analyzing network traffic data in search of potential security threats.

**Embedding:** An embedding is a technique that maps a diverse set of inputs into a compact vector representation, typically with fewer dimensions than the original input. This approach ensures that similar data are closely aligned in the vector space. In simpler terms, an embedding acts like a digital fingerprint, allowing for efficient comparison of data points. Formally, we define an embedding as a function that maps each input feature (a word, sentence, or network traffic data) to a lower-dimensional vector representation. This lower-dimensional space allows for efficient storage and comparison of the data. Mathematically, given a set of inputs $X = \{x_1, x_2, ..., x_m\}$ where each $x_i$ represent input feature, an embedding can be defined as a function $f : X \rightarrow \mathbb{R}^n$ that maps each input feature $x_i$ to an $n$-dimensional vector representation $\mathbf{e}_i$ defined as: $\mathbf{e}_i = f(x_i)$, where $\mathbf{e}_i$ is the $n$-dimensional vector representation (embedding) of $x_i$, and $n$ is the dimension of the embedding space less than dimension of original data space $m$. In IoT networks, embeddings can be used to transform network traffic data into unique vector representations. These representations can then be compared against a database of known attack signatures for faster and more accurate threat identification.

## IV. Threat Model

We assume an attacker with moderate to high technical skills targeting an IoT network, aiming to gain unauthorized access, disrupt operations, or exfiltrate sensitive data. The attacker may exploit unpatched vulnerabilities in IoT devices or communication protocols to launch various attacks. These attacks may include:

- **Denial of Service (DoS)** attack is conducted by flooding the network with bogus, traffic overwhelming devices and disrupting legitimate communication.
- **Brute Force** attack is the process of correctly guessing the login credentials of an IoT system through multiple trial-and-errors methods to gain unauthorized access to devices.
- **Spoofing** is the processing of disguising a communication details such as IP address from a illegitimate source as legitimate with the goal of obtaining an unauthorized access to devices or network.
- **Mirai** attacks are a specialized form of IoT attack that are carried out using a Mirai malware. Once an IoT device has been infected by the malware, it can be remotely controlled and used as bot in a Distributed Denial of Service (DDoS) attack.
- **Reconnaissance** attacks consist of a group of network surveillance techniques, including ping sweeps, operating system identification, vulnerability scanning, port scans, and host discovery on IoT devices within a network. The primary objective of these activities is to gather valuable network data that can be leveraged for potential exploitation.

## V. System Design

In this section, we present a comprehensive overview of our system design. We explore two principal approaches for LLM intrusion detection within an IoT ecosystem: Fine-tuning LLMs and Embedding Similarity. Figures 1 and 2 illustrates the workflow of our proposed approach. We provide detailed descriptions of each approach in Sections V-B and V-C.

### A. Data Processing

We utilize the CIC-IoT-2023 [20], an open-source dataset from the Canadian Institute for Cybersecurity. This dataset was created by generating realistic attacks on a range of IoT systems with a total of 150 IoT devices employed both as attackers and targets. The dataset was collected from 33 different attacks divided into 7 classes. We further process our dataset using the following feature engineering approach. This step involved collating each 34 intrusion labels into 7

