A Novel Framework for Network Intrusion Detection: The Hierarchical & Source-Aware K1K2NN Algorithm

The Evolving Threat Landscape and the Imperative for Advanced Intrusion Detection

The contemporary digital ecosystem is characterized by an unprecedented level of interconnectedness, driven by the proliferation of Internet of Things (IoT) devices, cloud computing, and distributed networks. While this hyper-connectivity fuels innovation and efficiency, it concurrently expands the attack surface, creating fertile ground for increasingly sophisticated cyber threats. In this environment, traditional Network Intrusion Detection Systems (IDS) are facing a crisis of efficacy. The escalating volume, velocity, and variety of network traffic, coupled with the intricate, multi-stage nature of modern attacks, demand a fundamental rethinking of our approach to network defense. The imperative is clear: a paradigm shift is needed from static, context-agnostic detection methods to dynamic, contextually rich, and structurally intelligent security frameworks.

The Limitations of Conventional Intrusion Detection Systems

For decades, network security has relied on two primary IDS methodologies: signature-based detection and anomaly-based detection. While both have served foundational roles, their inherent limitations are becoming increasingly pronounced in the face of the modern threat landscape.

Signature-Based IDS: This approach operates like an antivirus program, maintaining a vast database of known attack "signatures"—unique patterns or characteristics of malicious traffic.[1] When incoming network traffic matches a predefined signature, an alert is triggered. The principal advantage of this method is its high accuracy and low rate of false positives when detecting known threats. However, its fundamental weakness is its reactive nature. A signature-based IDS is incapable of identifying novel, or "zero-day," attacks for which no signature yet exists.[1] This leaves networks vulnerable during the critical window between the emergence of a new threat and the development and deployment of its corresponding signature.

Anomaly-Based IDS: In contrast, anomaly-based systems attempt to overcome this limitation by first building a model of "normal" network behavior. Any deviation from this established baseline is flagged as a potential intrusion.[4] This approach holds the promise of detecting novel and unforeseen attacks, as it does not rely on prior knowledge of specific threat patterns. However, its practical application is fraught with challenges. The primary difficulty lies in accurately and comprehensively defining "normal" behavior, especially in today's dynamic and heterogeneous networks where traffic patterns can vary widely and change frequently.[3] Consequently, anomaly-based systems are often plagued by a high rate of false positives, where legitimate but unusual network activity is incorrectly flagged as malicious. These false alarms can lead to alert fatigue among security analysts, causing genuine threats to be overlooked.[3] Furthermore, the sheer volume of modern network traffic imposes significant computational burdens, challenging the scalability and real-time performance of these systems.[5]

The Rise of Sophisticated and Multi-Stage Attacks

Compounding the limitations of traditional IDS is the evolution of attack methodologies. Modern cyberattacks are rarely monolithic, single-event intrusions. Instead, they are often meticulously planned, multi-stage campaigns that unfold over time, mirroring a strategic military operation. An attacker may begin with a reconnaissance or Probe phase, using techniques like port scanning to gather information about the target network.[7] This is often followed by an initial compromise, such as a

Remote to Local (R2L) attack, where the attacker gains a foothold on a single machine. From there, they may attempt to escalate privileges through a User to Root (U2R) attack to gain administrative control. The final objective could be data exfiltration, deploying ransomware, or leveraging the compromised network in a large-scale Denial of Service (DoS) or botnet attack.[7]

The proliferation of insecure IoT devices has dramatically amplified this threat. Devices such as smart cameras, thermostats, and industrial sensors often have weak security postures, making them easy targets for compromise. Once infected, these devices can be marshaled into massive botnets, like the infamous Mirai botnet, capable of launching devastating, coordinated attacks.[6] Conventional IDS, which typically analyze individual network packets or flows in isolation, are ill-equipped to handle such complex campaigns. They may detect individual stages of an attack but often fail to connect them, thereby missing the broader strategic intent of the adversary.

