IoT Device Classification Using Link-level Features for Traditional Machine Learning and Large Language Models

Gabriel Morales, Farhan Tajwar Romit, Adam Bienek-Parrish, Patrick Jenkins, and Rocky Slavin
Department of Computer Science, University of Texas at San Antonio, San Antonio, Texas, USA
{*firstname.lastname*}*@my.utsa.edu, rocky.slavin@utsa.edu*

Keywords: Internet-of-Things, LLM, Traffic Flow, Network Analysis, Networking Standards, Classification

Abstract: Technological advancement has made strides due in part to added convenience in our daily lives. This addition of automation and quick access to information has given rise to the Internet-of-Things (IoT), where otherwise normal items such as kitchen appliances, smartphones, and even electrical meters are interconnected and can access the Internet. Since IoT devices can be accessed anywhere and have user-set behaviors, they transmit data frequently over various networking standards which can be obtained by a malicious actor. While network data is often encrypted, the patterns they construct can be used by such an adversary to infer user behavior, device behavior, or the device itself. In this work, we evaluate various traditional machine learning models for device classification using network traffic features generated from link-level flows to overcome both encryption and differences in protocols/standards. We also demonstrate the viability of the GPT 3.5 large language model (LLM) to perform the same task. Our experiments show the viability of flow-based classification across 802.11 Wi-Fi, Zigbee, and Bluetooth Low Energy devices. Furthermore, with a considerably smaller dataset, the LLM was able to identify devices with an overall accuracy of 79% through the use of prompt-tuning, and an overall accuracy of 63.73% for a larger more common dataset using fine-tuning. Compared to traditional models, the LLM closely matches the performance of the lowest-performing models and even achieves higher accuracy than the best-performing models.

1 INTRODUCTION

In the era of pervasive connectivity, advances in wireless technologies are fueling the growth of the Internet of Things (IoT). Defined as an ecosystem where a diverse range of smart devices interconnect to gather and exchange data, IoT has significant implications for home automation and convenience (Chataut et al., 2023). These advances pave the way for smart homes, which incorporate intelligent devices to manage various household functions (Tripathi et al., 2022). However, as IoT devices proliferate, they bring along an expanded attack surface. Vulnerabilities such as inadequate security measures, insecure network services, and flawed update mechanisms expose these systems to an increased risk of exploitation (Allifah and Zualkernan, 2022).

The convenient factors provided by IoT devices are enabled by their connectivity. However, the degree of trust required to maintain these connections can easily be abused. For instance, one-way authentication can be "spoofed" by phantom devices (Wang et al., 2022). Furthermore, simply by collecting network traffic for these devices, it is possible to build profiles and behavioral examinations to identify them. (Hu et al., 2023). Finally, it has also been shown that even encrypted, anonymized, IoT traffic (for example through a VPN) can be de-anonymized and identify target devices and network behaviors (Li et al., 2022). Through the exploitation of such vulnerabilities, an individual's privacy, anonymity, and devices can be compromised. A first step to this compromise is device classification.

Though these approaches can successfully identify such device characteristics, there are some limitations in techniques that have not been considered which may pose new threats to privacy. One such limitation is the inability to classify IoT devices using multiple network protocols simultaneously (e.g., using one model to classify Wi-Fi, Bluetooth, and Zigbee devices). Such a protocol-agnostic approach would streamline both malicious *and* research endeavors. Similarly, pre-trained state-of-the-art models including widely-versatile natural language processing (NLP) models are becoming more widely available and can potentially be leveraged to improve device identification as they can effectively serve as broad knowledge-bases.

Our work addresses these considerations through the use of traffic flows constructed from *link-level* network data. Network data, by default, is an unorganized sequence of packets. To organize the packet data, traffic flows are used to represent each device's bidirectional communication. For consideration of agnostic device classification, we extract data from packets such as sizes, or addressing information, and calculate subsequent statistical attributes. For example, a flow consisting of one hundred forward packets would maintain a count of the total packets transmitted, the maximum size, among many other similar attributes. By removing network associations, many protocols can be considered simultaneously. This is possible because most protocols share some common features at the link level, such as the MAC-like addresses and the statistical features mentioned above. Furthermore, such statistical packet features (e.g., minimum/maximum packet size, number of packets transmitted, etc.) do not vary based on payload encryption, thus presenting a potentially valuable approach to broader classification techniques.

IoT devices, depending on the manufacturer, and services provided, have deterministic behaviors (Zahan et al., 2023). Correspondingly, we collect link-level traffic to classify them. Among classic methods ranging from a Random Forest Classifier to an SVM, we also consider the use of Large Language Models (LLMs), which have been trained on a large amount of natural language data, in an attempt to capture expressive context, patterns, or dependencies within the data. Through prompt engineering and fine-tuning, we demonstrate how LLMs can be conditioned to classify (in this case generate) a target class based on input examples.

Our contributions are as follows:

1. **Flow-based, Link-level Device Classification:** A flow-based approach to IoT device classification is developed such that it can generalize across different protocols using similar link-level features. We evaluate this method on two datasets, with nine 802.11 Wi-Fi, Zigbee, and Bluetooth Low Energy device types.
2. **Large Language Model Device Classification:** A large language model (LLM) is utilized to perform device classification for network flows from both encrypted and unencrypted traffic. We compare different approaches to prompt-tuning via instruction and fine-tuning to demonstrate the viability of using an LLM for a non-NLP task in this manner.
3. **PROTOFLOW:** An open source tool designed for extensibility to generate protocol-agnostic link-level flows. Additionally, we provide our training scripts and resources for the LLM [1].
4. **Encrypted IoT Traffic Dataset:** A labeled dataset of 10227 (before processing) network flows among 42 IoT Zigbee, Bluetooth Low Energy, and 802.11 Wi-Fi devices within nine device classes. We make this dataset available to the public with our PROTOFLOW repository.

The rest of this paper is organized as follows. Section 2 discusses the technical background. Section 3 presents closely related work. Section 4 presents our tool development, collecting custom data, understanding each protocol, and performing classification using the LLM. Section 5 describes our experiment design while Section 6 discusses the results. Finally, Section 7 presents future work and conclusions.

2 BACKGROUND

In this section we discuss the wireless technology standards we evaluated (Wi-Fi, Bluetooth Low Energy, and Zigbee) and the concept of intra-device data flows. Additionally, the basics of language models and their typical usage are discussed.