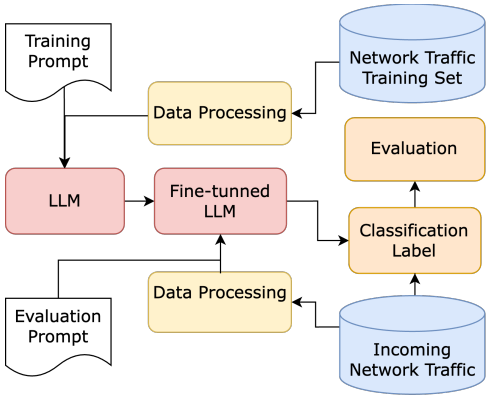


Fig. 1: LLM Fine-tuning System Approach

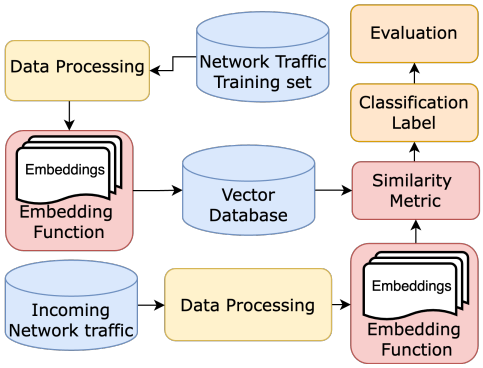


Fig. 2: Embedding Similarity System Approach

types defined in Table I. For example, ARP spoofing and DNS spoofing, both types of spoofing based intrusions, are combined under the label "spoofing". Table I provides an overview of the malicious and benign classes contained in our dataset. Our dataset consists of a large number of features, which can impede the performance of our model causing it to overfit. Feature importance involves ranking the input features in a dataset by its relevance. We employ random forest feature importance technique [21], to select the top 10 features used as input parameters for our fine-tuned LLM. Table II and Figure 3 provide descriptions and rankings of the selected features respectively.

| Dataset | Number of Messages |
|---|---|
| DoS | 215053 |
| Mirai | 13435 |
| BenignTraffic | 5600 |
| Spoofing | 2539 |
| Recon | 1860 |
| Brute Force | 63 |

TABLE I: Number of messages by type

| Feature | Description |
|---|---|
| IAT | Time difference with previous packet |
| Min | Minimum packet length in the flow |
| Max | Minimum packet length in the flow |
| Magnitue | Square root of sum of average incoming and outgoing packets |
| Tot Sum | Sum of packet lengths in a flow |
| AVG | Average packet length in the flow |
| RST Count | Number of packets with RST flag set in the same flow |
| URG Count | Number of packets with URG flag set in the same flow |
| Header Length | Length of the Headaer |

TABLE II: A description of the top ten features

### B. Fine-Tuning Approach

Fine-tuning LLMs involves training a base LLM model on domain-specific dataset to enhance performance on specific tasks. This process enables the model to learn patterns unique to that dataset and provide more tailored domain knowledge. Algorithm 1 details our fine-tuning process. The algorithm

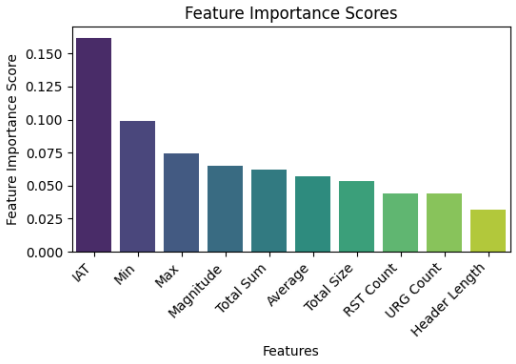


Fig. 3: Top ten features with their relevance scores

**Algorithm 1** Fine-tuning and Evaluating LLM with Network Features an Train and Test prompts

1: **Input:** Network features $F_{\text{network}} = \{f_1, f_2, \ldots, f_n\}$, Training Prompt $P_{\text{train}}$, Test Prompt $P_{\text{test}}$, and ground truth label $L_g = \{l_1, l_2, \ldots, l_n\}$
2: **Output:** Evaluation metrics $M_{\text{evaluation}} = \{m_1, m_2, \ldots, m_n\}$
3: Load base Large Language Model $\mathcal{L}$
▷ Fine-tuning LLM
4: **for** each feature set $f$ in $F_{\text{network}}$ **do**
5: Update $\mathcal{L}$ parameters using $f$ and $P_{\text{train}}$
6: **end for**
▷ Evaluating Fine-tuned LLM
7: $M_{\text{evaluation}} \leftarrow \{\}$
8: **for** each incoming or current feature set $f$ in $F_{\text{network}}$ **do**
9: $l_p \leftarrow$ Generate LLM output using $f$ and $P_{\text{test}}$
10: $m_f \leftarrow$ Compare output label $l_p$ with ground truth label $l_g$
11: Update $M_{\text{evaluation}}$ by appending $m_f$
12: **end for**
13: **return** $M_{\text{evaluation}}$

takes network features, ground truth labels (correct classifications), and a training prompt as input. The prompt is an important component that guides the model on specific tasks to perform given the input data. In Line 3, we load the base LLM and in Line 4 we update the base LLM given the training prompt which serves as our fine-tuned model. In line 7-11, we evaluate our fine-tuned LLM by comparing the ground truth label $l_g$ with the predicted labels $l_p$ generated by the fine-tuned models in line 9. Our evaluation metric $M_{\text{evaluation}}$ is updated and returned in line 13. This metric is used to gauge the performance of our fine-tuned model. Our framework is

robustly designed to integrate various LLMs, however, for our implementation, we utilize OpenAI's GPT-3.5 base model coupled with a training prompt to generate a fine-tuned model.

