IDS-Agent: An LLM Agent for Explainable Intrusion Detection in IoT Networks

Yanjie Li *
University of Illinois Urbana-Champaign
yjl.zju@gmail.com

Zhen Xiang
University of Georgia
zhen.xiang.lance@gmail.com

Nathaniel D. Bastian
United States Military Academy
nathaniel.bastian@westpoint.edu

Dawn Song
University of California Berkeley
dawnsong@cs.berkeley.edu

Bo Li
University of Chicago
bol@uchicago.edu

Abstract

Emerging threats to IoT networks have accelerated the development of intrusion detection systems (IDSs), characterized by a shift from traditional approaches based on attack signatures or anomaly detection to approaches based on machine learning (ML). However, current ML-based IDSs often fail to explicitly integrate domain knowledge, lack explainability, and struggle to address zero-day attacks. In this paper, we propose IDS-Agent, the first AI agent powered by large language models (LLMs) for intrusion detection. IDS-Agent predicts whether an input network traffic ios benign or malicious, with an explanation of the prediction results. The workflow of IDS-Agent involves a sequence of actions generated by its core LLM based on reasoning over the state observations. The action space of IDS-Agent includes data extraction and preprocessing, classification, knowledge, and memory retrieval, and results aggregation – these actions will be executed using abundant tools, mostly specialized for IDS. Furthermore, IDS-Agent is equipped with a memory and knowledge base that retains information from current and previous sessions, along with IDS-related documents, enhancing its reasoning and action generation capabilities. The system prompts of IDS-Agent can be easily customized to adjust detection sensitivity or identify previously unknown types of attacks. In our experiments, we demonstrate the strong detection capabilities of IDS-Agent compared with ML-based IDSs and an IDS based on LLM with prompt engineering. IDS-Agent outperforms these SOTA baselines on the ACI-IoT and CIC-IoT benchmarks, with 0.97 and 0.75 detection F1 scores, respectively. IDS-Agent also achieves a recall of 0.61 in detecting zero-day attacks, outperforming previous approaches specially designed for this task.

1 Introduction

In recent years, the Internet of Things (IoT) has emerged as a transformative technology, increasingly adopted across a wide range of applications [8]. Alongside its rapid development, security

*This work was done during the author's internship at University of Illinois Urbana-Champaign.

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

concerns have arisen within IoT networks due to the typically large number of devices with potential trustworthiness issues [3]. Therefore, the deployment of Intrusion Detection Systems (IDSs) has become essential, as they play a critical role in monitoring network traffic and identifying malicious activities [19]. Many traditional IDSs employ signature-based methods, which rely on databases that store signatures of known attacks [16, 26, 4]. They suffer from high false negative rates when variations in attack methodologies do not exactly match the existing signatures. Alternatively, IDS can also be designed as an anomaly detector to identify distributional deviations from normal traffic [18, 17]. However, defining the normal behavior in a network can be challenging, especially in diverse and dynamic environments where normal activity can change over time. Machine learning (ML)-based IDS was then proposed to address these problems by leveraging the representation power of ML models, such as deep neural networks (DNNs), to capture complex attack patterns from extensive training data [23]. However, ML-based IDS still face limitations due to constraints in the model architecture and training data, which hinder their detection capabilities, especially when addressing zero-day attacks [31]. Furthermore, the detection results from ML-based IDS often lack clear explanations, which can diminish their credibility, particularly in safety-critical IoT scenarios where understanding the rationale behind alerts is crucial [7, 2].

Recently, AI agents empowered by large language models (LLMs) have been created to handle complicated tasks in various application domains [37, 1, 29, 21, 10, 15, 40, 34]. These agents are characterized by their integrated capabilities to interact with the environment, conduct knowledge-assisted reasoning, and then take appropriate actions according to the user's requests. Typically, LLM agents are equipped with a knowledge/memory base for the retrieval of task-related information and a toolbox that includes tools such as local functions and third-party APIs. They utilize one or more LLMs for reasoning and the subsequent generation of actions, such as the selection of the most appropriate tools. The intelligence of LLM agents in reasoning (as well as analysis and criticism [30]) makes them both powerful operators and effective intermediaries between task execution and the environment/users.

In this paper, we propose `IDS-Agent`, the first LLM agent designed for intrusion detection, featuring its integration of external knowledge, and the capabilities for explanation, customization, and adaptation to zero-day attacks. `IDS-Agent` takes an input request for intrusion detection with a target raw data flow, and outputs detection results with a detailed explanation. The agent adopts a reasoning-followed-by-action pipeline ([36]) with a specialized action space for intrusion detection. Specifically, knowledge-enabled reasoning based on long-term memory from previous sessions is performed by the core LLM of the agent to decide the optimal tools (and their settings/parameters) for data extraction, preprocessing, classification, and results aggregation. Compared with existing ML-based IDSs, `IDS-Agent` achieves a stronger detection performance and better interpretability by harnessing the power of multiple ML models and external knowledge in a comprehensive way. It aggregates the classification results from multiple ML models by prompting an LLM with the top-k label predictions and the confidence scores for each model. The prompt also includes external knowledge (e.g. regarding particular attack types) obtained by calling a search engine and additional instructions, for example, to customize detection sensitivity or to reveal unseen attack types. The LLM is instructed to produce structured outputs, including detection results and an explanation. Our main technical contributions are summarized below:

- We propose `IDS-Agent`, the first LLM-powered agent for intrusion detection, featured by its capabilities of explicit knowledge integration, explanation, detection customization, and revealing of zero-day attacks.
- We propose a reasoning-followed-by-action pipeline for `IDS-Agent` with an action space and toolbox specialized for network traffic processing, knowledge retrieval and integration, and intrusion detection results aggregation.
- We demonstrate the effectiveness of `IDS-Agent` on two IDS benchmarks, ACI-IoT'23 and CIC-IoT'23. `IDS-Agent` achieves higher detection accuracy compared with the latest LLM-based method, various ML models, and the ensemble of ML models based on majority vote.
- Experiments show that when classifiers produce discrepancy predictions, `IDS-Agent` can utilize inherent and external knowledge to help the decision-making. `IDS-Agent` also demonstrates clearly better performance than existing approaches in detecting zero-day attacks. Moreover, we find that `IDS-Agent` effectively follows the sensitivity instructions without requiring expert intervention or additional tuning.

2 Related Work

Conventional IDS An Intrusion Detection System (IDS) is designed to detect malicious activities on computer systems, helping to ensure system security [19]. IDSs are generally classified into two main types: Signature-based Intrusion Detection Systems (SIDS) and Anomaly-based Intrusion Detection Systems (AIDS). SIDS relies on pattern-matching techniques to identify known attacks [16, 26, 4]. However, the rise in zero-day attacks has increasingly diminished the effectiveness of SIDS, as these new attacks lack existing signatures [31]. In contrast, AIDS constructs a model of normal system behavior using machine learning, statistical, or knowledge-based techniques. Any significant deviation from the constructed model is flagged as an anomaly, potentially indicating an intrusion [18, 17]. However, in a dynamically changing environment, a time-consuming regular update on the knowledge base is needed.

ML-based IDS Many machine learning models, such as MLP [5], KNN [22], Decision Tree [13], SVM [25], have been explored for anomaly-based intrusion detection. For IoT intrusion detection, Verma et al. [32] conducted a comprehensive comparison of ensemble and individual classifiers, including Random Forest (RF), AdaBoost (AB), and Gradient Boosted Machines (GBM). Roy et al. [28] proposed a lightweight IDS model utilizing machine learning to detect cyber-attacks and anomalies in resource-constrained IoT systems. Davis et al. [9] advanced this line of research by applying a quantum-annealing approach for feature selection in IoT intrusion detection. Compared with ML-based IDSs, our `IDS-Agent` not only achieves better empirical performance but also provides a detailed explanation of each intrusion detection result.

LLM-based IDS Large language models (LLMs), especially generative pre-trained commercial transformers, like GPTs, have recently demonstrated outstanding ability in information comprehension and reasoning tasks. This has motivated some studies in applying LLMs to abnormal detection tasks, such as compiler optimization [12] and software vulnerability detection [14]. Zhang et al. [38] is the first to use LLMs for IDS by employing a straightforward in-context learning approach with GPT-4, which provides it with a few labeled examples. Their method achieved over 90% accuracy on a simple dataset containing only five types of attacks. However, in this paper, we demonstrate that the performance of their method drops significantly when tested on more complex and diverse datasets. Different from this LLM-based IDS, our `IDS-Agent` uses LLM for reasoning and action planning, with integration of external knowledge and tools, leading to a huge performance gain on diverse datasets.

3 Method

`IDS-Agent` is designed to produce a prediction result with an explanation for each user request for IoT traffic inference, i.e., to determine if the traffic is benign or belongs to any particular attack type. We also allow `IDS-Agent` to handle requests for customized detection sensitivities or to detect new attack categories from the given IoT traffic flow. `IDS-Agent` is equipped with a) an abundant toolbox containing special IDS tools such as ML models for classification and general tools such as search engines to retrieve external knowledge and b) a memory and knowledge base storing the current session information, long-term memory from previous sessions, and supportive documents. These tools, memory, and external knowledge will be integrated to guide the decision-making of `IDS-Agent` in a structured manner, as detailed in the sequel.

3.1 Pipeline of `IDS-Agent`

The pipeline of our `IDS-Agent` is inspired by the ReAct agent [36]. The user request is fulfilled by executing a sequence of action steps $\{a_1, a_2, \cdots\}$, where each action step is generated by a core LLM based on previous reasoning and observations. For any input user request for intrusion detection and a traffic flow to be inspected, an initial observation o_0 is constructed by concatenating the user request with a system prompt, including a description for each available tool. This initial observation serves as the context for the agent to understand the task, facilitating subsequent reasoning and action generation.