2.1 Wireless Networking Protocols

IoT devices utilize a variety of networking protocols, each with unique characteristics influenced by aspects such as hardware limitations and operational environments. Analyzing these protocols individually is essential, yet integrating them could facilitate a comprehensive analysis within smarthome settings. Our work focuses on the widely-used 802.11 Wi-Fi, Bluetooth Low Energy (BTLE), and Zigbee protocols (Horyachyy, 2017; Danbatta and Varol, 2019; Stolojescu-Crisan et al., 2021). The Open Systems Interconnection (OSI) model serves as a standard framework for communication stacks (Kumar et al., 2014). In our approach, we concentrate on the link-level (layer two) of this model, where data encapsulation and physical transfer processes, including hardware addressing, are handled (Kumar et al., 2014; Suresh, 2016). Notably, most encryption methodologies are applied at higher OSI model levels, preserving the integrity of link-level data.

Wireless technologies selection is contingent on the specific application requirements. Wi-Fi, for example, facilitates both Internet connectivity and ex-

[1] https://github.com/Gabriel-Morales/LLM-Paper-Resources

tensive local communication (Banerji and Singha-Chowdhury, 2013). It allows IoT devices ot seamlessly integrate into home networks, establishing connections with other devices and cloud services (Horyachyy, 2017; Wukkadada et al., 2018). BTLE, known for its low-power consumption, supports data and audio transmission with minimal energy use and is capable of forming extensive mesh networks (Horyachyy, 2017). Zigbee[2], another low-power communication technology, operates at slower data rates compared to BTLE but excels in longevity (Ergen, 2004) and supports numerous nodes, ensuring ease of deployment, interoperability, and network resilience (Safaric and Malaric, 2006; Ramya et al., 2011).

2.2 Intra-Device Data Flows

In the context of device communication, a traffic flow is a sequence of related attributes within the interaction between two devices. This concept encompasses both uni- and bidirectional flows. A unidirectional flow represents the sequence of interactions from a source to its destination, whereas a bidirectional flow includes the exchange of communications between a source and its destination, as well as the reverse path (Hayes, 2018; Habibi Lashkari et al., 2017; Draper-Gil. et al., 2016). By aggregating multiple such flows, a comprehensive *flow table* can be constructed.

2.3 Large Language Models (LLMs)

As a general task, language modelling seeks to generate text by predicting the next word in a given sequence (Kandpal et al., 2022). Many traditional models have been proposed and used for language modelling; for example, the LSTM (Chang and Bergen, 2022). However, since the introduction of transformer models (Vaswani et al., 2017), language models have become more advanced and capable. Large language models are trained on massive amounts of data such as LLaMa (Touvron et al., 2023), BART (Lewis et al., 2020), and the GPT family (Roumeliotis and Tselikas, 2023). Versions of these models have been fine-tuned as conversational chatbots, for code generation, summarization, and more.

These types of models have also been adapted to other use-cases besides simple variants of text generation and conversation. Pearse, et. al utilize both commercial LLMs (e.g., OpenAI Codex) and local LLMs to study whether or not code completion capability of these models can help detect vulnerabilities within developer-written source code (Pearce et al., 2023). Furthermore, LLMs have the capability of being applied to software engineering tasks such as testing, requirements engineering, feedback, translation, and other portions of the software development lifecycle (Ozkaya, 2023). The aforementioned use-cases point to the usefulness and expressive power of adapting these LLMs for various tasks, as also discussed in Section 3. Adaptation of these models may be divided into the following categories:

1. Zero-shot Learning: Given desired attributes of an example, the model uses no additional labels to run an inference.
2. Prompt-Tuning: A few (or n) examples are provided as part of a prompt to the model to condition/steer its generation as desired.
3. Fine-Tuning: Using new data, the model is trained to include new examples as part of its knowledge to improve its domain specificity.

3 Related Work

We discuss some related works for device classification, network flow analysis, and LLMs in cross-domain tasks here.

IoTFFID (Hao and Rong, 2023) is an incremental IoT device fingerprinting method using two publicly available datasets: UNSW and Yourthings. This method achieves around 98% accuracy on both datasets. Their contribution includes a transformer-based model with a classification head to identify devices, and adapt to new devices, which avoids catastrophic forgetting as traffic changes. The work is able to expand into 50 device types from 20 with its incremental behavior, noting that the types are *specific devices*. In contrast, we focus on traffic requiring only link-level information across three protocols using an LLM with in-context learning capability.

Zhang et al. (Zhang et al., 2023) perform device identification using Bluetooth on the link layer. The frames are acquired by obtaining radio frequency data using a software defined radio and demodulating it. The packet data is fed into a deep neural network first by stripping out specific identifiers that could generally be “spoofed”: MAC address, company identifier, length of manufacturer-specific advertising data. The neural network contains six layers, with an output size of 15 classes (i.e., devices). The overall performance reaches almost 100%. Our work considers more than just Bluetooth and feature extraction uniformity that is obtainable from the link-layer on major networking protocols.

[2] https://csa-iot.org/all-solutions/zigbee/

Other work has introduced device identification and fingerprinting beyond flow- and packet-levels (Hamdaoui and Elmaghbub, 2022). In particular, a system is developed to perform classification using deep learning on the radio frequency (RF) spectrum. While operating on the LoRa protocol, the work studies inputs using inverse-quadrature (IQ) data and FFT-based data. The testbed consists of 25 transmitting devices, with collected data over different times and days. The authors state that imperfections during manufacturing of such devices cause cause unique impairments in the way the hardware transmits on the RF spectrum. These unique impairments can then be used to distinctly fingerprint the transmitting devices. Their experimental results achieve up to an 84% accuracy using the proposed feature set. While we do not observe the RF spectrum, we do perform fingerprinting; further, we also do not focus solely on a single protocol.