| Use | Prompt |
|---|---|
| Training | *A network connection in which the time difference with the previous packet was [IAT], minimum packet length in the flow was [MIN], maximum packet length in the flow was [MAX], magnitude was [MAGNITUDE], sum of packet lengths in the flow was [TOT_SUM], average packet length in the flow was [AVG], packet's length was [TOT_SIZE], number of packets with RST flag set in the same flow was [RST_COUNT], number of packets with URG flag set in the same flow was [URG_COUNT], and the header length was [HEADER_LENGTH], was a [ATTACK_TYPE] attack.*" |
| Testing | *What is the type of network connection in which the time difference with the previous packet was [IAT], minimum packet length in the flow was [MIN], maximum packet length in the flow was [MAX], magnitude was [MAGNITUDE], sum of packet lengths in the flow was [TOT_SUM], average packet length in the flow was [AVG], packet's length was [TOT_SIZE], number of packets with RST flag set in the same flow was [RST_COUNT], number of packets with URG flag set in the same flow was [URG_COUNT], and the header length was [HEADER_LENGTH] ? Is the connection type malicious or non-malicious?* |

TABLE III: Prompts used for training and testing.

### C. Embedding Approach

In addition to fine-tuning a base LLM, we employ an embedding similarity technique to assess the similarities between network traffic embeddings, thereby identifying deviations that might indicate anomalous network activity. This process involves recording labeled network embeddings, which provides network scenarios, including both malicious and benign instances. Network traffic is converted into embeddings using a specified LLM embedding function (in our case, OpenAI's GPT-3.5 model) which serves as a basis of our network embedding repository, stored within a vector database for subsequent analysis. For our embedding generation, we utilize network features as defined in section V-A as input to our embedding generation function. This function utilizes OpenAI's embedding API to convert network features into embeddings and produces a set of vectors with a fixed dimension size. To store network embeddings, we utilize faiss [22], an open-source search similarity vector database. Incoming network traffic is evaluated by generating its corresponding embedding and checking if there are any similar embedding stored in our vector database. This comparison allows us to determine if an incoming traffic closely resembles any of the stored embeddings. For this comparison, we employ similarity metrics such as Euclidean distance which measures the distance between two vectors $P = \{p_1, p_2 \dots p_n\}$ and $Q = \{q_1, q_2 \dots q_n\}$ using the formula:

$$d(P,Q) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + \cdots + (p_n - q_n)^2}$$

A score closer to zero indicates a high similarity to an existing malicious or benign stored embedding instance which leads to a classification of an incoming network signal. The objective of this method is to offer a lightweight approach to network traffic classification, relying on similarity metrics. To enhance the precision of our classifier, we define a threshold for our similarity metric. If the most similar embedding among the stored network embeddings is benign and falls below this threshold, it indicates that the closest match in our database is not sufficiently similar, suggesting that the incoming embedding does not belong to the presumed benign category. Consequently, the incoming network traffic is classified as belonging to the malicious category for further analysis. We apply a threshold of 5% which indicates that an embedding of an incoming network traffic must be 95% similar to a stored benign network embedding to be classified as benign. A detailed explanation of our embedding methodology can be found in Algorithm 2. The initialization of the vector database occurs in line 3, where each network embedding representing network interactions, denoted as $E_{\text{train}}$ is stored. From lines 6-12, the procedure involves comparing each new incoming network embedding with those previously stored in database $V$. This comparison also utilizes a predefined threshold to assess the similarity between the stored and incoming embeddings. Finally, evaluation metrics are produced in line 14.

**Algorithm 2** Evaluate embeddings based on similarity threshold

1: **Input:** Training set embeddings $E_{\text{train}}$, Test set embeddings $E_{\text{test}}$, Ground truth labels $L_{\text{ground}}$, Similarity threshold $\tau$
2: **Output:** , Accuracy $A$, Precision $P$, Recall $R$, F1-score $F1$
3: Initialize a Vector database $V$ with a dimension of $E_{\text{train}}$
4: Add $E_{\text{train}}$ to $V$
5: Initialize empty list $L_{\text{pred}}$ for predicted labels
6: **for** each embedding $e$ in $E_{\text{test}}$ **do**
7: $D, i \leftarrow$ Search in $V$ for the nearest neighbor of $e$
8: **if** $D < \tau$ and $L_{\text{ground}}[i] = 1$ **then**
9: Append 1 to $L_{\text{pred}}$ ▷ Embedding is considered non-malicious
10: **else**
11: Append 0 to $L_{\text{pred}}$ ▷ Embedding is considered malicious or flagged
12: **end if**
13: **end for**
14: $A, P, R, F1 \leftarrow$ Calculate accuracy, precision, recall, and F1-score between $L_{\text{ground}}$ and $L_{\text{pred}}$
15: **return** $A, P, R, F1$