Specifically, `IDS-Agent` iterates over the following three steps:
1) Reasoning: The core LLM generates a thought (in plain text) about the next action by $r_i = \text{LLM}(s_i)$, where $s_i = \{o_0, \{r_1, a_1, o_1\}, \cdots, \{r_{i-1}, a_{i-1}, o_{i-1}\}\}$ is the short-term memory of the current session up to the $(i-1)$-th iteration (with $s_1 = \{o_0\}$). Reasoning can optionally adopt

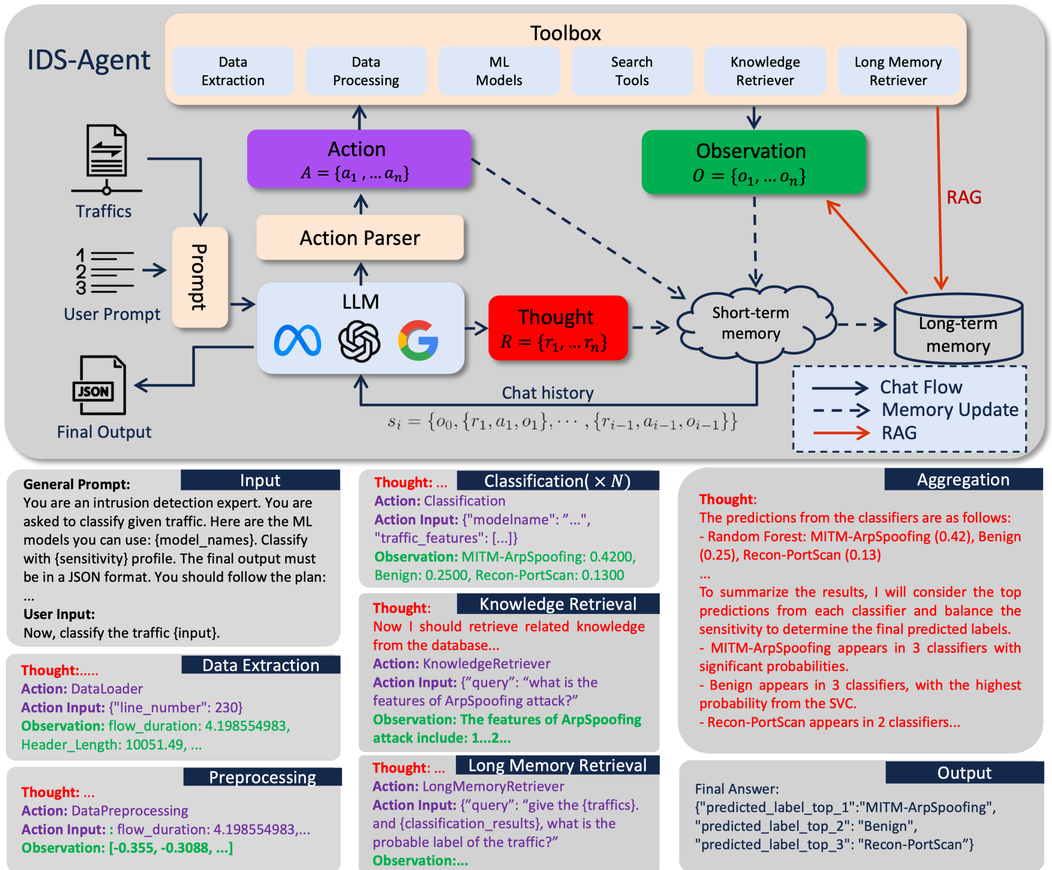


Figure 1: **Top** (framework of `IDS-Agent`): `IDS-Agent` adopts a core LLM to generate thoughts (i.e. reasoning) and actions based on input traffic and user prompt. It is equipped with a toolbox for action execution and a memory base for knowledge retrieval. `IDS-Agent` iteratively conducts thought generation, action generation and execution, and observation update. **Bottom** (an example reasoning trace): in this example, several classifiers are adopted by the agent, with same number of classifiers predicting the input traffic as 'Benign' and 'MITM-ArpSpoofing'. Based on this observation, `IDS-Agent` decides to perform 'Knowledge Retrieval' and 'Memory Retrival', and finally aggregate these observations, which leads to correct attack detection.

long-term memory from previous sessions for in-context demonstration.
2) Action generation: The action is generated based on the reasoning/thought by $a_i = \text{LLM}(r_i, s_i)$. Notably, each action we generate is a structured JSON file containing an *action name* and an *action input*. The action input consists of the name of the tool(s) to be used and the associated settings or parameters of each tool. Such a structured generation of the action allows its efficient and accurate execution using the specified tools. For example, if the action is to adopt a classifier, the action name will be "Classification" and the action input will be the model name and the associated settings.
3) Observation update: After executing the generated action a_i, we obtain a new observation o_i by converting the outputs of the tool(s) into plain text. For example, the the observation after applying a classifier will be the top-k labels and their corresponding prediction confidences.
The iterations terminate when the observation is updated by a 'final answer' headline followed by a JSON file. This JSON file, which encapsulates the final prediction on the traffic data and related analysis and explanation, will be the output of `IDS-Agent`.

3.2 Action Space and Tool Design

Our `IDS-Agent` is designed with a comprehensive action space, allowing it to handle various tasks in the pipeline of data processing and classification through iterative reasoning and execution. The action space includes the key actions described below.

Data Extraction: The goal is to accurately extract network traffic records x specified in the user request from the data flow (a dataset in our experiments) for further analysis. We design the data extraction tool as a function that takes the given flow ID (or line number for stored traffic dataset) as the input and outputs a structured traffic sample.

Preprocessing: The goal is to clean, normalize, and transform the extracted data into a standard format for subsequence processing, especially for classification. Our preprocessing tools are designed as functions for diverse data analysis operations, including feature scaling, data encoding, handling missing values, and selecting important features for classification.

Classification: This action applies machine learning models to the preprocessed data to obtain classification results, i.e., to predict whether the traffic is benign or falling into any malicious category. The inputs to the classification tool include the preprocessed traffic features and a classifier, while the outputs include the top-k labels and their corresponding confidence scores. Note that our classification tool is not merely an ML model; it advances ML-based IDS by adaptively considering diverse model types and incorporating more information from the classification results into the inference procedure. The types of classifiers used by `IDS-Agent` include Random Forest, SVM, MLP, Decision Tree, KNN, etc. It is important to note that our classification toolbox is extensible. In real-world deployments, users can easily add new classifiers to the toolbox without the need for any LLM fine-tuning. This flexibility allows for easy updates and adaptation to new attack patterns or changes in the network environment. Moreover, the classifiers can also be open-sourced models trained by third parties (callable through APIs), or models locally trained based on the data collected by the user.

Knowledge Retrieval: This action aims to obtain knowledge regarding the particular types of attacks predicted by the classifiers. The knowledge can be external that is retrieved by calling search engines such as Google and Wikipedia API, or stored locally in the knowledge base. The retrieval from the local knowledge follows the RAG approach [20]. The retrieved knowledge will be used to guide the aggregation of classification results from the individual ML models.

Long-Term Memory Retrieval: Long-Term Memory carries information from previous sessions (will be discussed in Sec. 3.3) that can inform the decision making of `IDS-Agent` in the current session. The retrieval of long-term memory can be activated optionally by adding a specific instruction in the system prompt in o_0.

Aggregation: This action aims to comprehensively integrate the results from multiple steps of classification action (based on different classifiers) to generate a structured final inference decision. The core of the aggregation tool is an LLM where the prompt is designed to include 1) the detailed results from the classifiers, 2) the short-term memory, and 3) demonstrative inputs and outputs aggregated by the LLM from previous sessions. Note that the short-term memory also includes the external knowledge previously extracted and the system prompts in o_0. This system prompt, as shown in Fig. 3, includes a specification for the detection sensitivity; it can also include an instruction for revealing new attack categories. Compared to naive aggregation, such as majority voting, our method incorporates more information to resolve any discrepancies between different model outputs in a more comprehensive way.

3.3 Memory and Knowledge Base

The memory and knowledge base of `IDS-Agent` stores 1) the short-term memory, 2) the long-term memory, and 3) supportive documents for IDS from the external. **Short-term Memory (STR).** The STR, as described in Sec. 3.1, includes the historical reasoning trace, actions, and observations of the current session in a structured format, and is renewed after each observation update. The major goal of the STM here is to track the agent's iterative reasoning process and ensure consistency between steps in real-time.

Long-term Memory (LTM). The LTM consists of agent decisions and contextual information from previous use cases [33, 41]. Here, a structured LTM example is defined by $\phi = \{t, x, R, A, O, \hat{y}\}$, where t is a timestamp, $\boldsymbol{x}$ is the feature vector after preprocessing, $R = [r_1, \cdots, r_n]$ is the reasoning trace, $A = [a_1, \cdots, a_n]$ contains all the action steps, $O = [o_1, \cdots, o_n]$ are the observations, and $\hat{y}$ is the final label prediction. The long-term memory base can be initialized by running the agent on a validation dataset. During the inference, only sessions with correct agent decisions will be stored in the long-term memory base. In a real-world intrusion detection scenario, such correctness can be validated by human experts.