TabLLM (Hegselmann et al., 2023) applies large language models and templated prompting to perform classification on a non-NLP-related task. Specifically, inputs from tabular data are serialized (i.e., transformed) into a natural language-compatible input for the LLM. As an example, a column feature name "heart_rate" with value "93" would be transformed into "The heart rate is 82.". This approach provides meaningful context to the LLM about what the features represent, and evaluate their approach on nine datasets. With the number of examples (shots) ranging from zero to 512, the performance of the classification for TabLLM increases in a linear fashion. Furthermore, the authors state that with a larger number of examples, the template matters less, but with a smaller amount of classes, the LLM relies on the values as well as the feature names. Interestingly, the performance indicates the applicability of LLMs for classification tasks. Our task is not a generic study on LLMs and tabular data, but more focused on using link-level network flows with LLMs for classification.

LLMs also have applications which mix natural language with other inputs, otherwise known as multimodal. Ma et al. introduce LLaViLo, a model to retrieve relevant frames (start and end times) from a video given a query. To achieve the task, an adapter module is developed which fuses embeddings for an input query and an input set of video frames and outputs them as tokens to the LLM. The LLM, using a special masking strategy, is then prompted using guided instruction to predict the time frames in natural language. In comparison to state of the art, their results achieve a higher score, which shows that the trajectory of using LLMs for multi-domain and multimodal tasks is a worthy endeavor. While not specific to security, this work helps motivate a reason why we would like to explore the use of LLMs for our task.

To the authors' knowledge, no other work has performed a protocol-agnostic device classification on the link-layer in addition to using LLMs to attempt the same classification task as a whole.

4 Methodology

To perform device classification, the datasets of network flows must be generated from traffic captures. These captures can vary between protocols, but have common properties. Networking protocols share various attributes within their packet information. These attributes include timing information, addressing information, and length information. These three categories, and variations of such categories, are the only ones always available across protocols at the link-level. This allows us to achieve protocol-agnostic flow generation for classification purposes. In that manner, no specific protocol information, for example Wi-Fi IP addresses and ports, are included. With this in mind, flows are generalizable even to packets that are captured on unencrypted networks; as mentioned earlier, this indicates that even more data can be acquired.

4.1 PROTOFLOW

We seek to classify IoT devices universally. Since classification methods such as machine learning require a good feature set, we need a way to represent uniform features across multiple packet types and protocols. PROTOFLOW[1] is designed to swap between protocols by detecting what type of protocol is present within the first few packets. In addition, parsing the packet types is as simple as adding a new method within the detection branch, as the rest of the parsing logic remains identical. PROTOFLOW generates flow tables (CSV of network flows) as long as the packet format is a universally recognizable one (e.g., pcap). Not only is it able to perform parsing for raw frame traffic (i.e., 802.11 Wi-Fi, Zigbee, link-layer BT), it is also able to identify regular LAN traffic, which means that PROTOFLOW can even generate the flows for UNSW pcap files.

Each flow has its statistical attributes calculated and added as an entry within a dictionary data structure. As PROTOFLOW parses the capture file, it runs through all packets within the file and fills the dictionary for the corresponding communication pairs (reminder that a flow is a bidirectional communication between devices). In addition, an auxilliary list is kept for each pair representing the running sizes for each packet, which can be used to add new statistical attributes if desired.

As part of our experiments, we use two datasets: a larger publicly available dataset of unencrypted, labeled traffic from The University of New South Wales (UNSW) and a smaller, encrypted dataset from our own lab titled ZBW (Zigbee-BTLE-WiFi) [1]. The next sections discuss capturing traffic, protocol preparation, and LLM methodologies.

4.2 Link-Level Feature Set and Classification

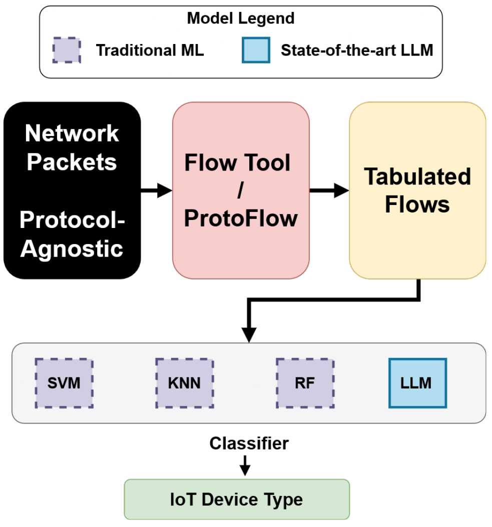

Figure 1: Flow to ML Pipeline.

Prior to device classification, data must be collected. Network data is raw, binary, data which is captured through network cards or other radio-frequency peripherals. Typically, each protocol that one sniffs traffic for uses a special peripheral to capture its data (due to considerations such as protocol specifications or bandwidth). One important consideration prior to creating the data for classification is how one wishes to capture packets: in segments or continuous. This particular matter will alter the number of flows generated and create more or less than desired, even if programmatically set to separate flows. For instance, two packet files will generate more flows, as they would be separated sessions. All of our packet captures are made in segments where, after a timeout of a few minutes, they are split and the capture is looped to save space as they are uploaded to a storage for use later.

As displayed in Figure 1, we begin by capturing raw packet data on various protocols. The block entitled "Network Packets" indicates that packet captures from any protocol (i.e., protocol agnostic) is captured; this represents both our custom dataset acquired from our own lab, and the UNSW dataset, as discussed shortly. Any dataset we acquire which is in the form of packet capture data, is then passed through a flow creation tool (e.g., PROTOFLOW) to output organized flow tables of the data for ML algorithms. PROTOFLOW produces a flow table of 38 features by default, which we reduce to a set of 30 features most generalizable and least likely to cause overfitting or memorization as displayed in Table 1. After the flow tables are created and curated to the feature set we desire, we pass them into various ML classifiers. Each model will then ideally produce a device type during test time corresponding to the patterns it has been trained on.

The custom data we capture is derived from our lab, which consists of 22 unique devices from three protocols, as indicated by Table 2. We also use the UNSW dataset which consists of 31 unique devices. Some device types overlap such as UNSW's Android Phone to our Google Pixel 4a, but many are different. As we are classifying devices on two datasets, and to keep things fair and consistent, we decide to abstract common device types into a category of its own. In particular, we abstract these devices into nine device types, shown in Table 2.

Table 1: Flow Entry Features.