TABLE IV: Performance Metrics for both LLM model and ML model trained with numerical data

| Training Data | Class Type | Accuracy | Precision | Recall | F1 Score |
|---|---|---|---|---|---|
| Balanced Data (200 Network Traffic) | GPT-Binary Class | 0.490 | 0.372 | 0.490 | 0.337 |
| | GPT-Multi Class | 0.130 | 0.192 | 0.234 | 0.203 |
| Imbalanced Data (200 Network Traffic) | GPT-Binary Class | 0.230 | 0.210 | 0.230 | 0.216 |
| Balanced Data (4000 Network Traffic ) | GPT-Binary Class | 0.015 | 0.007 | 0.500 | 0.148 |
| Imblanced Data (4773 Network Traffic) | GPT-Binary Class | 0.015 | 0.007 | 0.500 | 0.014 |
| | Gradient Boosting | 0.994 | 0.994 | 0.994 | 0.994 |
| | Random Forest | 0.994 | 0.993 | 0.994 | 0.993 |
| | Embedding Similarity | 0.976 | 0.976 | 1.000 | 0.988 |

## VI. Evaluation

### A. Evaluation Metrics

We evaluate the feasibility of both the fine-tuning and embedding similarity approach using evaluation metrics such as precision, recall, and F-1 score. Precision defines the number of true malicious traffic out of all the malicious traffic detected by the classifiers. This is defined as: Precision $= \frac{\text{True Positive}}{\text{True Positive+False Positive}}$. Recall is defined as the number of malicious instances detected out of the entire malicious traffic present, which is characterized using: Recall $= \frac{\text{True Positive}}{\text{True Positive+False Negative}}$. F-1 score is defined as the harmonic mean of the precision and recall: $\text{F1} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision+Recall}}$.

### B. Fine-tuning using OpenAI's GPT-3.5

Our goal is to compare the performance of the fine-tunned LLM across various scenarios, including balanced and imbalanced class distributions, as well as using numeric and categorical data. Our dataset consists of numeric data representing various network features. To evaluate the performance of the fine-tuned model, we evaluate the performance of the fine-tuned LLMs in both binary and multi-class classification tasks. The transformation of numerical data into categorical values involves grouping the numerical values into predefined categories such as 'low', 'medium', and 'high'. This categorization helps in assessing the model's performance across different levels of data intensity. To further facilitate our model's understanding of the network features, we convert each row of data into a descriptive sentence where each of the features corresponds to a feature name as outlined in table III. Below we provide further details on the diverse evaluation scenarios employed to test the performance of our fine-tunned LLM.

*1)* ***Balanced and Imbalanced Datasets****:* To evaluate the model's performance across balanced and imbalanced datasets, we randomly selected samples from all seven classes. For enhancing the representation of minority classes, we employed the Synthetic Minority Oversampling Technique (SMOTE) [23]. In the balanced dataset, each class is equally represented, ensuring uniform sample sizes across all classes. Conversely, the proportion of the imbalanced data across classes from the original dataset is reflected in the training dataset.

*2)* ***Binary class****:* For binary classification, we assigned the label "non-malicious" to the benign class and the label "malicious" to all six types of malicious classes. Our objective was to assess the performance of the model when trained on a data with only two class labels, under both balanced and imbalanced data scenarios. We also evaluated the model's performance by comparing training outcomes using various dataset sizes. For the imbalanced dataset, we utilized 200 network dataset and 4,773 network traffic. For the balanced dataset, we compared results using sets of 200 and 4,000 network traffic data.

*3)* ***Categorical and non-categorical****:* To convert the numerical network features into categorical values, we utilize the quantile-based approach, dividing the dataset into five distinct groups. These groups are labeled as 'low', 'moderate', 'high', and 'maximum' which corresponds to the increasing quantiles of data distribution. Specifically, 'minimal' represents the lowest one-fifth of the numerical values for each feature, while 'maximum' represents the highest one-fifth. This method of quantile grouping was consistently applied across all features within the dataset, and was utilized for both balanced and imbalanced datasets as well as for binary and multi-class classification task.