The Need for Context: Source-Awareness and Hierarchical Classification

The central thesis of this work is that to overcome these challenges, the next generation of IDS must move beyond simple pattern matching and incorporate a deeper, more contextual understanding of network activity. Effective intrusion detection requires analyzing not just *what* is happening (the traffic characteristics), but also *who* is generating the traffic (the source) and *how* this activity fits into a broader, structured understanding of potential threats. This necessitates the integration of two key concepts: source-awareness and hierarchical classification.

Source-Awareness: The identity and established behavioral patterns of a network device provide invaluable context for interpreting its traffic. For instance, a series of failed login attempts originating from a corporate server is far more alarming than the same activity from a guest Wi-Fi network. Similarly, a smart thermostat that suddenly begins communicating with an unknown external server on an unusual port is a significant anomaly, whereas similar behavior from a developer's workstation might be routine.[9] By creating and maintaining detailed profiles of each device on the network—a practice known as device profiling—an IDS can establish a highly specific baseline of normal behavior for each source. This source-specific context is critical for accurately distinguishing between malicious and benign anomalies, thereby drastically reducing the false positive rate that plagues traditional anomaly detection systems.[12]

Hierarchical Classification: Cyberattacks exhibit a natural taxonomic structure. A specific FTP-Patator brute-force attempt is a type of Brute Force attack, which in turn falls under the broader category of Attack. A flat multi-class classification model, which treats FTP-Patator and SQL Injection as equally distinct classes, fails to capture this inherent relationship. A hierarchical classification approach, however, can be designed to mirror this structure.[13] Such a model would first make a high-level determination (e.g.,

Benign vs. Attack), followed by progressively more granular classifications at lower levels of the hierarchy (e.g., attack family, then specific attack type). This structure offers a more nuanced and accurate diagnostic capability. Critically, it aligns better with the operational realities of network security: misclassifying one type of attack as another (e.g., a DoS as a

Probe) is a far less severe error than misclassifying an attack as Benign traffic.[13] A hierarchical model, by prioritizing the initial

Attack vs. Benign decision at its root, is inherently designed to minimize this most critical type of error.

The fundamental shortcoming of many contemporary machine learning-based IDS is not necessarily the choice of algorithm, but rather the simplistic, "flat" representation of the problem. They address intrusion detection as a standard multi-class classification task, thereby discarding the rich, structured context available in both the network's topology and the adversary's methodology. While standard classifiers can achieve impressive accuracy scores on benchmark datasets [7], the persistence of real-world challenges like high false positive rates and vulnerability to zero-day attacks indicates a disconnect between laboratory performance and operational robustness.[1] The missing element is context. A network flow is not merely an isolated vector of features; it originates from a specific

source with a unique behavioral profile and may be a single step in a larger, temporally correlated *sequence* of actions. The next evolution in intrusion detection, therefore, must transition from this context-free classification to a model of context-rich, structured prediction. This principle directly motivates the development of the Hierarchical and Source-Aware K1K2NN algorithm proposed herein.

Foundational Concepts: From K-Nearest Neighbors to Multi-Stage, Multi-Label Classification

The proposed Hierarchical & Source-Aware K1K2NN (HSA-K1K2NN) algorithm is a synthesis of several powerful concepts from machine learning and data science. To fully appreciate its architecture and novelty, it is essential to first understand its foundational building blocks: the classical K-Nearest Neighbors algorithm, advanced hierarchical search techniques that address its limitations, and the innovative K1K2NN paradigm that introduces a new dimension of relational analysis into the classification process.

The K-Nearest Neighbors (KNN) Algorithm in Network Security

The K-Nearest Neighbors (KNN) algorithm is a cornerstone of non-parametric, instance-based machine learning.[4] Its core principle is intuitive and straightforward: an object is classified based on the class of its closest neighbors in a multi-dimensional feature space.[7] Unlike "eager" learning algorithms that build a generalized model from training data (e.g., a decision tree or a neural network), KNN is a "lazy" learner. It simply stores the entire training dataset in memory. When a new, unclassified data point (a test sample) is introduced, the algorithm computes its distance to every single point in the training set.[17] It then identifies the 'k' nearest training points and assigns the test sample to the class that is most common among these 'k' neighbors, a process known as majority voting.[18]