Flow Entry Features	
Source OUI	Dest OUI
Bidirectional Total Packets	Bidirectional Total Bytes
Source to Dest Total Bytes	Dest to Source Total Bytes
Source to Dest Total Packets	Dest to Source Total Packets
Source to Dest Total Duration (ms)	Dest to Source Total Duration (ms)
Bidirectional Total Duration (ms)	Source to Dest Min Packet Size
Source to Dest Max Packet Size	Source to Dest Mean Packet Size
Source to Dest Stdev Packet Size	Dest to Source Min Packet Size
Dest to Source Max Packet Size	Dest to Source Mean Packet Size
Dest to Source Stdev Packet Size	Bidirectional min Packet Size
Bidirectional Max Packet Size	Bidirectional Mean Packet Size
Bidirectional Stdev Packet Size	Source to Dest Transmission Rate (ms)
Dest to Source Transmission Rate (ms)	Bidirectional Transmission Rate (ms)
Source to Dest Transmission Rate Bytes (ms)	Dest to Source Transmission Rate Bytes (ms)
Bidirectional Transmission Rate Bytes (ms)	Protocol

Table 2: Base Link-level Lab Devices Captured.

Category	Wireless Standard	Detail
router	Wi-Fi	Asus Router RT-N12 Asus RT-AC1200GE
router	Wi-Fi & Zigbee	Tp-Link Kasa Router
smart speaker	Bluetooth & Wi-Fi	Bose Home Speaker 300
smart speaker	Bluetooth & Wi-Fi	Sonose One SL
smart speaker	Wi-Fi & Zigbee	Amazon Echo with Hub
smart switch	Wi-Fi	5x C by GE 3-Wire On/Off Toggle
smart assistant	Wi-Fi & Zigbee	Amazon Echo with Hub
mobile	Wi-Fi & Bluetooth	Galaxy A21 Google Pixel 4a
smart bridge	Zigbee & Wi-Fi	Philips Hue Bridge
smart bulb	Zigbee	x3 Philips Hue Bulb
smart camera	Wi-Fi	x2 Blink mini camera x2 Kasa Spot

4.3 LLM-based IoT Device Classification

To study the use of LLMs for device classification using network flows, we consider different ways to interact with the model. The GPT 3.5 LLM is used to perform this task due to its popularity and well-documented API for model interaction. Our pipeline to use the LLM is also represented by Figure 1, where the LLM is the classifer.

4.3.1 Model Preparation

GPT 3.5 follows the format of system role prompt, user query, and assistant response. Because of this format and the generative nature of the model, it is not sufficient to give these models an input and an expected output for training or fine-tuning as with traditional models. A natural language guidance (i.e., instruction) will tell the model exactly what its inputs and expected outputs are to be; this takes place within the system prompt. The system prompt contains an instruction to the model indicating that it is to receive features extracted from the flow table in a specific order, and that its objective is to classify the device type based on those features. A list of names and descriptions for the features it is expected to receive is then provided. Furthermore, the model's prompt indicates that it will receive the inputs for the features in the same order in which they were described. The user queries will be given as a tuple of feature values of a flow (instead of feature names and values) to save token counts. These user queries are given iteratively from the datasets and the response from the model for that query is the device type. GPT 3.5 is tuned to the specific task using the following adaptation techniques:

Zero-shot Learning. Here, the base GPT 3.5 model is used as is, without giving it any examples for the device classes. This gives us a baseline for the model's performance that can be compared with our prompt-tuning and fine-tuning approaches.

Prompt-Tuning. In this case, the same instructions and classes as above are provided, except now we provide various examples of each device type's attributes along with their label within the prompt. This approach, in conjunction with the fine-tuning approach, lets us answer two important questions: how many examples per class are needed to ensure sufficient classification accuracy, and whether prompt-tuning or fine-tuning is the better way to feed those examples to the model.

Fine-Tuning. The process for extracting device classifications from the fine-tuned model is divided into two phases: training and querying.

In the training phase, the model is provided with five training examples for each of the device abstractions. The examples are provided to the model using the fine-tuning methods from the OpenAI API. Each training examples has 3 parts: a system role prompt, a user query, and an ideal assistant response. Unlike the other two adaptations, the prompt contains no descriptions of the inputs given to the model or the order in which they are given. This is done not only to save costs, but to also mimic the training environment of traditional machine-learning tasks, where the model is usually not given any description of the input features. The user query is the same as the other two adaptations, and the ideal assistant response is a string containing the correct device abstraction.

In the querying phase, the fine-tuned model is given the instructions in the same format as described above, similar to the other two adaptations.

The justification for using fine-tuning is the same as that for using prompt-tuning.

5 Experimental Design

In this section we describe the process and implementation of collecting our data and experiment design to evaluate whether it is possible to use both types of data to perform link-level device classification using the models of our choice. In the following, we formulate our research questions (RQs):

- **RQ1:** Can link-layer features extracted from datasets acquired across different protocols and locations be used to identify the IoT devices comprising those networks?
- **RQ2:** Can IoT devices be identified using a LLM using link-layer features?
- **RQ3:** Is an LLM viable for device identification compared to non-LLM models?

5.1 Link-Level Feature Set and Classification

As part of RQ1, we describe the practical design for collecting traffic in our datasets. Our customized dataset, 'ZBW', includes devices running on the Zigbee, Bluetooth and Wi-Fi protocols. Each of the protocols are sniffed without association to the respective networks, enabling us to collect solely link-level data. All Wi-Fi data is captured using monitor mode, which enables us to sniff traffic non-specific to one location; for instance, any devices broadcasting around us (such as another building) can have its data collected if the signal is strong enough. Bluetooth and Bluetooth LE traffic is captured using the Ubertooth One dongle, using the `ubertooth-btle` command for Bluetooth LE traffic and the `ubertooth-rx` for Bluetooth traffic. Zigbee traffic is captured using an Apimote and the `zbdump` command to capture Zigbee frames. In addition, we download a secondary dataset from the University of South Wales (UNSW) consisting of only Wi-Fi-based IoT devices. Furthermore, the UNSW dataset was captured on the local area network (LAN) traffic. For the purposes of the study in this work, however, the primary distinctions between our dataset and the UNSW dataset is simply how they are captured. The significance to this comes down only to obtaining link-level features, which is available in *both* datasets.