In our framework, LTM retrieval provides the agent with additional information while aggregating the results for individual classifiers. Here, we set the LTM retriever input as the current timestamp t and observations from previous data processing and classification actions, denoted by $\tilde{O} = [o_1, \ldots, o_{m-1}]$, where m is an arbitrary iteration where the LTM retrieval kicks in. The retriever obtains the top-k relevant final reasoning based on the weighted sum of timestamp distance and the cosine similarity between the embedding of $\tilde{O}$ and the observation embeddings $O^{(j)}$ of previous LTM examples $\{\phi^{(1)}, \cdots, \phi^{(L)}\}$. Specifically, we obtain the top-k solutions to

$$\mathrm{argmax}_j [\lambda_1 r(t, t^{(j)}) + \lambda_2 \mathrm{cosim}(\mathcal{E}(\tilde{\mathrm{O}}), \mathcal{E}(\mathrm{O}^{(\mathrm{j})}))], \tag{1}$$

where $\mathcal{E}(\cdot)$ is the encoder. $r(t, t^{(j)}) = 1 - (t - t^{(j)}) / \max_k (t - t^{(k)})$ is the recency of the memory. The equation ensures that both recent observations and content-wise similar observations are considered to address the evolving nature of intrusion data. Then, the observation o_m for the iteration m contains the input-prediction pairs $(x^{(j)}, \hat{y}^{(j)})$ of each retrieved $\phi^{(j)}$. In our experiments, we set $k = 5$ to retrieve the top-5 relevant structured LTM examples.

Supportive Documents from the External. In addition to the external knowledge obtained by calling search engines, such as Google and Wikipedia, `IDS-Agent` is also equipped with a vector database $\{\psi^{(1)}, \cdots, \psi^{(K)}\}$ containing related research papers and intrusion detection blogs (both parsed into chunks with fixed token length). The retrieval from this knowledge base is similar to the retrieval of LTM. We obtain the top-k solutions to $\mathrm{argmax}_j \mathrm{cosim}(\mathcal{E}(q), \mathcal{E}(\psi^{(j)}))$, where q is the query generated by the core LLM (based on the reasoning) as the action input to action step a_m for an arbitrary iteration m for knowledge retrieval. The retrieved document chunks are summarized (for compression) using an LLM (may be the same as the core LLM or an independent LLM) and are used to update the observation o_m. The definition, characteristics, and detection methods for various attacks recorded in the retrieved chunks will facilitate `IDS-Agent` to better understand the potential risks while aggregating the classification results for the ML models. For example, if an attack type can potentially lead to catastrophic results, `IDS-Agent` will be more sensitive to it when any classifier makes such a prediction.

4 Experiments

4.1 Experiment Settings

Dataset. This paper focuses on intrusion detection in the IoT environment, which presents more complexities and challenges than traditional networks. We consider the following two datasets commonly used in previous works.
1) ACI-IoT'23 [6]: This dataset contains both benign and malicious network traffic captured from a variety of IoT devices. The dataset includes simulations of several attack types, such as Reconnaissance (e.g., Host Discovery, OS Scan, Ping Sweep, and Port Scan), DoS (e.g., ICMP Flood, SYN Flood, UDP Flood, and Slowloris) and Brute Force (e.g., Dictionary Attacks). We randomly select 10% of the data to train the ML-models for classification. For evaluation, we construct a test dataset from the remaining samples in ACI-IoT'23 by randomly selecting 200 benign samples and 20 samples per attack category.
2) CIC-IoT 2023 [27]: This dataset simulates large-scale, real-time IoT environments and comprises 33 distinct types of attacks, which is even more difficult for intrusion detection than ACI-IoT'23. The dataset includes network traffic from a broad IoT topology with 105 devices. From the dataset, we selected the 24 most common attack types along with benign samples to create our training and testing datasets. The remaining 9 attack types were excluded from the training data and designated as unknown attacks for evaluating zero-day attack detection performance. Again, we use 10% of the data for training the machine learning classifiers. The test dataset is constructed with 100 benign samples and 10 samples for each attack type.

Evaluation Metrics. We are interested in the performance of `IDS-Agent` in both the binary classification of benign and malicious flows and the multi-classification that also requires recognizing the specific attack type when a flow is deemed malicious. For binary classification, we use *accuracy*, and *false alarm rate* (FAR) as the evaluation metrics. Accuracy is the ratio of correctly predicted samples over the total number of samples in the dataset, measuring the overall effectiveness of the `IDS-Agent` in detecting both benign and malicious flows. FAR measures the proportion of false positives (benign flow incorrectly classified as malicious).

For *multi-classification*, we treat each attack category (and also the benign category) as a class. We use the per-class *precision*, *recall*, and *F1-score* as the evaluation metrics. Detailed results for the detection of each attack category are deferred to the appendix due to space limitations. In the main paper, we report the *macro-averaged* precision, recall, and F1-score across all classes as the overall performance of the IDS being evaluated. This macro-average is computed by averaging the metric over all classes with equal weights.

4.2 Implementation Details

The core LLM. We consider three LLM choices: GPT-3.5-Turbo, GPT-4o-mini, and GPT-4o.

Tool Design. 1) **Data preprocessing tool**. We implement Python functions that will be sequentially called for data preprocessing. First, we remove from each flow the fields irrelevant to the traffic features, including the label, time-stamp, and flow ID. Second, we encode the non-numeral fields into numbers, including the connection type and protocol type. Third, we conduct feature selection based on an F-test for linear dependency between features and labels. Finally, the extracted features are standardized. 2) **Classification tool**. The core of the classification tool is an ML model for intrusion detection, which can be self-trained or an off-the-shelf model trained by a third party. Here, we pretrained six (multi-)classifiers, including random forest (RF), K-Nearest Neighbors (KNN), logistic regression (LR), decision tree (DT), multi-layer perceptrons (MLP), and support vector classifier (SVC) using the training set from our benchmarks. The output of the classification tool is the top-3 label predictions with their confidence scores. 3) **Knowledge retrieval tool**. We construct a knowledge base for various IoT attacks by collecting 50 online blogs and 50 research papers. These documents are then split into chunks of 1000 tokens with an overlap of 200 tokens. These chunks were embedded using the OpenAI encoder and stored in a vector database powered by ChromaDB.

4.3 Baselines

1) **Machine learning methods.** We compare `IDS-Agent` with the state-of-the-art ML-based IDS, which uses a quantum-annealing method for feature selection [9]. In Appendix A.3, we also compare our `IDS-Agent` with six IDSs using the six ML models we pretrained, respectively.

2) **LLM-based methods**. We compared our `IDS-Agent` with the latest GPT-4-based intrusion detection approach [38], which leverages the model's reasoning capabilities and in-context demonstrations. The authors particularly demonstrate that providing GPT-4 with a few labeled examples can improve the accuracy of intrusion detection. We further improve the performance of this baseline by first clustering their in-context examples using a Gaussian Mixture Model (GMM) and then selecting in-context examples from different clusters to cover as many attack cases as possible. Compared to this baseline with a fixed set of demonstrations, `IDS-Agent` retrieves LTM for demonstration dynamically based on input similarity.

3) **Ensemble learning methods (majority vote).** We create a strong baseline by ensemble the results from the six ML models we pretrained through majority voting.

4.4 Experiment Results

Quantitative Analysis. The quantitative results on the ACI-IoT'23 dataset are summarized in Table 1. Our `IDS-Agent` achieves the best general performance when GPT-4o is used as the core LLM. Comparable performance is achieved when GPT-4o-mini is used as the core LLM, which is over 60% more cost-effective than GPT-3.5 Turbo. `IDS-Agent` also shows a clearly better recall than the baselines. The detailed results for `IDS-Agent` in detecting each attack category of ACI-IoT'23 compared with the baselines are shown in Appendix A.3. In particular, when detecting UDP flood attacks, `IDS-Agent` achieves a recall of 0.80 compared with 0.20 for the LLM baseline (GPT-4o) and 0.55 for the majority voting baseline. We also found that for some attack types such as DNS-Flood, Slowloris, and Dictionary attacks, `IDS-Agent` demonstrates slightly lower precision than majority voting. This decrease in precision is due to the `IDS-Agent`'s heightened sensitivity to high-threat attacks, leading it to classify a sample as an attack even with less than 50% voting from the classifiers involved.

Table 1: Binary-classification and multi-classification performance of `IDS-Agent` compared with baseline approaches on the ACI-IoT'23 and CIC-IoT'23 datasets. We compare our method with vanilla GPT-4o-based method enhanced by in-context learning [38], a Random Forest classifier that uses a quantum-annealing method for feature selection [9], and a strong baseline of majority voting by six ML classifiers.

Dataset	Metric Types	Metrics	GPT-4o	RF	Majority Vote	IDS-Agent (GPT-3.5)	IDS-Agent (GPT-4o-mini)	IDS-Agent (GPT-4o)
ACI-IoT'23	Binary-Class	Binary-Class Accuracy ↑	0.721	0.890	0.960	0.954	0.963	**0.965**
		FAR ↓	0.497	0.060	**0.020**	0.050	0.041	0.030
	Multi-Class	Multi-Class Accuracy ↑	0.678	0.790	**0.980**	0.972	0.976	**0.980**
		Macro Avg Precision ↑	0.682	0.785	0.980	0.971	0.980	**0.982**
		Macro Avg Recall ↑	0.754	0.760	0.961	0.952	0.972	**0.972**
		Macro Avg F-Score ↑	0.682	0.750	0.962	0.923	0.974	**0.975**
CIC-IoT'23	Binary-Class	Binary-Class Accuracy ↑	0.750	0.825	0.882	0.876	0.894	**0.904**
		FAR ↓	0.144	0.050	0.040	0.050	**0.030**	**0.030**
	Multi-Class	Multi-Class Accuracy ↑	0.610	0.751	0.771	0.762	0.788	**0.795**
		Macro Avg Precision ↑	0.580	0.755	0.760	0.759	0.790	**0.800**
		Macro Avg Recall ↑	0.450	0.692	0.700	0.694	0.723	**0.733**
		Macro Avg F-Score ↑	0.510	0.680	0.699	0.700	0.722	**0.750**

It is worth noting that the vanilla GPT-4o-based method [38], despite enhancements through in-context learning, performs unsatisfactorily. Moreover, when there are more attack categories, more in-context demonstrations will be required (to effectively inform all attack categories) for their method, posing a significant challenge due to GPT-4o's limited token input length and the increase in cost. Our method integrates ML models and utilizes RAG to retrieve the most related knowledge from the memories, significantly reducing the token cost.