Mathematical Formulation:
Given a test sample q and a training set P, where each point p ∈ P has a feature vector and a class label, the KNN process is as follows:

1. **Define a Feature Space:** Each network flow is represented as a vector of n numerical features (e.g., duration, packet count, byte count).
2. Choose a Distance Metric: The "closeness" of two points is measured using a distance metric. The most common is the Euclidean distance, defined for two points p and q in an n-dimensional space as:

   d(p,q)=i=1Σn(pi−qi)2

   Other metrics, such as the Manhattan distance, can also be used depending on the nature of the data.18
3. **Find the k Nearest Neighbors:** Calculate d(q, p) for all p ∈ P. Sort these distances and select the k points with the smallest distances.
4. **Assign a Class:** Assign q to the class that appears most frequently among the k selected neighbors.

Strengths and Weaknesses:
The simplicity and interpretability of KNN have made it a popular choice for various classification tasks, including network intrusion detection.7 It is inherently suited for multi-class problems and makes no assumptions about the underlying data distribution.4 However, the naive implementation of KNN suffers from significant drawbacks that limit its practicality in large-scale, real-time security applications.

- **Computational Cost:** The necessity of calculating the distance from the test sample to every training sample results in a computational complexity that is linear with the size of the training set. For datasets with millions of network flows, this process is prohibitively slow.[17]
- **The Curse of Dimensionality:** As the number of features (dimensions) increases, the distance between any two points in the space tends to become more uniform. This makes the concept of "nearest neighbor" less meaningful, as the distance to the nearest neighbor can approach the distance to the farthest neighbor. This phenomenon severely degrades the performance of distance-based algorithms like KNN.[17]
- **Sensitivity to Hyperparameters and Feature Scaling:** The performance of KNN is highly dependent on the choice of k and the distance metric. Furthermore, because it

relies on distance calculations, it is sensitive to the scale of features; features with larger ranges can disproportionately influence the distance metric unless the data is properly normalized.[22]

Optimizing Neighbor Search: Hierarchical Approaches

To overcome the critical scalability limitations of the naive KNN algorithm, researchers have developed more sophisticated data structures and search strategies that avoid an exhaustive search of the entire dataset. These methods, particularly hierarchical ones, dramatically improve search efficiency, making nearest neighbor techniques viable for large-scale applications.

k-Means for k-NN (kMkNN): One effective approach is to preprocess the training data by organizing it into a more structured format. The kMkNN algorithm leverages k-means clustering for this purpose.[23] In a preliminary "buildup" stage, the entire training dataset is partitioned into a predefined number of clusters using the k-means algorithm. For each point, its distance to its assigned cluster center is pre-calculated and stored. During the "searching" stage, when a new query point is introduced, the algorithm first calculates the distances to all cluster centers. It then begins its search in the nearest clusters. Crucially, it employs the triangle inequality to prune the search space. If the distance from the query point to a cluster center, minus the maximum distance of any point within that cluster to the center, is already greater than the distance to the current k-th nearest neighbor found so far, the entire cluster can be safely ignored without performing any individual distance calculations for its members. This technique can lead to a significant reduction in computations, especially for well-structured data.[23]

Hierarchical Navigable Small World (HNSW): A state-of-the-art technique for approximate nearest neighbor (ANN) search is the Hierarchical Navigable Small World (HNSW) algorithm.[24] HNSW constructs a multi-layered graph from the data points. The lowest layer (Layer 0) contains all the data points, with edges connecting points that are close to each other, forming a proximity graph. Each subsequent higher layer is a subset of the layer below it, creating a hierarchical structure. The key innovation is that the connections in the higher layers are typically "long-range," spanning larger distances across the feature space, while connections in the lower layers are "short-range" and more localized.[25]

The search process begins at the top, most sparse layer. The algorithm greedily traverses the graph at this level to find the point closest to the query. This point then serves as the entry point to the next layer down. This process is repeated, progressively refining the search with more granular, short-range connections at each lower level until the bottom layer is reached. This coarse-to-fine search strategy allows the algorithm to quickly navigate to the correct region of the feature space, bypassing vast numbers of irrelevant points. This results in a search complexity that scales logarithmically with the dataset size, making HNSW exceptionally fast and scalable for high-dimensional data.[27]

The K1K2NN Paradigm: A Novel Multi-Label Classification Approach

While hierarchical search methods address the efficiency of KNN, the K1K2NN algorithm, first proposed by Das et al. in the context of computational biology, introduces a novel conceptual enhancement to the classification logic itself.[29] It transforms the classification process from a single-step lookup into a two-stage, multi-label inference task.