Our prior study (Morales et al., 2023) focuses on transferability between encrypted and unencrypted datasets: using encrypted data and decrypted data interchangeably due to consistent packet sizes (as shown in the original paper). However, the study performed in this work redesigns and implements new experiments focusing on classification methods using the original features as a basis: For the first experiment, we try to understand and evaluate how well our link-level feature set works for classification in *both datasets* (i.e., we run classifiers on both). In addition, we reduce the abstraction complexity to cover nine new device types as opposed to 15 device types, and PROTOFLOW has been expanded to work on LAN traffic (thus removing original transferability constraints). Finally, we evaluate these new device types. For the second experiment, we utilize newer, more powerful state-of-the-art models to compare with traditional classification methods. Specifically, using chat-based LLMs, we reason that by treating them like humans to perform an instructed task, they may be able to scale and perform tasks for non-NLP domains. To that end, we treat ChatGPT (GPT 3.5) as a classifier with instructions to contextually analyze and understand link-level features extracted from these two datasets.

As a baseline comparison, we employ three simple to deploy, yet robust, ML algorithms: Random Forest (RF), Support Vector Machine (SVM), and K-Nearest Neighbors (KNN). The choice of these three algorithms allows for an empirical view of the contrast between classic methods and the state-of-the-art models. As preprocessing steps, a standard scaler is used to scale the numerical features of the training data. Then, one hot encoders are used for non-numerical features. Furthermore, as the datasets use addressing identifiers (e.g., MAC addresses, Zigbee addresses), we also extract the organizationally unique identifier (OUI); this particular attribute is reserved to specific manufacturers for each device, which can be mapped and added as a unique feature. Consequently, we build a mapping between OUI addresses, the devices we have in our lab, and those given as a table from the UNSW dataset. Afterward, we add the manufacturer identifier as a feature to our feature set. We believe that this feature will have additional weight in creating contextual representations in the datasets for the LLM. Seven examples per class are provided as training data to the traditional models. In all experiments, we test on 1,012 and 1,023 samples for the encrypted ZBW dataset ($\mathcal{D}_{ZBW}$) and the unencrypted UNSW dataset ($\mathcal{D}_{UNSW}$) respectively. Our choice for the number of testing samples was based on the token-based monetary cost required for using the GPT 3.5 API. These datasets are used across all subsequent experiments.

5.2 LLM-based IoT Device Classification

For RQ2 and RQ3, we now describe how we interact with each model and implement the experiments for each adaptation technique.

Our interactions with the model are enabled by a script written in Python and the OpenAI API. Before issuing requests, price is approximated using the `tiktoken` library and the price-per-token of the model. The $\mathcal{D}_{UNSW}$ and $\mathcal{D}_{ZBW}$ datasets are used for testing. The base prompt which we give our model for the prompt-tuning task is as follows:

```
Your task is to use various attributes of the
    traffic transmitted/received by different
    IoT devices to classify them into one of 9
     specific device types. You'll be provided
     a comma-separated tuple with 30 traffic
    attributes in the following order:
Communication Attributes [attributes]
    Transmission Attributes (Time in
    milliseconds, Size in bytes, Packets in
    count) [attributes]
Packet Size Attributes (Size in bytes) [
    attributes]
Transmission Rate Attributes (Bytes per
    millisecond, Packets per millisecond) [
    attributes]
You can only classify the traffic into only
    one of the following 9 IoT device types,
    exactly as they appear below:
[attributes]

You can only classify the traffic into only
    one of the following 9 IoT device types,
    exactly as they appear below:
```

```
camera
router
mobile
switch
speaker
bulb
computer
motion_sensor
printer

Note: Not all attributes might be available
    for every frame, and some may be more
    telling than others in identifying the
    device type.

Now, you will be given [N] examples of tuples
    for each type of device. The order of the
    attributes in the tuples will be same as
    the order in which they were defined above

Examples for camera:
[N examples]

Examples for router:
[N examples]

Examples for mobile:
[N examples]

Examples for switch:
[N examples]

Examples for speaker:
[N examples]

Examples for bulb:
[N examples]

Examples for computer:
[N examples]

Examples for motion_sensor:
[N examples]

Examples for printer:
[N examples]
```

Where the attributes enclosed by square brackets are expanded by the authors to include the full feature list and their descriptions.

The base prompt for the fine-tuning task is as follows:

```
Given a comma-separated tuple with 30 traffic
    attributes of the traffic transmitted/
    received by different IoT devices,
    classify the input tuple to a device type.
```

5.2.1 Zero-Shot

The temperature set for the model generation is set to 0.5. The temperature refers to how imaginative the model will be. According to the OpenAI documentation, the lower the temperature, the more deterministic and strict the output will be; the higher the temperature, the more imaginative and random the model will be. The temperature value can range from zero to two with a default value of one[3]. Our chosen value for temperature is half the default value in order to get more deterministic outputs than the default and avoid potentially unpredictable behavior from extreme temperature values such as 1.

5.2.2 Prompt-Tuning

Three sub-experiments with three, five, and seven examples *per class* are conducted to identify an optimal number of examples to include in the prompts. Given the nine classes, a total of 27, 45, and 63 examples are provided, respectively. For each sub-experiment, the performance of the model is evaluated on $\mathcal{D}_{UNSW}$ and $\mathcal{D}_{ZBW}$. Just as with the zero-shot task, we set the temperature to be 0.5. All of the examples in the prompt-tuning sub-experiments are from the $\mathcal{D}_{UNSW}$ dataset.

5.2.3 Fine-Tuning

In the fine-tuning task, we provide five examples for each of the nine device classes for a total of 45 examples as JSON file entries for the API. These examples are the same as those for the prompt-tuning sub-experiment with five examples per class. As with the prompt-tuning experiments, we conduct queries on $\mathcal{D}_{UNSW}$ and $\mathcal{D}_{ZBW}$. The temperature of the model is set to 0.5 for this experiment.

6 Experimental Results

Here, we discuss the results of our experimentation.

6.1 RQ1: Link-level Device Classification

Tables 3 and 4 show the results for traditional ML algorithms on the link-level data. $\mathcal{D}_{UNSW}$ has a low accuracy of 28.73% and 24.53% for both the KNN and SVM algorithms, respectively. However, the RF algorithm reaches a 52.49% accuracy. From $\mathcal{D}_{ZBW}$t, both the KNN and SVM algorithms reflect the same pattern as before with low accuracy. RF performs best across both datasets with the highest accuracy

[3] https://platform.openai.com/docs/api-reference/chat/create

Table 3: UNSW (Unencrypted) Link-level Classification Results.