Table 1 also presents the evaluation results on the CIC-IoT'23 dataset. Powered by both reasoning ability and tool calls, `IDS-Agent` achieves higher accuracy than the LLM baseline and majority voting method. Moreover, `IDS-Agent` achieves high detection accuracy on some very challenging attacks, such as ArpSpoofing and Host Discovery. The detailed results for each attack type are deferred to the Appendix A.3.

Case study While the majority voting baseline shows relatively strong performance, their classification results usually lack interpretability [35]. In contrast, we leverages the reasoning capabilities of LLMs to enhance interpretability which helps to improve the detection performance. One example is illustrated on the right of Figure 1 where the `IDS-Agent` concludes that the MITM ARP-Spoofing classification is more likely, as it appears in three classifiers' top-3 predictions with significant confidence in general. Another case for the decision-making is shown in Figure 2, with the ground truth label being 'reconnaissance activities'. Despite 3 out of 6 machine learning models predicting the traffic as benign, `IDS-Agent` accurately labels it as a reconnaissance attack. This decision is rooted in the `IDS-Agent`'s understanding that both Host Discovery and OS Scanning belong to reconnaissance attacks. By considering the relationship between these types of attacks, the `IDS-Agent` demonstrates its ability to detect subtle patterns that individual classifiers may overlook, thus improving the overall detection accuracy. Additional examples are provided in the appendix to demonstrate its decision-making process further.

4.5 Zero-Day Attack

We find that `IDS-Agent` not only excels in detecting known attacks but also shows promise in identifying zero-day threats. Table 2 presents the detection results on the CIC-IoT'23 dataset, comparing `IDS-Agent` with two recent zero-day intrusion detection methods: ACGAN [39] and RealNVP [24]. We reproduced ACGAN using its official code and RealNVP based on the FrEIA framework, training separate normalizing flows for benign and malicious samples. We set the detection threshold of RealNVP at the 90th percentile of the negative log-likelihood from the INN model; samples exceeding this threshold are considered unknown attacks or out-of-distribution (OOD). We selected nine attack types from the CIC-IoT'23 dataset as 'unseen' for evaluation, excluding

Thought: I have obtained the predictions from multiple classifiers. Here are the results:
1. **Random Forest**: - **Recon-HostDiscovery**: 0.2800, MITM-ArpSpoofing: 0.2500, BenignTraffic: 0.2400
2. **K-Nearest Neighbors**: **BenignTraffic**: 0.4000, Recon-HostDiscovery: 0.4000, MITM-ArpSpoofing: 0.2000
3. **Logistic Regression**: **Recon-OSScan**: 0.3909, Recon-PortScan: 0.2442, Recon-HostDiscovery: 0.1493
4. **MLP**: **BenignTraffic**: 0.5555, MITM-ArpSpoofing: 0.3421, Recon-OSScan: 0.0586
5. **Support Vector Classifier**: **BenignTraffic**: 0.6324, Recon-HostDiscovery: 0.2326, Recon-OSScan: 0.0855
6. **Decision Tree**: **Recon-HostDiscovery**: 1.0000, Recon-PortScan: 0.0000, DDoS-UDP_Flood: 0.0000

Now, I will analyze the results to determine the final classification. The predictions show a mix of benign and attack classifications,... I will search for more information regarding the features of the attacks and benign traffic to make a more informed decision.
Action: knowledge_retriever
Action Input: {"query": "what is Recon-HostDiscovery and how to detect it'"
Observation: The provided context does not specifically mention \"Recon-HostDiscovery\" or provide details on what it is or how to detect it. Based on general knowledge, "Recon-HostDiscovery" typically refers to the reconnaissance phase in cybersecurity where an attacker attempts to discover hosts on a network. This phase involves scanning the network to identify active devices, open ports, and services running on those devices...
Final Answer:

```
{
"line_number": 229,
 "analysis": "The traffic features were classified by multiple models with discrepancies. The Random Forest and Decision Tree models indicated Recon-HostDiscovery, while Logistic Regression suggested Recon-OSScan. The K-Nearest Neighbors, MLP, and Support Vector Classifier models leaned towards benign traffic. Given the context of the features and the nature of the predictions, the final classification leans towards reconnaissance activities.",
"predicted_label_top_1": "Recon-HostDiscovery",
"predicted_label_top_2": "Recon-OSScan",
"predicted_label_top_3": "BenignTraffic",
}
```

Figure 2: A case study of IDS-Agent successfully detecting reconnaissance activities. Despite most machine learning models classifying the traffic as benign, IDS-Agent ultimately predicts a reconnaissance attack. This decision is based on its understanding that both Host Discovery and OS Scan belong to reconnaissance activities.

them from training data. Each attack type was evaluated with 50 samples. IDS-Agent's classifiers were trained on data from other known attack categories. Additionally, we enhanced the system prompt with specific instructions to classify ambiguous samples as "Unknown" if most classifiers' confidence was below a certain threshold. We set the threshold as 0.7 in our experiments. The recall for IDS-Agent in predicting an 'unknown attack' from these unseen classes was measured, achieving a top-1 recall of 0.61. Notably, IDS-Agent showed high recall for Vulnerability Scan and SQL Injection attacks, likely due to their distinct deviation from the training data distribution, resulting in low classification confidence. This indicates IDS-Agent's capability to detect unknown attacks, particularly when diverse machine learning models yield divergent results with low confidence or when traffic features are anomalous. The recall for benign examples decreased from 0.91 to 0.86 after introducing the zero-day detection prompt. This drop is attributed to the tendency of the model to classify OOD benign examples as unknown attacks.

Table 2: IDS-Agent outperforms existing approaches specially designed for detecting zero-day attacks, with higher recalls in detecting all ten categories of "unseen" attacks.

Methods	Backdoor	DNS Spoofing	Uploading Attack	XSS	Dictionary BruteForce	Command Injection	VulScan	Browser Hijacking	SQL	Avg Recall
ACGAN	0.38	0.35	0.59	0.32	0.36	0.42	0.88	0.05	0.38	0.41
RealNVP	0.45	0.37	0.68	0.49	0.42	0.45	0.89	0.03	0.45	0.47
IDS-Agent	**0.64**	**0.46**	**0.74**	**0.65**	**0.45**	**0.55**	**0.95**	**0.15**	**0.86**	**0.61**

4.6 Ablation Study

We conduct an ablation study focusing on two key modules: the Knowledge Retrieval Module and the Long-Term Memory Module. We evaluate how each module affects the detection performance on two settings: 1) 'in-distribution' setting with all attack labels known and 2) zero-day attack setting in Section 4.5 with a subset of unknown attacks. We also evaluate the performance under different detection sensitivities.

Effect of Knowledge Retrieval Module The Knowledge Retrieval Module is designed to augment the classifiers' outputs with relevant information from an external knowledge base, enhancing the agent's understanding of complex attack patterns. To assess its impact, we disable this module and compare the performance of the modified agent with the original IDS-Agent. Table 3 summarizes the detection performance with and without the Knowledge Retrieval Module. When the module is removed, we observe significant decrements in the detection accuracy for both settings. Specifically, the recall for detecting zero-day attacks drops from 0.61 to 0.42. This indicates that without access to external knowledge, the agent has a diminished ability to recognize patterns that are not represented in the training data.

Table 3: Importance of knowledge retrieval.

Recall	With KRM	Without KRM
In-Distribution	0.733	0.710
Zero-Day	0.610	0.420

Table 4: Importance of long-term memory.

Recall	With LMM	Without LMM
In-Distribution	0.733	0.702
Zero-Day	0.610	0.560

Effect of Long-Term Memory Module The Long-Term Memory Module allows `IDS-Agent` to maintain a history of previous observations and decisions, which is essential for detecting attacks exhibiting temporal dependencies and utilizing previous success experiences. To evaluate its effect, we disable the Long-Term Memory Module and assess the agent's performance on the same evaluation set. We set the λ_1 and λ_2 as 0.5 in Eq. 1 to balance the recency similarity to retrieve the most relevant past examples from the agent's LTM. As presented in Table 4, the removal of the Long-Term Memory Module leads to a degradation in detection performance, particularly for attacks that unfold over time, such as Brute Force and Distributed Denial of Service (DDoS) attacks. The overall detection accuracy decreases from 0.733 to 0.702, and the recall for zero-day attacks drops to 0.56. The decrease in performance underscores the importance of the Long-Term Memory Module in capturing temporal features and improving the agent's ability to detect attacks that evolve over time. By retaining historical information, `IDS-Agent` can identify suspicious patterns that may not be apparent when considering individual events in isolation.

Detection Sensitivity. Sensitive configuration is a critical function of intrusion detection systems (IDS). In signature-based IDS, experts need to manually adjust detection sensitivity, which can be both costly and time-consuming [11]. In contrast, the detection sensitivity of our `IDS-Agent` can be easily adjusted through input prompts. Here, we can optionally instruct the `IDS-Agent` to operate under three different sensitivity levels: *aggressive*, *balanced*, and *conservative*. This sensitivity level controls the trade-off between the false alarm rate and the missed detection rate. The results are shown in Table 5. For the aggressive detection, the `IDS-Agent` achieves high recall for attacks (0.97) but lower recall for benign examples (0.90). Conversely, conservative detection shows relatively lower recall for attacks (0.85) but higher recall for benign examples (0.98). The complete prompts and detection results are shown in Appendix A.4.