The original K1K2NN model was designed to predict the side effects of COVID-19 drugs and was built on two core propositions [29]:

1. Chemically similar drugs are likely to have similar side effects.
2. Side effects that are known to co-occur frequently can be jointly assigned to chemically similar drugs.

This logic is operationalized in a two-stage process:

1. **Step 1 (K1 Neighbors):** For a given test drug, the algorithm first identifies the K1 most similar drugs from the training set based on their chemical properties (the instance-based similarity step). It then performs a majority vote on the known side effects (labels) of these K1 neighbors to generate an initial set of predicted side effects.
2. **Step 2 (K2 Neighbors):** The algorithm then leverages a pre-computed "side effect co-occurrence matrix." This matrix quantifies the similarity between every pair of side effects, typically using a metric like the Jaccard similarity coefficient. For each side effect identified in the K1 step, the algorithm looks up the K2 most similar, or most frequently co-occurring, side effects from this matrix. These additional, highly correlated side effects are then added to the final prediction set for the test drug.

The **Jaccard Similarity Coefficient** is the mathematical tool used to build the co-occurrence matrix. For two sets of labels, A and B, it is defined as the size of their intersection divided by the size of their union [30]:

$J(A,B)=|A \cup B|\,|A \cap B|$

A score of 1 indicates identical sets, while a score of 0 indicates no common elements.31
The innovation of the K1K2NN algorithm lies not just in its two-step procedure, but in its fundamental fusion of two different types of information. The K1 step relies on traditional instance-based similarity, answering the question: "What does this new sample look like?" The

K2 step, however, introduces a relational or semantic model of the *label space itself*, answering the question: "Given what this sample looks like, what else is likely to be true based on known relationships between outcomes?"

A standard KNN classifier's knowledge is limited to the labels directly attached to a sample's neighbors.[16] If a particular attack label is not present in the immediate vicinity of a suspicious network flow, it cannot be predicted. The K1K2NN paradigm overcomes this limitation by incorporating a broader, pre-compiled knowledge base about the relationships between the labels. The K1 step generates an initial, evidence-based hypothesis (e.g., "This network flow's neighbors are primarily associated with

Port Scanning"). The K2 step then enriches this hypothesis with contextual knowledge derived from the co-occurrence matrix (e.g., "Our historical data shows that Port Scanning is very frequently a precursor to Host Discovery and Vulnerability Scanning as part of a larger Reconnaissance campaign"). This transforms the classification from a simple pattern-matching exercise into a more powerful inferential process. It is this conceptual leap—the ability to infer associated labels based on semantic relationships—that makes the K1K2NN paradigm uniquely suited for adaptation to the complex, multi-stage challenges of network intrusion detection.

The Hierarchical & Source-Aware K1K2NN (HSA-K1K2NN) Algorithm: A Novel Framework for Network Security

Building upon the foundational concepts of nearest neighbor classification, hierarchical search, and multi-stage inference, this section formally introduces the Hierarchical & Source-Aware K1K2NN (HSA-K1K2NN) algorithm. This novel framework is architected specifically to address the multifaceted challenges of modern network security by synthesizing these advanced techniques into a single, cohesive system. It represents a departure from traditional, context-agnostic classifiers, proposing a model that is structurally aligned with the nature of cyber threats and contextually informed by the network environment it protects.