		UNSW								
		KNN			RF			SVM		
Class	Support	Precision	F1	Recall	Precision	F1	Recall	Precision	F1	Recall
Bulb	12	4.72	8.75	58.33	14.28	25	100	4.76	8.80	58.33
Camera	203	42.85	2.85	1.47	39.28	38.59	37.93	52	20.55	12.80
Computer	342	47.36	23.68	15.78	90.47	42.50	27.77	33.96	9.11	5.26
Mobile	101	15.04	15.88	16.83	30.12	37.45	49.50	31.70	18.30	12.87
Motion Sensor	134	33.81	45.47	69.40	94.05	80.85	70.89	60.62	59.00	57.46
Printer	22	21.33	32.98	72.72	50.0	65.62	95.45	7.88	14.22	72.72
Router	80	35.16	37.42	40.0	90.80	94.61	98.75	31.74	27.97	25
Speaker	79	42.42	47.19	53.16	42.51	57.72	89.87	21.89	31.42	55.69
Switch	50	29.70	39.73	60.0	49.33	59.2	74	21.73	31.91	60
Macro Avg	1023	30.27	28.22	43.08	55.65	55.72	71.57	29.59	24.59	40.01
Weighted Avg	1023	38.24	25.33	28.73	67.37	52.61	52.49	38.20	23.24	24.53
Total Accuracy		28.73			52.49			24.53		

Table 4: ZBW (Encrypted) Link-Level Classification Results.

		ZBW								
		KNN			RF			SVM		
Class	Support	Precision	F1	Recall	Precision	F1	Recall	Precision	F1	Recall
Bulb	1	0	0	0	0	0	0	0	0	0
Camera	115	0	0	0	10.63	18.95	86.95	5.47	9.31	31.30
Computer	0	0	0	0	0	0	0	0	0	0
Mobile	23	1.05	1.94	13.04	60	21.42	13.04	0	0	0
Motion Sensor	0	0	0	0	0	0	0	0	0	0
Printer	0	0	0	0	0	0	0	0	0	0
Router	860	50.94	5.91	3.13	42.85	0.69	0.34	0	0	0
Speaker	9	0	0	0	100	50	33.33	0	0	0
Switch	4	0	0	0	0	0	0	0	0	0
Macro Avg	1012	6.49	0.98	2.02	26.68	11.38	16.71	0.68	1.16	3.91
Weighted Avg	1012	43.31	5.07	2.96	39.88	3.67	10.77	0.62	1.05	3.55
Total Accuracy		2.96			10.77			3.55		

for both. The scores for $\mathcal{D}_{ZBW}$ suggest that the original data distribution is too low for the chosen devices. $\mathcal{D}_{UNSW}$ has a larger and much broader data distribution of traffic flows, which can point to its better performance.

6.2 RQ2: LLM-based IoT Device Classification

Tables 5 and 6 show the prompt-tuning results for the UNSW and ZBW Testing Sets, whereas Tables 8 and 7 show the results for fine-tuning.

Overall, the weighted accuracy score (percentage of devices correctly identified) across both $\mathcal{D}_{UNSW}$ and $\mathcal{D}_{ZBW}$ is 11.70% for zero-shot learning, 49.97% for prompt-tuning with three examples, 50.56% for prompt-tuning with 5 examples, 49.92% for prompt-tuning with seven examples, and 33.23% for fine-tuning with five examples. Therefore, prompt-tuning with five examples provides the best overall performance. Fine-tuning with five examples provides the best classification accuracy for the $\mathcal{D}_{UNSW}$ (63.73%) while prompt-tuning with five examples provides the best accuracy for $\mathcal{D}_{ZBW}$ (79.44%). The model achieves better classification accuracy on the $\mathcal{D}_{ZBW}$ than the UNSW Testing dataset for all the experiments except the fine-tuning experiment. In nine out of the 10 experiments, the weighted precision is higher than the weighted recall (the only exception is the zero-shot experiment with $\mathcal{D}_{ZBW}$).

For zero-shot learning, the model performs poorly with both datasets. It achieves a classification accuracy of 9.67% for the UNSW Testing dataset and 13.63% for the ZBW Testing dataset. However, the weighted F1 score for the ZBW Testing dataset in the zero-shot experiment (21.82%) is more than four times higher than that for the UNSW Testing dataset (5.33%), suggesting that the model knowledge base might contain more information about the types of devices present in our lab than those present in the UNSW Testing dataset.

In prompt-tuning, the F1 and accuracy scores for $\mathcal{D}_{ZBW}$ in the prompt-tuning experiments are significantly higher than those in $\mathcal{D}_{UNSW}$. The average prompt-tuning F1 score for $\mathcal{D}_{ZBW}$ (80.47%) is almost

Table 5: LLM-Based UNSW Dataset Classification Results via Prompting.

		UNSW											
		Zero-shot			3 Ex/class			5 Ex/class			7 Ex/class		
	Support	Precision	F1	Recall	Precision	F1	Recall	Precision	F1	Recall	Precision	F1	Recall
Bulb	12	2.56	4.44	16.66	22.22	35.08	83.33	15.90	25	58.33	24	32.43	50
Camera	203	29.09	12.40	7.88	26.82	9.01	5.41	36.66	16.73	10.83	34.28	17.58	11.82
Computer	342	0	0	0	96.70	40.64	25.73	96	13.07	7.01	88.09	19.27	10.81
Mobile	101	5.88	1.69	0.99	31.81	11.38	6.93	29.62	12.5	7.92	26.92	11.02	6.93
Motion Sensor	134	0	0	0	4.59	3.61	2.98	23.75	25.85	28.35	15.89	20.37	28.35
Printer	22	61.53	66.66	72.72	23.75	37.25	86.36	27.94	42.22	86.36	24.39	38.46	90.90
Router	80	8.64	15.60	80	14.41	23.33	61.25	16.41	26.50	68.75	13.95	22.04	52.50
Speaker	79	0	0	0	36.58	25	18.98	40.90	29.26	22.78	64.28	33.64	22.78
Switch	50	0	0	0	12.31	20.85	68	11.15	18.70	58	16.66	26.92	70
Macro Avg	1023	11.96	11.20	19.80	29.91	22.91	39.88	33.15	23.31	38.70	34.27	24.64	38.23
Weighted Avg	1023	8.38	5.33	9.67	46.72	22.96	23.16	51.18	18.76	21.50	48.67	20.53	22.18
Total Accuracy (%)		9.67			23.16			21.50			22.18		

Table 6: LLM-Based ZBW Dataset Classification Results via Prompting.