### C. Baseline Ensemble Models: Random Forest and Gradient Boosting

We evaluated the effectiveness of our fine-tuned LLM against baseline machine learning models which are known to perform well on intrusion detection tasks [24]. Specifically, we trained two ensemble learning techniques that combine predictions from multiple decision trees to produce a final output. This method effectively captures the underlying patterns in both majority and minority classes by aggregating predictions from multiple trees. On the other hand, Gradient Boosting builds an ensemble of weak learners sequentially. Each new learner is specifically trained to address errors made by the preceding learners, gradually improving the model's accuracy. For data preparation, we used label encoding to represent the seven classes labels numerically, ranging from 0 to 6. The dataset was subsequently divided into training (80%) and test (20%) subsets, setting the stage for a comprehensive evaluation of these models compared to our fine-tuned LLM.

### D. Results

As indicated in Tables IV and V, the performance of the fine-tuned models produced less compelling results across all scenarios when compared to the baseline models. Furthermore, models trained on balanced datasets consistently outperformed those trained on imbalanced datasets in binary classification

TABLE V: Performance Metrics for LLM model trained with categorical data

| Training Data | Class Type | Accuracy | Precision | Recall | F1 Score |
|---|---|---|---|---|---|
| Balanced Data (200 Network Traffic) | GPT-Binary Class | 0.455 | 0.380 | 0.455 | 0.352 |
| | GPT-Multi Class | 0.100 | 0.037 | 0.098 | 0.044 |
| Imbalanced Data (200 Network Traffic) | GPT-Binary Class | 0.305 | 0.281 | 0.305 | 0.285 |
| Balanced Data (4000 Network Traffic) | GPT-Binary Class | 0.015 | 0.007 | 0.500 | 0.148 |
| Imbalanced Data (4773 Network Traffic) | GPT-Binary Class | 0.985 | 0.492 | 0.500 | 0.496 |

tasks, regardless of whether the data was numerical or categorical. The Machine Learning ensemble learning models (Gradient Boosting and Random Forest) achieved a better performance with the numerical dataset. This can largely be attributed to their proficiency in handling numerical data, which aligns closely with the nature of the original dataset used in this study. Specifically, among balanced binary class label, the models trained with numerical features slightly outperformed those trained with categorical data. Interestingly, the results from the fine-tuning model trained with smaller datasets achieved better results than those trained with larger datasets, a phenomenon observed in both balanced and imbalanced datasets for both categorical and numerical data types. This counterintuitive outcome suggests that smaller, well-curated datasets might be more effective for training under certain conditions. Finally, it can also be noted that the embedding similarity approach produced detection results comparable to those of ensemble learning learning models. This might be as a result of the signature-based approach which the embedding similarity utilizes by creating and comparing network embeddings similar to a unique signature of the network traffic data.

## VII. Conclusion and Future Work

In this paper, we evaluate the effectiveness of employing LLMs for intrusion detection within the IoT ecosystem. We provide a comprehensive analysis of fine-tuned LLMs across multiple scenarios to explore their potential in detecting security threats. While this approach shows promise, it generally does not yield results as compelling as those achieved with traditional machine learning techniques or a signature-based embedding similarity approach. This disparity may be attributed to the opaque, 'black box' nature of most LLMs, which can obscure the interpretability of their decision-making processes for specific tasks. This analysis represents an initial step towards establishing a robust and efficient natural language-based framework for intrusion detection in the IoT environment. Future work entails evaluating additional open-source LLMs and modifying the architecture of the neural networks used, to optimize them specifically for intrusion detection tasks.

## References


[1] I. Butun, S. D. Morgera, and R. Sankar, "A survey of intrusion detection systems in wireless sensor networks," *IEEE communications surveys & tutorials*, vol. 16, no. 1, pp. 266–282, 2013.

[2] S. Şen and J. A. Clark, *Intrusion detection in mobile ad hoc networks*. Springer, 2009.

[3] OpenAI, "Gpt-4 technical report," 2023.

[4] D. Araci, "Finbert: Financial sentiment analysis with pre-trained language models," *arXiv preprint arXiv:1908.10063*, 2019.