Table 5: The classification results of `IDS-Agent` on the ACI-IoT'23 dataset for three different detection sensitivities. The core LLM of `IDS-Agent` is GPT-4o. Aggressive detection leads to high recall for attacks, while conservative detection results in higher recall for benign examples.

Sensitivity	Aggressive			Balanced			Conservative		
Metrics	Precision	Recall	F1-score	Precision	Recall	F1-score	Precision	Recall	F1-score
Benign	0.96	0.90	0.92	0.87	0.96	0.91	0.60	0.98	0.75
Attack (Macro Avg)	0.97	0.97	0.97	0.98	0.95	0.96	0.95	0.85	0.87

5 Conclusion

In this paper, we propose `IDS-Agent`, the first LLM-powered agent for intrusion detection. We design an iterative reasoning-followed-by-action pipeline for `IDS-Agent` to extract data from the network traffic, preprocess the data, consult different machine learning models for classification results and details, retrieve both internal and external knowledge, and summarize the final detection inference. These agent actions are facilitated by a memory module and a wide array of tools for intrusion detection and general purposes. Empirically, `IDS-Agent` outperforms diverse types of SOTA IDSs on ACI-IoT'23 and CIC-IoT'23. We find that when classifiers produce discrepancy predictions, `IDS-Agent` can utilize inherent and external knowledge to assist decision-making. Moreover, `IDS-Agent` can be easily adapted to detect zero-day attacks, exhibiting better performance than existing methods. Finally, we find that `IDS-Agent` effectively follows the sensitivity instructions in detection without requiring expert intervention or additional tuning.

References

[1] Mahyar Abbasian, Iman Azimi, Amir M. Rahmani, and Ramesh Jain. Conversational health agents: A personalized llm-powered agent framework, 2024.

[2] Tanveer Ahmad and Dongdong Zhang. Using the internet of things in smart energy systems and networks. *Sustainable Cities and Society*, 68:102783, 2021.

[3] Yara Alghofaili and Murad A. Rassam. A trust management model for iot devices and services based on the multi-criteria decision-making approach and deep long short-term memory technique. *Sensors*, 22(2), 2022.

[4] Mutep Y AlYousef and Nabih T Abdelmajeed. Dynamically detecting security threats and updating a signature-based intrusion detection system's database. *Procedia Computer Science*, 159:1507–1516, 2019.

[5] Karan Bajaj and Amit Arora. Dimension reduction in intrusion detection features using discriminative machine learning approach. *International Journal of Computer Science Issues (IJCSI)*, 10(4):324, 2013.

[6] Nathaniel Bastian, David Bierbrauer, Morgan McKenzie, and Emily Nack. ACI IoT Network Traffic Dataset 2023. IEEE Dataport, December 29 2023.

[7] Luca Catarinucci, Danilo de Donno, Luca Mainetti, Luca Palano, Luigi Patrono, Maria Laura Stefanizzi, and Luciano Tarricone. An iot-aware architecture for smart healthcare systems. *IEEE Internet of Things Journal*, 2(6):515–526, 2015.

[8] Robin Chataut, Alex Phoummalayvane, and Robert Akl. 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), 2023.

[9] Patrick J Davis, Sean Coffey, Lubjana Beshaj, and Nathaniel D Bastian. Quantum machine learning for feature selection in internet of things network intrusion detection. In *Quantum Information Science, Sensing, and Computation XVI*, volume 13028, pages 78–92. SPIE, 2024.

[10] Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Samuel Stevens, Boshi Wang, Huan Sun, and Yu Su. Mind2web: Towards a generalist agent for the web, 2023.

[11] Jesús Díaz-Verdejo, Javier Muñoz-Calle, Antonio Estepa Alonso, Rafael Estepa Alonso, and Germán Madinabeitia. On the detection capabilities of signature-based intrusion detection systems in the context of web attacks. *Applied Sciences*, 12(2), 2022.

[12] Qiuhan Gu. Llm-based code generation method for golang compiler testing. In *Proceedings of the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, ESEC/FSE 2023, page 2201–2203, New York, NY, USA, 2023. Association for Computing Machinery.

[13] Azidine Guezzaz, Said Benkirane, Mourade Azrour, and Shahzada Khurram. A reliable network intrusion detection approach using decision tree with enhanced data quality. *Security and Communication Networks*, 2021(1):1230593, 2021.

[14] Yuejun Guo, Constantinos Patsakis, Qiang Hu, Qiang Tang, and Fran Casino. Outside the comfort zone: Analysing llm capabilities in software vulnerability detection. In *European Symposium on Research in Computer Security*, pages 271–289. Springer, 2024.

[15] Izzeddin Gur, Hiroki Furuta, Austin V Huang, Mustafa Safdari, Yutaka Matsuo, Douglas Eck, and Aleksandra Faust. A real-world webagent with planning, long context understanding, and program synthesis. In *The Twelfth International Conference on Learning Representations*, 2024.

[16] Philokypros Ioulianou, Vasileios Vasilakis, Ioannis Moscholios, and Michael Logothetis. A signature-based intrusion detection system for the internet of things. In *Information and Communication Technology Form*, June 2018.

[17] Huaping Jia, Jun Liu, Min Zhang, Xiaohu He, and Weixi Sun. Network intrusion detection based on ie-dbn model. *Computer Communications*, 178:131–140, 2021.

[18] Ansam Khraisat, Iqbal Gondal, and Peter Vamplew. An anomaly intrusion detection system using c5 decision tree classifier. In *Trends and Applications in Knowledge Discovery and Data Mining: PAKDD 2018 Workshops, BDASC, BDM, ML4Cyber, PAISI, DaMEMO, Melbourne, VIC, Australia, June 3, 2018, Revised Selected Papers 22*, pages 149–155. Springer, 2018.

[19] Ansam Khraisat, Iqbal Gondal, Peter Vamplew, and Joarder Kamruzzaman. Survey of intrusion detection systems: techniques, datasets and challenges. *Cybersecurity*, 2(1):1–22, 2019.

[20] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive nlp tasks. In *Advances in Neural Information Processing Systems*, 2020.

[21] Junkai Li, Siyu Wang, Meng Zhang, Weitao Li, Yunghwei Lai, Xinhui Kang, Weizhi Ma, and Yang Liu. Agent hospital: A simulacrum of hospital with evolvable medical agents, 2024.

[22] Wenchao Li, Ping Yi, Yue Wu, Li Pan, and Jianhua Li. A new intrusion detection system based on knn classification algorithm in wireless sensor network. *Journal of Electrical and Computer Engineering*, 2014(1):240217, 2014.

[23] Ziadoon Kamil Maseer, Robiah Yusof, Nazrulazhar Bahaman, Salama A Mostafa, and Cik Feresa Mohd Foozy. Benchmarking of machine learning for anomaly based intrusion detection systems in the cicids2017 dataset. *IEEE access*, 9:22351–22370, 2021.

[24] Brian Matejek, Ashish Gehani, Nathaniel D Bastian, Daniel J Clouse, Bradford J Kline, and Susmit Jha. Safeguarding Network Intrusion Detection Models from Zero-day Attacks and Concept Drift. *AAAI Workshop on Artificial Intelligence for Cyber Security (AICS)*, 2024.

[25] Mokhtar Mohammadi, Tarik A Rashid, Sarkhel H Taher Karim, Adil Hussain Mohammed Aldalwie, Quan Thanh Tho, Moazam Bidaki, Amir Masoud Rahmani, and Mehdi Hosseinzadeh. A comprehensive survey and taxonomy of the svm-based intrusion detection systems. *Journal of Network and Computer Applications*, 178:102983, 2021.

[26] S. Nagaraju, B. Shanmugham, and K. Baskaran. High throughput token driven fsm based regex pattern matching for network intrusion detection system. *Materials Today: Proceedings*, 47:139–143, 2021.

[27] 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. *Sensor*, 2023. (submitted to Journal of Sensors).

[28] Souradip Roy, Juan Li, Bong-Jin Choi, and Yan Bai. A lightweight supervised intrusion detection mechanism for iot networks. *Future Generation Computer Systems*, 127:276–285, 2022.

[29] Wenqi Shi, Ran Xu, Yuchen Zhuang, Yue Yu, Jieyu Zhang, Hang Wu, Yuanda Zhu, Joyce Ho, Carl Yang, and May D. Wang. Ehragent: Code empowers large language models for few-shot complex tabular reasoning on electronic health records, 2024.

[30] Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning, 2023.

[31] Farhan Ullah, Hamad Naeem, Sohail Jabbar, Shehzad Khalid, Muhammad Ahsan Latif, Fadi Al-Turjman, and Leonardo Mostarda. Cyber security threats detection in internet of things using deep learning approach. *IEEE access*, 7:124379–124389, 2019.

[32] Abhishek Verma and Virender Ranga. Machine learning based intrusion detection systems for iot applications. *Wireless Personal Communications*, 111(4):2287–2310, 2020.

[33] Weizhi Wang, Li Dong, Hao Cheng, Xiaodong Liu, Xifeng Yan, Jianfeng Gao, and Furu Wei. Augmenting language models with long-term memory. *Advances in Neural Information Processing Systems*, 36, 2024.

[34] Zhen Xiang, Linzhi Zheng, Yanjie Li, Junyuan Hong, Qinbin Li, Han Xie, Jiawei Zhang, Zidi Xiong, Chulin Xie, Carl Yang, Dawn Song, and Bo Li. Guardagent: Safeguard llm agents by a guard agent via knowledge-enabled reasoning, 2024.

[35] Zhen Yang, Xiaodong Liu, Tong Li, Di Wu, Jinjiang Wang, Yunwei Zhao, and Han Han. A systematic literature review of methods and datasets for anomaly-based network intrusion detection. *Computers Security*, 116:102675, 2022.

[36] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*, 2023.