		ZBW											
		Zero-shot			3 Ex/class			5 Ex/class			7 Ex/class		
Class	Support	Precision	F1	Recall	Precision	F1	Recall	Precision	F1	Recall	Precision	F1	Recall
Bulb	1	33.33	50	100	14.28	25	100	16.66	28.57	100	33.33	50	100
Camera	115	95.83	33.09	20	68.75	16.79	9.5	85.71	18.60	10.43	42.85	4.91	2.60
Computer	0	0	0	0	0	0	0	0	0	0	0	0	0
Mobile	23	11.11	6.25	4.3	6.66	5.2	4.34	9.5	9.09	8.69	0	0	0
Motion Sensor	0	0	0	0	0	0	0	0	0	0	0	0	0
Printer	0	0	0	0	0	0	0	0	0	0	0	0	0
Router	860	51.62	20.65	12.90	95.35	91.66	88.25	95.13	93.04	91.04	95.09	92.53	90.11
Speaker	9	100	36.36	22.22	13.33	16.66	22.22	27.27	30	33.33	42.85	37.50	33.33
Switch	4	0	0	0	0.97	1.86	25	4.34	8.21	75	2.66	5.06	50
Macro Avg	1012	32.43	16.26	17.71	22.15	17.47	27.71	26.51	20.83	35.39	24.08	21.11	30.67
Weighted Avg	1012	55.93	21.82	13.63	89.13	80.10	76.58	91.08	81.72	79.44	86.10	79.60	77.47
Total Accuracy (%)		13.63			76.58			79.44			77.47		

four times higher than that for $\mathcal{D}_{UNSW}$ (20.75%). One of our features is a manufacturer identifier, which we believe the model attempts to make a contextual connection with the class label. As $\mathcal{D}_{UNSW}$ is larger in distribution (therefore more manufacturers) than the $\mathcal{D}_{ZBW}$, the performance for prompt tuning is more degraded for $\mathcal{D}_{UNSW}$ than $\mathcal{D}_{ZBW}$.

As seen in Table 7, fine-tuning significantly improves the performance for $\mathcal{D}_{UNSW}$, as the fine-tuning F1 score for that dataset (64.59%) is almost three times higher than the highest prompt-tuning F1 score of 22.96%, with three examples. The model receives the same examples in the fine-tuning experiment and the prompt-tuning experiment with five examples per class, suggesting that given the same training data, fine-tuning produces better results for performing classification on $\mathcal{D}_{UNSW}$ flows. However, fine-tuning has the exact opposite effect on $\mathcal{D}_{ZBW}$. The average F1 score for $\mathcal{D}_{ZBW}$ drops from 80.47% (for the prompt-tuning experiments) to 2.10% for the fine-tuning experiment (Table 8).

Given the performance seen across the tables, specifically the fine-tuning task for $\mathcal{D}_{UNSW}$, the LLM is capable of identifying IoT devices using link-level features.

6.3 RQ 3: LLM Task Viability

Comparing the results between the traditional algorithms and GPT 3.5, GPT 3.5 outperforms the traditional algorithms for the smaller $\mathcal{D}_{ZBW}$. For instance, with only five examples per class and prompt tuning (Table 6) it reached 79.44% compared to the weaker 10.77% of the RF model. Even for the individual classes in both fine-tuning and prompting tasks, it is able to classify more compared to the many zeros seen for the traditional algorithms. Additionally, fine-tuning with five examples per class outperforms traditional algorithms for the same dataset with 63.73% compared to 52.49% for the RF model on $\mathcal{D}_{UNSW}$. These results imply feasibility of device classification by LLMs compared to traditional approaches and suggest stronger viability with a larger training set.

7 Conclusion and Future Work

In this work, we utilize methods to classify IoT devices using link-level flows across three networking protocols with various models. This task is performed for both unencrypted and encrypted traffic using both traditional ML algorithms and newer, more powerful, LLMs using prompting and fine-tuning techniques.

We successfully classify devices on these three networking protocols; using prompt tuning, the LLM

Table 7: Fine-Tuning $\mathcal{D}_{UNSW}$ with Five Examples Per Class.

	Fine-Tuning UNSW			
Class	Support	Precision (%)	F1 (%)	Recall (%)
Bulb	12	100	100	100
Camera	203	46.87	54.98	66.50
Computer	342	98.57	75.22	80.81
Mobile	101	59.00	72.51	94.05
Motion Sensor	134	91.66	15.06	8.20
Printer	22	95.65	97.77	100
Router	80	96.72	83.68	73.75
Speaker	70	100	93.95	88.60
Switch	50	33.89	47.61	80
Macro Avg	1023	72.23	64.08	67.19
Weighted Avg	1023	80.26	64.59	63.73
Total Accuracy	63.73			

Table 8: Fine-Tuning $\mathcal{D}_{ZBW}$ with Five Examples Per Class.

	Fine-Tuning ZBW			
Class	Support	Precision (%)	F1 (%)	Recall (%)
Bulb	1	100	100	100
Camera	115	33.33	1.69	0.86
Computer	0	0	0	0
Mobile	23	90.90	58.82	43.47
Motion Sensor	0	0	0	0
Printer	0	0	0	0
Router	860	100	0.46	0.23
Speaker	9	5.14	9.65	77.77
Switch	4	0	0	0
Macro Avg	1012	36.58	18.95	24.70
Weighted Avg	1012	90.97	2.10	2.07
Total Accuracy	2.07			

successfully classified devices on a smaller dataset reaching an accuracy of 79.44%. Through fine tuning, the LLM successfully classified devices on a larger dataset reaching an accuracy of 63.73%. The F1 score for these two instances are also in-line with them, differing by no more than two. The LLM outperforms the traditional models with the same data distribution.

For future work, we plan to transition to using LLMs entirely for this task by generating wider datasets and further incorporating the knowledge embedded in pre-trained language models. Furthermore, we plan to evaluate this and similar tasks with other LLMs (e.g., Llama 2) as the results of this work imply feasibility in the use of LLMs for network classification tasks, thus opening potential avenues for their use in network security and privacy.