Core Principles and Design Philosophy

The design philosophy of HSA-K1K2NN is rooted in the understanding that effective intrusion detection is not a simple classification problem but a complex inferential task that requires a deep, contextual model of the network and its potential adversaries. The algorithm is built upon three integrated pillars, each designed to counteract a specific weakness of conventional IDS:

1. **Hierarchical Classification:** The framework adopts a tree-based classification structure to mirror the natural taxonomy of network attacks. This allows for a coarse-to-fine diagnostic process that is both computationally efficient and logically aligned with the security analyst's workflow. It prioritizes the most critical classification decision—distinguishing malicious from benign traffic—before proceeding to more granular analyses.
2. **Source-Awareness:** The algorithm moves beyond the analysis of traffic in isolation by incorporating rich contextual information about the traffic's origin. By creating and leveraging detailed profiles of both trusted network devices and known malicious actors, HSA-K1K2NN can make more accurate and nuanced judgments, significantly reducing the ambiguity that leads to high false positive rates in traditional anomaly detection systems.
3. **K1K2NN Multi-Stage Logic:** At its core, the algorithm employs the two-stage K1K2NN inference mechanism. This enables it to not only identify the most probable attack type based on direct similarity (the K1 step) but also to predict other likely, co-occurring attack behaviors based on a learned model of attacker tactics (the K2 step). This feature is specifically designed to detect and contextualize the components of sophisticated, multi-stage attack campaigns.

The Hierarchical Structure

The HSA-K1K2NN algorithm organizes the classification task into a multi-level decision tree, a structure that has been shown to be highly effective in the cybersecurity domain.[13] This hierarchical design offers several advantages over a "flat" multi-class model, including improved efficiency by breaking a complex problem into a series of simpler ones, and enhanced interpretability of the results. The proposed hierarchy consists of three levels:

- **Level 1 (Root): Binary Classification (Benign vs. Attack):** The entry point for any new network flow is the root classifier. Its sole purpose is to make the most critical distinction: is this traffic legitimate or is it part of an attack? This binary classification acts as a high-performance filter. As research has shown, minimizing the misclassification of an attack as benign traffic is the single most important goal of an IDS.[13] By dedicating the first stage exclusively to this task, the model can be optimized for high recall on the Attack class, ensuring that suspicious activities are not prematurely dismissed.
- **Level 2: Broad Attack Categorization:** If a flow is classified as Attack at Level 1, it is

passed down to the second level of the hierarchy. At this stage, a set of classifiers, one for each broad attack family, determines the general nature of the threat. The categories at this level correspond to well-established attack types, such as Denial of Service (DoS), Probe (reconnaissance), Remote to Local (R2L), User to Root (U2R), and Botnet activity.[7] This classification provides an immediate, high-level situational awareness to security analysts.
- **Level 3: Granular Attack Identification:** Once an attack family is identified at Level 2, the flow is routed to a specialized classifier at Level 3. This final level is responsible for identifying the specific, granular attack vector being used. For example, within the Probe family, this classifier would distinguish between a Port Scan, an OS Scan, or a Vulnerability Scan. Within the DoS family, it might identify a TCP Flood, a UDP Flood, or a Slowloris attack. This level of detail is crucial for forensic analysis and for deploying targeted countermeasures.

Each node within this hierarchy is not a generic classifier but a specialized HSA-K1K2NN instance, trained exclusively on the subset of data relevant to its specific decision. To ensure scalability and real-time performance, the nearest neighbor search at each node is accelerated using a highly efficient graph-based index, such as that provided by the Hierarchical Navigable Small World (HNSW) algorithm.[24]

The Source-Awareness Module

A cornerstone of the HSA-K1K2NN framework is its Source-Awareness Module. This component is responsible for creating and maintaining a dynamic database of **Source Profiles**, which provide essential context for the classification engine. This module transforms the classification problem from one that only considers traffic features to one that considers both the traffic and its originator's typical behavior.