[37] Yangyang Yu, Haohang Li, Zhi Chen, Yuechen Jiang, Yang Li, Denghui Zhang, Rong Liu, Jordan W. Suchow, and Khaldoun Khashanah. Finmem: A performance-enhanced llm trading agent with layered memory and character design, 2023.

[38] Han Zhang, Akram Bin Sediq, Ali Afana, and Melike Erol-Kantarci. Large language models in wireless application design: In-context learning-enhanced automatic network intrusion detection, 2024.

[39] Qingling Zhao, Mingqiang Chen, Zonghua Gu, Siyu Luan, Haibo Zeng, and Samarjit Chakrabory. Can bus intrusion detection based on auxiliary classifier gan and out-of-distribution detection. *ACM Transactions on Embedded Computing Systems (TECS)*, 21(4):1–30, 2022.

[40] Boyuan Zheng, Boyu Gou, Jihyung Kil, Huan Sun, and Yu Su. Gpt-4v(ision) is a generalist web agent, if grounded. *arXiv preprint arXiv:2401.01614*, 2024.

[41] Wanjun Zhong, Lianghong Guo, Qiqi Gao, He Ye, and Yanlin Wang. Memorybank: Enhancing large language models with long-term memory. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 19724–19731, 2024.

A Appendix A

A.1 General Prompt and Reasoning Trace of `IDS-Agent`

The general prompt of the `IDS-Agent` is illustrated in Figure 3. The process begins with instructing the `IDS-Agent` to load network traffic data and perform feature preprocessing. Afterward, we utilize a range of classifiers to analyze the data. To enhance decision-making, the `IDS-Agent` retrieves prior successful examples from its knowledge base for comparison. In cases where discrepancies arise between the predictions of different models, we prompt the `IDS-Agent` to consult internal or external knowledge bases for additional insights to resolve the conflict. Finally, the `IDS-Agent` consolidates the findings and presents the result in a structured JSON format. Figure 4 provides an example of the reasoning trace produced by the `IDS-Agent` during this process.

A.2 Additional Case Studies

The cases in Figure 5 and Figure 6 highlight the enhanced reasoning ability of `IDS-Agent` with the knowledge retriever. It is shown that `IDS-Agent` not only considers the top-1 predictions but also the second and third predictions and their confidences. Moreover, in these examples, when the models have discrepancies in the predictions, the `IDS-Agent` automatically accesses external databases to extract additional knowledge, aiding in feature analysis and supporting its final decision. By leveraging these external knowledge sources, the `IDS-Agent` gains a deeper understanding of complex attack patterns and anomalies, enhancing both accuracy and decision-making. This dynamic capability allows the `IDS-Agent` to better adapt to new or evolving threats in the IoT environment.

A.3 The Performance of Different ML Classifiers

Table 6 shows the F-score of different ML classifiers on the ACI-IoT'23 dataset as well as our method. Among the six classifiers, MLP achieves the highest F-score of 0.96. The `IDS-Agent` outperforms

General Prompt:

You are a helpful assistant that can implement multi-step tasks, such as intrusion detection. I will give you the traffic features, you are asked to classify it using tools. The final output must be in a JSON format according to the classifier results. You should plan first such as:

1. Load the traffic features from the CSV file. You can use the load_data_line tool to obtain the complete traffic.
2. Preprocessing the feature. This can be done using the data_preprocessing tool. Input the traffic in the original format.
3. load classifiers for classification. This can be done using the classifier tool. You can use multiple classifiers. The tool params include a classifier name, which must be one from {model_names} and the preprocessed features.
4. Retrieve previous successful reasonings to help you predict. This can be done using the memory_retrieve tool with the classifier's names and their classification results as input.
5. When there are discrepancies/disagreements for different models, you can search from vector database/google/wiki to get more information about the difference of attacks to help you make decisions.
6. At the end, you should summarize the results from these classifiers and provide a final result. Summarize the classification with Balance sensitivity, which means balancing the false alarm rate and the missing alarm rate. The predicted label should be the original format of classifier prediction. The final output format **must** be:

Final Answer:
```json
{ 'line_number': \line_number,
'analysis': str, \here is the Analysis,
'predicted_label_top_1': str,
'predicted_label_top_2': str,
'predicted_label_top_3': str,
}
```

User Input:

Now, classify the traffic from file name {file_name} with index {line_number}

Figure 3: General prompt and user input

Thought: I need to follow the plan to classify the traffic features from line number 230. I will start by loading the traffic features from the CSV file.
Action: load_data_line
Action Input: {"line_number": 230}
Observation: flow_duration: 4.198554983, Header_Length: 10051.49, ...
Thought: I have successfully loaded the traffic features from line number 230. Now, I will proceed to preprocess the loaded traffic features to prepare them for classification.
Action: data_preprocessing
Action Input: {"traffic_features": "flow_duration: 4.198554983, Header_Length: 10051.49,..."}
Observation: [-0.355, -0.3088, ...]
Thought: I have successfully preprocessed the traffic features. Now, I will proceed to classify the preprocessed features using multiple classifiers to determine if the traffic record is an attack.
Action: classifier
Action Input: {"modelname": "Random Forest", "traffic_features": [-0.355, -0.3088, ...]}
Observation: Top predictions: DDoS-SlowLoris: 0.9100, DDoS-HTTP_Flood: 0.0600, DoS-HTTP_Flood: 0.0200
Thought: I have obtained the classification results from the Random Forest model. Now, I will classify the same preprocessed features using additional classifiers to gather more predictions.
...

Thought: I have obtained the classification results from all five classifiers. Here are the predictions:
...
Now, I will summarize the results and determine the final predicted labels based on the highest confidence scores from the classifiers.
Summary of Predictions:
- **DDoS-SlowLoris**: Predicted by Random Forest (0.9100), K-Nearest Neighbors (1.0000), and MLP (0.8543).
- **DDoS-HTTP_Flood**: Predicted by Logistic Regression (0.2586), MLP (0.1033), and Support Vector Classifier (0.6331).
- **Recon-PortScan**: Predicted by Logistic Regression (0.1300) and K-Nearest Neighbors (0.0000).
- **BenignTraffic**: Predicted by Support Vector Classifier (0.2320).
Final Decision:
The most consistent prediction across the classifiers is **DDoS-SlowLoris**, which received the highest confidence scores from multiple models.
Now, I will prepare the final output in the required JSON format.

Final Answer:
```json
{
  "line_number": 230,
  "analysis": "The traffic features indicate a high likelihood of a DDoS-SlowLoris attack based on the predictions from multiple classifiers, particularly the Random Forest and K-Nearest Neighbors models.",
  "predicted_label_top_1": "DDoS-SlowLoris",
  "predicted_label_top_2": "DDoS-HTTP_Flood",
  "predicted_label_top_3": "Recon-PortScan"
}
```

Figure 4: An example of the reasoning trace and final answer.

all six classifiers and the majority vote method. Moreover, we achieve a high F-score on the UDP Flood attack, while the majority method only has an F-score of 0.55.

Table 7 shows the F-score of different ML classifiers on the CIC-IoT'23 dataset as well as our method. Among the six classifiers, Random Forest achieves the highest F-score of 0.75. For the `IDS-Agent`, we use the GPT-4o as the core LLM. It is shown that our attack achieved a higher F-score compared with the majority vote classifier. Moreover, we achieved a higher F-score on the benign traffic compared with six classifiers and the majority vote method, which means our method has a lower false alarm rate, which is an important metric for intrusion detection. Figure 7 shows the confusion matrix of the majority voting classifier and `IDS-Agent`.

Table 6: The F-score of different ML classifiers on the ACI-IoT'23 dataset. For the `IDS-Agent`, we use the GPT-4o as the core LLM.

Model	RF	LR	KNN	MLP	DT	SVC	Majority Vote	IDS-Agent
Benign	0.90	0.59	0.91	0.91	0.91	0.80	0.91	0.91
DNS Flood	0.95	0.10	0.80	0.95	0.91	0.91	1.00	0.95
Dictionary Attack	1.00	0.71	0.98	0.95	1.00	0.92	1.00	1.00
ICMP Flood	1.00	0.98	0.98	1.00	0.95	0.98	0.98	0.98
OS Scan	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
Ping Sweep	0.98	0.98	0.97	0.98	0.97	0.98	1.00	1.00
Port Scan	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
SYN Flood	1.00	1.00	1.00	1.00	0.98	1.00	1.00	1.00
Slowloris	1.00	0.43	1.00	1.00	1.00	0.97	1.00	1.00
UDP Flood	0.60	0.00	0.45	0.74	0.50	0.00	0.55	0.80
Vulnerability Scan	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
Macro Avg	**0.95**	**0.71**	**0.92**	**0.96**	**0.93**	**0.87**	**0.96**	**0.97**

Table 7: The F-score of different ML classifiers on the CIC-IoT'23 dataset. For the `IDS-Agent`, we use the GPT-4o as the core LLM.