REFERENCES

Allifah, N. M. and Zualkernan, I. A. (2022). Ranking security of iot-based smart home consumer devices. *Ieee Access*, 10:18352–18369.

Banerji, S. and SinghaChowdhury, R. (2013). Wi-fi & wi-max: A comparative study. *CoRR*, abs/1302.2247.

Chang, T. A. and Bergen, B. K. (2022). Word Acquisition in Neural Language Models. *Transactions of the Association for Computational Linguistics*, 10:1–16.

Chataut, R., Phoummalayvane, A., and Akl, R. (2023). Unleashing the power of iot: A comprehensive review of iot applications and future prospects in healthcare, agriculture, smart homes, smart cities, and industry 4.0. *Sensors*, 23(16):7194.

Danbatta, S. J. and Varol, A. (2019). Comparison of zigbee, z-wave, wi-fi, and bluetooth wireless technologies used in home automation. In *2019 7th International Symposium on Digital Forensics and Security (ISDFS)*, pages 1–5.

Draper-Gil., G., Lashkari., A. H., Mamun., M. S. I., and A. Ghorbani., A. (2016). Characterization of encrypted and vpn traffic using time-related features. In *Proceedings of the 2nd International Conference on Information Systems Security and Privacy - ICISSP,*, pages 407–414. INSTICC, SciTePress.

Ergen, S. C. (2004). Zigbee/ieee 802.15. 4 summary. *UC Berkeley, September*, 10(17):11.

Habibi Lashkari, A., Draper Gil, G., Mamun, M., and Ghorbani, A. (2017). Characterization of tor traffic using time based features. pages 253–262.

Hamdaoui, B. and Elmaghbub, A. (2022). Deep-learning-based device fingerprinting for increased lora-iot security: Sensitivity to network deployment changes. *IEEE network*, 36(3):204–210.

Hao, Q. and Rong, Z. (2023). Iottfid: An incremental iot device identification model based on traffic fingerprint. *IEEE Access*.

Hayes, M. (2018). What is a network traffic flow?

Hegselmann, S., Buendia, A., Lang, H., Agrawal, M., Jiang, X., and Sontag, D. (2023). Tabllm: Few-shot classification of tabular data with large language models. In *International Conference on Artificial Intelligence and Statistics*, pages 5549–5581. PMLR.

Horyachyy, O. (2017). *Comparison of Wireless Communication Technologies used in a Smart Home: Analysis of wireless sensor node based on Arduino in home automation scenario*. PhD thesis.

Hu, T., Dubois, D. J., and Choffnes, D. (2023). Behaviot: Measuring smart home iot behavior using network-inferred behavior models. In *Proceedings of the 2023 ACM on Internet Measurement Conference*, pages 421–436.

Kandpal, N., Wallace, E., and Raffel, C. (2022). Deduplicating training data mitigates privacy risks in language models. In *International Conference on Machine Learning*, pages 10697–10707. PMLR.

Kumar, S., Dalal, S., and Dixit, V. (2014). The osi model: Overview on the seven layers of computer networks. *International Journal of Computer Science and Information Technology Research*, 2(3):461–466.

Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., Stoyanov, V., and Zettlemoyer, L. (2020). Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880.

Li, Q., Yu, K., Chen, D., Sha, M., and Cheng, L. (2022). Trafficspy: Disaggregating vpn-encrypted iot network traffic for user privacy inference. In *2022 IEEE Conference on Communications and Network Security (CNS)*, pages 145–153. IEEE.

Morales, G. A., Bienek-Parrish, A., Jenkins, P., and Slavin, R. (2023). Protocol-agnostic iot device classification on encrypted traffic using link-level flows. In *Proceedings of Cyber-Physical Systems and Internet of Things Week 2023*, pages 19–24.

Ozkaya, I. (2023). Application of large language models to software engineering tasks: Opportunities, risks, and implications. *IEEE Software*, 40(3):4–8.

Pearce, H., Tan, B., Ahmad, B., Karri, R., and Dolan-Gavitt, B. (2023). Examining zero-shot vulnerability repair with large language models. In *2023 IEEE Symposium on Security and Privacy (SP)*, pages 2339–2356. IEEE.

Ramya, C. M., Shanmugaraj, M., and Prabakaran, R. (2011). Study on zigbee technology. In *2011 3rd International Conference on Electronics Computer Technology*, volume 6, pages 297–301. IEEE.

Roumeliotis, K. I. and Tselikas, N. D. (2023). Chatgpt and open-ai models: A preliminary review. *Future Internet*, 15(6):192.

Safaric, S. and Malaric, K. (2006). Zigbee wireless standard. In *Proceedings ELMAR 2006*, pages 259–262. IEEE.

Stolojescu-Crisan, C., Crisan, C., and Butunoi, B.-P. (2021). An iot-based smart home automation system. *Sensors*, 21(11).

Suresh, P. (2016). Survey on seven layered architecture of osi model. *International Journal of research in computer applications and robotics*, 4(8):1–10.

Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., et al. (2023). Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Tripathi, A., Sindhwani, N., Anand, R., and Dahiya, A. (2022). Role of iot in smart homes and smart cities: Challenges, benefits, and applications. In *IoT Based Smart Applications*, pages 199–217. Springer.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 30.

Wang, Z., Liu, D., Sun, Y., Pang, X., Sun, P., Lin, F., Lui, J. C., and Ren, K. (2022). A survey on iot-enabled home automation systems: Attacks and defenses. *IEEE Communications Surveys & Tutorials*.

Wukkadada, B., Wankhede, K., Nambiar, R., and Nair, A. (2018). Comparison with http and mqtt in internet of things (iot). In *2018 International Conference on Inventive Research in Computing Applications (ICIRCA)*, pages 249–253.

Zahan, H., Al Azad, M. W., Ali, I., and Mastorakis, S. (2023). Iot-ad: A framework to detect anomalies among interconnected iot devices. *IEEE Internet of Things Journal*.

Zhang, J., Li, X., Li, J., Dai, Q., Ling, Z., and Yang, M. (2023). Bluetooth low energy device identification based on link layer broadcast packet fingerprinting. *Tsinghua Science and Technology*, 28(5):1–11.