- **Device Profiling:** For all devices within the monitored network, particularly in heterogeneous IoT environments, the module establishes a baseline of normal behavior. During a learning phase, it passively monitors traffic from each device (identified by its IP or MAC address) and builds a statistical profile. This profile can include features such as the set of protocols and ports it typically uses, the average and variance of its packet sizes and flow durations, the entropy of destination IPs it communicates with, and the periodicity of its connections.[9] This detailed, per-device baseline is the key to accurately identifying anomalous behavior.
- **Attacker Profiling:** The module also creates profiles for known malicious sources. Using historical data from past attacks (e.g., from benchmark datasets or internal incident logs), it characterizes the Tactics, Techniques, and Procedures (TTPs) associated with different attackers or botnets. This could include preferred scanning methods, common exploit

payloads, or typical command-and-control (C2) communication patterns.[34]

- **The Profile Vector:** Each source on the network is ultimately represented by a numerical **profile vector**. This vector, which summarizes the source's identity and behavioral tendencies, is then concatenated with the feature vector of each network flow it generates. This creates an augmented input vector that feeds into the HSA-K1K2NN classifiers, ensuring that every classification decision is made with full awareness of the source's context.

Integrating K1K2NN Logic into the Hierarchy

The core inferential power of the HSA-K1K2NN algorithm comes from the adaptation and integration of the K1K2NN multi-stage classification logic at each node of the hierarchical framework. This requires translating the original propositions from the biomedical domain to the language of network security:

- **Adapted Proposition 1:** *Network flows that originate from similar sources (as defined by their profiles) and exhibit similar traffic characteristics are highly likely to belong to the same attack class.* This forms the basis of the K1 step. It asserts that similarity should be measured not just in terms of traffic patterns but also in terms of behavioral context.
- **Adapted Proposition 2:** *Attack techniques that are known to frequently co-occur as part of a recognized multi-stage attack campaign can be probabilistically assigned to a suspicious network flow, even if only one of the techniques is directly observed.* This forms the basis of the K2 step. It introduces a model of attacker methodology into the classification process, allowing the system to infer intent and predict the next stages of an attack.

This adaptation transforms the K1K2NN paradigm from a model for predicting drug side effects into a sophisticated framework for identifying and contextualizing complex cyber threats.

The integration of the source-awareness module fundamentally redefines the concept of "similarity" within the nearest neighbor framework. In a standard KNN implementation, the distance between two network flows is calculated based on a vector of traffic features like packet counts, duration, and protocol flags.[7] Two flows are considered "close" if these numerical values are similar. This approach, however, is context-blind. For example, a small burst of UDP traffic from an IoT security camera to an unknown external IP address might have a similar feature vector to a DNS query from a corporate server. A standard KNN might view them as neighbors, but from a security perspective, their implications are vastly different.

By concatenating the source profile vector with the flow feature vector, the HSA-K1K2NN algorithm injects this critical context directly into the feature space. The distance calculation now implicitly includes a "source similarity" component. As a result, the anomalous flow from the IoT camera will be considered mathematically "closer" to historical examples of malicious traffic from other compromised IoT devices, even if its raw traffic features differ from, for example, a classic DoS attack originating from a dedicated server. This creates a more meaningful, "contextual distance" metric, where the algorithm learns that *who* is sending the traffic is just as important as *what* traffic is being sent. This directly addresses the high false positive rate of traditional anomaly detection systems by providing a much stronger, source-specific basis for what constitutes "normal" versus "anomalous" behavior.[3]

To better position the proposed algorithm within the landscape of existing techniques, the following table provides a comparative analysis.

Algorithm	Classification Structure	Search Complexity	Source Context-Awareness	Handles Multi-Stage Attacks	Primary Limitation
Standard KNN	Flat	O(n)	No	No	Poor scalability with large datasets; Curse of dimensionality.[17]
Weighted KNN	Flat	O(n)	No	No	Same scalability issues as Standard KNN; requires feature weighting.
kMkNN	Clustered (Pre-processing)	Sub-linear (Average Case)	No	No	Performance degrades on uniformly distributed data; overhead of clustering.[23]

HNSW (as search)	Hierarchical Graph	$O(\log n)$ (Approximate)	No	No	It is a search method, not a full classifier; can have approximation errors.[27]
HSA-K1K2NN	**Hierarchical Tree**	**$O(\log n)$ (Approximate)**	**Yes**	**Explicitly (via K2 step)**	High data dependency for profiles and co-occurrence matrix; increased model complexity.