Model	DT	KNN	LR	MLP	RF	SVC	Majority Vote	IDS-Agent
BenignTraffic	0.79	0.77	0.79	0.75	0.75	0.73	0.74	0.84
DDoS-ACK_Fragmentation	0.98	0.95	0.95	0.93	0.95	0.98	0.95	1.00
DDoS-HTTP_Flood	0.58	0.53	0.24	0.79	0.68	0.38	0.69	0.70
DDoS-ICMP_Flood	0.98	0.95	0.98	0.95	1.00	1.00	1.00	1.00
DDoS-ICMP_Fragmentation	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
DDoS-PSHACK_Flood	1.00	1.00	0.98	1.00	1.00	1.00	1.00	0.95
DDoS-RSTFINFlood	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
DDoS-SYN_Flood	0.72	0.08	0.63	0.73	0.76	0.64	0.75	0.75
DDoS-SlowLoris	0.76	0.74	0.00	0.89	0.79	0.44	0.79	0.82
DDoS-SynonymousIP_Flood	0.70	0.74	0.70	0.72	0.74	0.65	0.74	0.78
DDoS-TCP_Flood	0.69	0.08	0.66	0.67	0.69	0.67	0.67	0.71
DDoS-UDP_Flood	0.71	0.23	0.77	0.75	0.71	0.74	0.71	0.74
DDoS-UDP_Fragmentation	0.97	0.95	0.95	0.97	0.97	0.97	0.97	0.95
DoS-HTTP_Flood	0.65	0.79	0.73	0.85	0.76	0.89	0.83	0.84
DoS-SYN_Flood	0.62	0.57	0.31	0.72	0.76	0.37	0.72	0.80
DoS-TCP_Flood	0.32	0.60	0.00	0.00	0.32	0.00	0.17	0.33
DoS-UDP_Flood	0.53	0.65	0.55	0.52	0.48	0.46	0.48	0.46
MITM-ArpSpoofing	0.54	0.58	0.09	0.60	0.58	0.58	0.62	0.67
Mirai-greeth_flood	0.95	0.98	0.90	0.97	0.97	1.00	0.97	0.95
Mirai-greip_flood	0.98	0.97	0.89	0.98	0.98	1.00	0.98	0.95
Mirai-udpplain	0.95	0.98	1.00	0.98	0.98	1.00	0.98	1.00
Recon-HostDiscovery	0.54	0.45	0.40	0.48	0.55	0.39	0.47	0.53
Recon-OSScan	0.30	0.30	0.08	0.00	0.15	0.10	0.17	0.15
Recon-PortScan	0.41	0.36	0.24	0.31	0.44	0.31	0.37	0.31
Macro Avg	**0.74**	**0.68**	**0.62**	**0.73**	**0.75**	**0.68**	**0.70**	**0.75**

Thought: I have obtained predictions from all classifiers. Now, I will summarize the results to determine the final classification.
The predictions from the classifiers are as follows:

- Random Forest: DDoS-SYN_Flood (0.6458)
- K-Nearest Neighbors: DoS-SYN_Flood (1.0000)
- Logistic Regression: DDoS-SYN_Flood (0.6473)
- MLP: DDoS-SYN_Flood (0.7261)
- Support Vector Classifier: DDoS-SYN_Flood (0.6329)
- Decision Tree: DDoS-SYN_Flood (0.6736)

The majority of classifiers predict "DDoS-SYN_Flood" as the top label. However, K-Nearest Neighbors predicts "DoS-SYN_Flood" with a certainty of 1.0000. To resolve this discrepancy, I will search for more information regarding the features of DDoS and DoS attacks to understand the differences better.
Action: knowledge_retriever
Action Input: {"query": "What are the features of DoS and DDoS attacks and their differences?"
Observation: DoS (Denial-of-Service) and DDoS (Distributed Denial-of-Service) attacks have several features and key differences:
Features of DoS Attacks:...
Features of DDoS Attacks:...
Key Differences:

- **Source of Attack:** DoS attacks come from one source, while DDoS attacks come from multiple sources.
- **Volume of Traffic:** DDoS attacks typically generate a larger volume of traffic, making them more damaging.
- **Detection and Mitigation:** DDoS attacks are harder to detect and eliminate quickly due to their distributed nature, whereas DoS attacks can be easier to manage since they originate from a single source.

Overall, DDoS attacks are generally considered more severe and challenging to defend against compared to DoS attacks.
Final Answer:
{
"line_number": 83,
"analysis": "The majority of classifiers predict 'DDoS-SYN_Flood', indicating a distributed denial-of-service attack. The K-Nearest Neighbors classifier predicts 'DoS-SYN_Flood' with high certainty, but given the nature of the features and the consensus among other classifiers, 'DDoS-SYN_Flood' is deemed the more appropriate classification.",
"predicted_label_top_1": "DDoS-SYN_Flood",
"predicted_label_top_2": "DoS-SYN_Flood",
"predicted_label_top_3": "DDoS-SynonymousIP_Flood"
}

Figure 5: Additional case study on the DDoS attacks of the `IDS-Agent`. From the final analysis, it is shown that the `IDS-Agent` not only considers the predicted labels but also considers the confidence of different classifiers.

A.4 Details for sensitivity Customization

We adjust the detection sensitivity by prompting the core LLM with "Summarize the classification with {sensitivity} sensitivity, {sensitivity details}". Here, 'sensitivity details' will be 'discover the attack as the priority', 'balance the false alarm rate and the missing alarm rate', and 'do not alert unless you are very sure', for 'sensitivity' being 'aggressive', 'balanced' and 'conservative'. The detection performances of `IDS-Agent` for different detection sensitivities are shown in Table 8. It is shown that the 'Aggressive' command achieves a higher recall on the attacks while the 'Conservative' command achieves a higher recall on the benign examples. The classification results, detailed in Table 8 of the appendix, show that the `IDS-Agent` effectively follows these sensitivity instructions without requiring expert intervention or additional tuning.

A.5 The Zero-day Attack Detection Details

We prompt GPT-4o to classify an example as an unknown attack if multiple classifiers output low confidence for their top predictions or if there are conflicting predictions among different classifiers. This is based on our observation that, for unknown attacks, machine learning models typically exhibit relatively low confidence levels, as shown in Figure 8. Specifically, we instruct the LLM to consider an example as a potential unknown attack if more than two models have low confidence (e.g., below a threshold of 0.7). Moreover, if more than two models have low confidence or if different models produce significantly divergent predictions, we direct `IDS-Agent` to search the knowledge base for characteristics of the most probable predicted attacks. If the traffic features do not match these attack characteristics, we confirm the example as an unknown attack and provide this as the final output.

A.6 The Influence of Hyperparameters

To assess the influence of different values of λ_1 and λ_2 in Eq. 1, we conducted experiments by varying these parameters and measuring the impact on retrieval effectiveness and overall classification

Thought: I have obtained the predictions from multiple classifiers. Here are the results:

1. **Random Forest**: - **MITM-ArpSpoofing**: 0.4200 - BenignTraffic: 0.2500 - Recon-PortScan: 0.1300
2. **K-Nearest Neighbors**: - **Recon-PortScan**: 0.4000 - MITM-ArpSpoofing: 0.4000 - BenignTraffic: 0.2000
3. **Logistic Regression**: - **BenignTraffic**: 0.3689 - MITM-ArpSpoofing: 0.3105 - DoS-UDP_Flood: 0.1391
4. **MLP**: - **MITM-ArpSpoofing**: 0.4019 - BenignTraffic: 0.3296 - Recon-HostDiscovery: 0.1095
5. **Support Vector Classifier**: - **BenignTraffic**: 0.6327 - MITM-ArpSpoofing: 0.2328 - Recon-HostDiscovery: 0.0855
6. **Decision Tree**: - **Recon-OSScan**: 1.0000 - Recon-PortScan: 0.0000 - DDoS-UDP_Flood: 0.0000

Now, I will analyze the results to determine the final classification. The predictions show a mix of benign and attack classifications, with some classifiers leaning towards "MITM-ArpSpoofing" and others towards "BenignTraffic." The Decision Tree classifier is particularly confident in classifying it as "Recon-OSScan."To resolve the discrepancies, I will search for more information regarding the features of the attacks and benign traffic to make a more informed decision.

Action: knowledge_retriever

Action Input: {"query": "what is the features of MITM-ArpSpoofing and how to detect it"}

Observation: The features of MITM-ARP-Spoofing include: 1. **Interception of Communication**: The attacker positions themselves between two network devices, allowing them to intercept and potentially modify the data being transmitted.

2. **ARP Cache Poisoning**: The attacker sends falsified ARP messages to both devices, associating their MAC address with the IP address of the other device, thus redirecting traffic through their machine....

Final Answer:

{
"line_number": 83,
"analysis": "The traffic features indicate a potential MITM-ArpSpoofing attack based on the predictions from multiple classifiers, particularly Random Forest and MLP. The Decision Tree classifier also suggests a reconnaissance attack, but the overall consensus leans towards MITM.",
"predicted_label_top_1": "MITM-ArpSpoofing",
"predicted_label_top_2": "BenignTraffic",
"predicted_label_top_3": "Recon-PortScan",
}

Figure 6: Additional case study on the MITM attacks of the `IDS-Agent`. From the final analysis, it is shown that the `IDS-Agent` not only considers the top 1 prediction but also considers the second and third predictions.

Table 8: The classification results of different detection sensitivities.

Sensitivity	Aggressive			Balance			Conservative		
Metrics	Precision	Recall	F1-score	Precision	Recall	F1-score	Precision	Recall	F1-score
Benign	0.96	0.90	0.92	0.87	0.96	0.91	0.60	0.98	0.75
DNS Flood	0.91	1.00	0.95	0.91	1.00	0.95	0.94	0.80	0.86
Dictionary Attack	0.91	1.00	0.95	1.00	1.00	1.00	1.00	0.65	0.79
ICMP Flood	0.95	1.00	0.89	0.95	1.00	0.98	0.95	1.00	0.98
OS Scan	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
Ping Sweep	0.95	1.00	0.98	1.00	1.00	1.00	1.00	1.00	1.00
Port Scan	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
SYN Flood	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
Slowloris	0.95	1.00	0.98	1.00	1.00	1.00	1.00	0.40	0.57
UDP Flood	1.00	0.80	0.89	1.00	0.53	0.69	1.00	0.47	0.64
Vulnerability Scan	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
Macro Avg	**0.97**	**0.97**	**0.97**	**0.98**	**0.95**	**0.96**	**0.95**	**0.85**	**0.87**

performance. Table 9 summarizes the results of our experiments. The experimental results indicate that both recency and content similarity are crucial for effective LTM retrieval. A balanced approach, where λ_1 and λ_2 are equal, provides the best performance, suggesting that the agent benefits from considering both embedding similarity and recency.