[5] Y. Liu and M. Lapata, "Text summarization with pretrained encoders," *arXiv preprint arXiv:1908.08345*, 2019.

[6] W. Jiao, W. Wang, J.-t. Huang, X. Wang, and Z. Tu, "Is chatgpt a good translator? a preliminary study," *arXiv preprint arXiv:2301.08745*, 2023.

[7] F. Li, H. Shen, J. Mai, T. Wang, Y. Dai, and X. Miao, "Pre-trained language model-enhanced conditional generative adversarial networks for intrusion detection," *Peer-to-Peer Networking and Applications*, pp. 1–19, 2023.

[8] M. A. Ferrag, M. Ndhlovu, N. Tihanyi, L. C. Cordeiro, M. Debbah, and T. Lestable, "Revolutionizing cyber threat detection with large language models," *arXiv preprint arXiv:2306.14263*, 2023.

[9] C. Almodovar, F. Sabrina, S. Karimi, and S. Azad, "Can language models help in system security? investigating log anomaly detection using bert," in *Proceedings of the The 20th Annual Workshop of the Australasian Language Technology Association*, 2022, pp. 139–147.

[10] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[11] L. G. Nguyen and K. Watabe, "Flow-based network intrusion detection based on bert masked language model," in *Proceedings of the 3rd International CoNEXT Student Workshop*, 2022, pp. 7–8.

[12] E. Aghaei, X. Niu, W. Shadid, and E. Al-Shaer, "Securebert: A domain-specific language model for cybersecurity," 2022.

[13] C. Wressnegger, G. Schwenk, D. Arp, and K. Rieck, "A close look on n-grams in intrusion detection: anomaly detection vs. classification," in *Proceedings of the 2013 ACM workshop on Artificial intelligence and security*, 2013, pp. 67–76.

[14] K. Rieck and P. Laskov, "Language models for detection of unknown attacks in network traffic," *Journal in Computer Virology*, vol. 2, pp. 243–256, 2007.

[15] N. Sultana, N. Chilamkurti, W. Peng, and R. Alhadad, "Survey on sdn based network intrusion detection system using machine learning approaches," *Peer-to-Peer Networking and Applications*, vol. 12, no. 2, pp. 493–501, 2019.

[16] Y. Chang, X. Wang, J. Wang, Y. Wu, K. Zhu, H. Chen, L. Yang, X. Yi, C. Wang, Y. Wang *et al.*, "A survey on evaluation of large language models," *arXiv preprint arXiv:2307.03109*, 2023.

[17] J. Achiam, S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, F. L. Aleman, D. Almeida, J. Altenschmidt, S. Altman, S. Anadkat *et al.*, "Gpt-4 technical report," *arXiv preprint arXiv:2303.08774*, 2023.

[18] Y. Bai, A. Jones, K. Ndousse, A. Askell, A. Chen, N. DasSarma, D. Drain, S. Fort, D. Ganguli, T. Henighan *et al.*, "Training a helpful and harmless assistant with reinforcement learning from human feedback," *arXiv preprint arXiv:2204.05862*, 2022.

[19] H. Touvron *et al.*, "Llama 2: Open foundation and fine-tuned chat models," *arXiv preprint arXiv:2307.09288*, 2023.

[20] E. C. P. Neto, S. Dadkhah, R. Ferreira, A. Zohourian, R. Lu, and A. A. Ghorbani, "Ciciot2023: A real-time dataset and benchmark for large-scale attacks in iot environment," *Sensors*, vol. 23, no. 13, p. 5941, 2023.

[21] B. H. Menze, B. M. Kelm, R. Masuch, U. Himmelreich, P. Bachert, W. Petrich, and F. A. Hamprecht, "A comparison of random forest and its gini importance with standard chemometric methods for the feature selection and classification of spectral data," *BMC bioinformatics*, vol. 10, pp. 1–16, 2009.

[22] M. Douze, A. Guzhva, C. Deng, J. Johnson, G. Szilvasy, P.-E. Mazaré, M. Lomeli, L. Hosseini, and H. Jégou, "The faiss library," 2024.

[23] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "Smote: synthetic minority over-sampling technique," *Journal of artificial intelligence research*, vol. 16, pp. 321–357, 2002.

[24] S. S. Dhaliwal, A.-A. Nahid, and R. Abbas, "Effective intrusion detection system using xgboost," *Information*, vol. 9, no. 7, p. 149, 2018.