This comparison highlights the unique synthesis of features within the HSA-K1K2NN framework. It leverages the logarithmic search complexity of modern ANN methods like HNSW, but embeds this within a formal classification hierarchy. Most importantly, it introduces two novel dimensions not present in its predecessors: explicit source-awareness to provide context, and the K2 inference step to explicitly model and detect the relationships between stages of an attack.

Architectural Deep Dive: Mathematical Formulation and Operational Steps of HSA-K1K2NN

This section provides a formal, technical specification of the HSA-K1K2NN algorithm. It details the data representation, the construction of the core knowledge models, and the step-by-step operational flow of the classification process. This formulation is intended to provide a clear and unambiguous blueprint for researchers and practitioners seeking to understand or implement the framework.

Data Representation and Preprocessing

The effectiveness of any machine learning model is contingent upon the quality and representation of its input data. The HSA-K1K2NN algorithm utilizes an augmented feature vector that combines real-time traffic characteristics with historical source context.

- **Network Flow Vector (Vflow):** Each unidirectional or bidirectional network flow is first converted into a fixed-length numerical vector, Vflow. This vector is generated using a flow analysis tool such as CICFlowMeter, which extracts a comprehensive set of features designed to capture the statistical and temporal properties of the communication.[37] A representative set of features includes, but is not limited to:
  - **Basic Features:** Source/Destination IP and Port, Protocol.
  - **Flow Timing Features:** Flow Duration, Forward/Backward Inter-Arrival Time (Mean, Std, Max, Min).
  - **Packet and Byte Count Features:** Total Forward/Backward Packets, Total Forward/Backward Bytes.
  - **Packet Size Features:** Forward/Backward Packet Length (Mean, Std, Max, Min).
  - **TCP Flag Features:** Counts of FIN, SYN, RST, PSH, ACK, URG flags.
  - Sub-flow Features: Packets per second, Bytes per second.
    This results in a high-dimensional vector: $Vflow\in Rdflow$, where dflow is the number of traffic features (typically > 80).36
- **Source Profile Vector (Vprofile):** For each unique source IP address in the network, a corresponding profile vector, Vprofile, is generated. This vector is a statistical summary of the source's historical behavior, learned from the training data. It can be constructed as a concatenation of features such as:
  - A histogram of destination ports contacted.
  - The Shannon entropy of destination IP addresses contacted (to measure communication randomness).
  - The ratio of outgoing to incoming traffic.
  - Average flow duration and packet count for the source.
    This results in a vector: $Vprofile\in Rdprofile$, where dprofile is the number of profiling features.9
- Augmented Input Vector (X): The final input to the classifier is the augmented vector X, created by concatenating the flow and profile vectors:

  $X=Vflow\oplus Vprofile$

  where $\oplus$ denotes the concatenation operation. The resulting vector $X\in Rdflow+dprofile$ encapsulates both the instantaneous state of the traffic and the historical context of its originator.
- **Feature Scaling:** Given that the KNN algorithm and its distance calculations are highly sensitive to the range of feature values, all components of the augmented vector X must be normalized. A standard scaling technique, such as Z-score normalization (scaling to

zero mean and unit variance) or Min-Max scaling (scaling to a range), is applied to the entire training dataset and subsequently used to transform test samples.[20]

Building the Attack Co-occurrence Matrix (Mco)

The Attack Co-occurrence Matrix, Mco, is a critical knowledge base that fuels the K2 inference step of the algorithm. It represents the learned relationships between different types of granular attacks, quantifying which attack techniques tend to appear together within the same broader campaign. This matrix is pre-computed from the labeled training dataset.

- **Methodology:** The construction process involves analyzing the training data to identify sessions or campaigns that contain multiple distinct attack labels. A "session" can be defined as all traffic from a single source IP address within a specific time window (e.g., 5 minutes).
    1. **Session Grouping:** Group all labeled attack flows in the training data by their source IP and a discretized time window.
    2. **Label Set Creation:** For each session, create a set of the unique attack labels present within that session.
    3. Co-occurrence Calculation: Let $L=\{l1,l2,\ldots,lN\}$ be the set of all unique granular attack labels in the dataset. Let $S(li)$ be the set of all sessions in which attack label li appears. The co-occurrence matrix Mco is an $N\times N$ matrix where the entry $Mco[i,j]$ is the Jaccard similarity between the sets of sessions containing attacks li and lj:

       $Mco[i,j]=J(S(li),S(lj))=|S(li)\cup S(lj)|\,|S(li)\cap S(lj)|$

       This calculation provides a normalized score from 0 to 1, where a higher value indicates that attacks li and lj are more frequently observed together in the same attack campaign.29 This matrix effectively serves as a learned graph of attack relationships.