Table 9: Performance metrics for different values of λ_1 and λ_2.

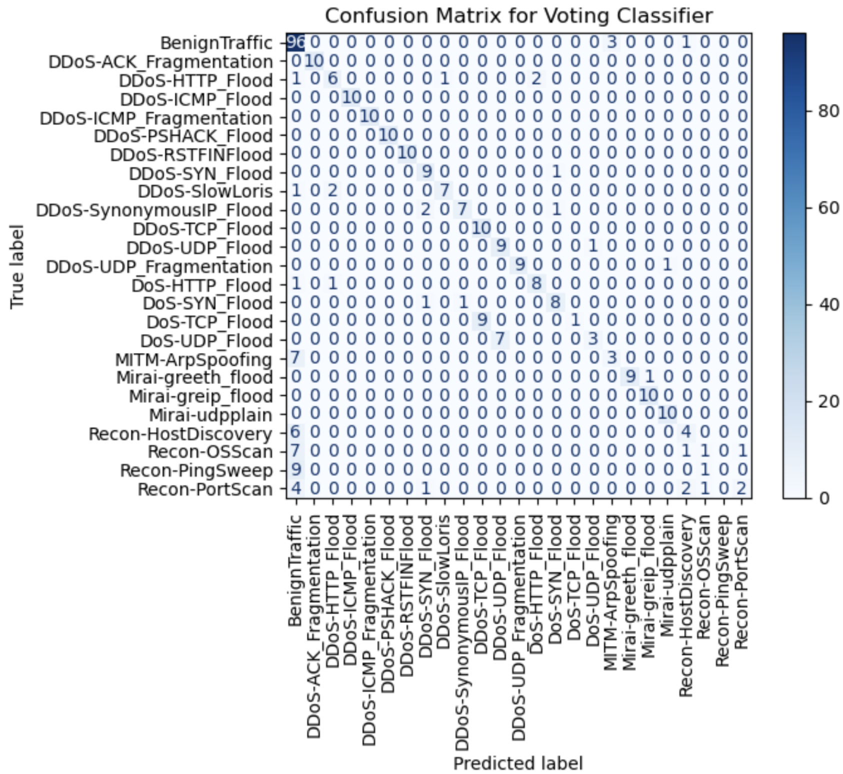
λ_1	λ_2	Accuracy (%)	Precision (%)	Recall (%)
0.1	0.9	97.2	97.2	96.5
0.5	0.5	**98.0**	**98.2**	**97.2**
0.9	0.1	97.3	97.1	96.1

A.7 Excution Time of `IDS-Agent`

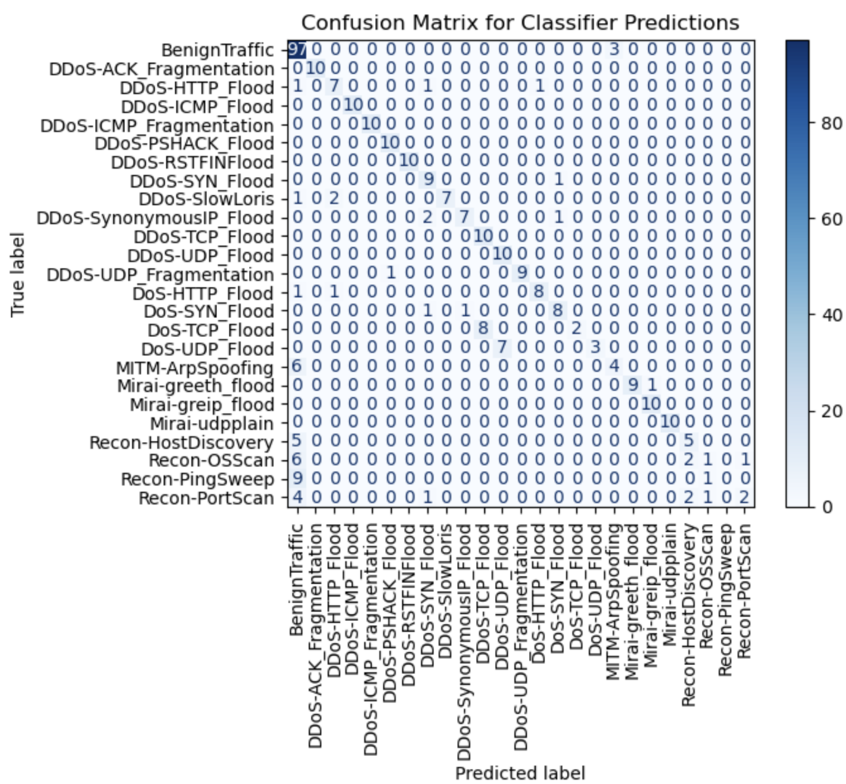
In this section, we evaluate the execution time of the proposed `IDS-Agent` and compare it with the in-context-learning-based GPT-4 approach. We conducted the execution time experiments with the Intel Core i7 CPU of 3.8GHz. The operating system is MacOS 14.6. As shown in Table 10, the `IDS-Agent` balances performance and efficiency, averaging 8.65 seconds per instance. We use GPT-4o API as the core LLM of `IDS-Agent`. The additional time compared to the GPT-4 method is due to the knowledge retrieval and aggregation process, but it remains well within acceptable limits for real-time applications.

Table 10: Execution time comparison between different methods.

Method	GPT-4	`IDS-Agent`
Average Time per Instance (s)	3.36	8.65



(a) Confusion matrix of majority voting classifer



(b) Confusion matrix of IDS-Agent

Figure 7: The confusion matrix of majority voting classifier and IDS-Agent on the CIC-IoT'23 dataset.

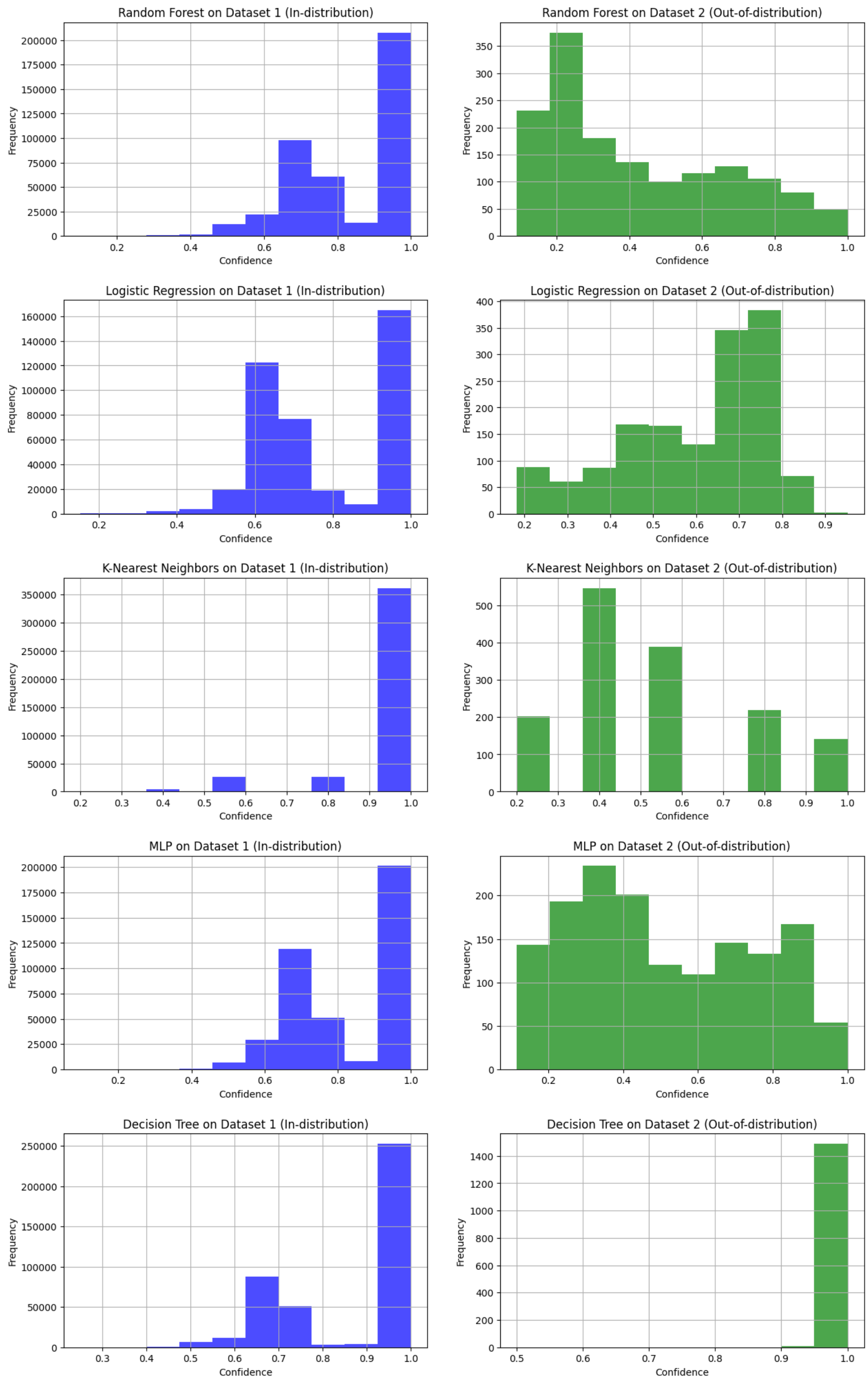


Figure 8: The confidence distributions of difference classifiers on the in-distribution dataset and out-of-distribution dataset.