The HSA-K1K2NN Algorithm: Step-by-Step Operational Flow

The following steps detail the classification process for a new, incoming network flow, represented by its augmented vector Xtest. The process proceeds top-down through the pre-trained hierarchical model.

Input: A new augmented vector Xtest, integer hyperparameters K1 and K2, the trained hierarchical model, and the pre-computed co-occurrence matrix Mco.

Step 1: Hierarchical Traversal (Root Node - Level 1)

1. The vector Xtest is submitted to the Level 1 classifier, which is trained to distinguish between Benign and Attack traffic.
2. K1-NN Classification:
   a. Using an HNSW-accelerated search, find the K1 nearest neighbors of Xtest within the Level 1 training dataset. Let this set of neighbors be denoted as NK1(1).
   b. Perform a majority vote on the class labels of the neighbors in NK1(1). The resulting prediction is Lpred(1).
3. Decision:
   a. If Lpred(1) is Benign, the process terminates. The flow is classified as normal, and no further action is taken.
   b. If Lpred(1) is Attack, the flow is passed down to the appropriate child node(s) in Level 2 for further analysis.

Step 2: Broad Attack Classification (Level 2)

1. The vector Xtest is now processed by the Level 2 classifier, which is trained on attack traffic only and categorized into broad families (e.g., DoS, Probe).
2. K1-NN Classification:
   a. Find the K1 nearest neighbors of Xtest within the Level 2 training dataset, resulting in the set NK1(2).
   b. Collect the set of unique labels from these neighbors, denoted as LK1. The most frequent label in this set is the initial predicted attack family, Lfamily.
3. K2-NN Label Enrichment:
   a. For each unique label l∈LK1, consult the co-occurrence matrix Mco. Identify the K2 labels that have the highest Jaccard similarity to l.
   b. Aggregate all these retrieved labels into a new set, LK2.
   c. The final set of predicted labels for this level is the union of the K1 and K2 sets: Lfinal(2)=LK1∪LK2.
4. **Decision:** The primary classification for this level is the attack family that has the highest frequency count within the enriched set Lfinal(2). The flow is then passed to the corresponding specialized classifier in Level 3.

Step 3: Granular Attack Identification (Level 3)

1. The process is repeated at the appropriate Level 3 node, which is trained only on data from a specific attack family (e.g., the Probe classifier).
2. The K1-K2 classification and enrichment steps are performed as in Level 2, but using the more specialized Level 3 training data.
3. **Output:** The final output of the algorithm is a structured "threat package" containing:
   - **Primary Classification:** The single most likely granular attack type (e.g., Port Scan).
   - **Associated Labels:** The full enriched set of likely co-occurring attacks (e.g., {Host

Discovery, Vulnerability Scan}). This provides crucial context about the potential scope and intent of the attack.

Mathematical Formulation Summary

The core mathematical operations of the HSA-K1K2NN algorithm can be summarized as follows:

- Weighted Distance Metric: To allow for differential importance between traffic features and source profile features, a weighted Euclidean distance can be employed:

  $$Dist(X_a,X_b)=w_{flow}^2\sum_{i=1}^{d_{flow}}(a_{flow,i}-b_{flow,i})^2+w_{profile}^2\sum_{j=1}^{d_{profile}}(a_{profile,j}-b_{profile,j})^2$$

  where w_{flow} and $w_{profile}$ are hyperparameters that can be tuned to balance the influence of the two vector components.
- K1 Prediction (at any level l):

  $$L_{K1}=\text{mode}(\{\text{label}(n)\mid n\in N_{K1}^{(l)}(X_{test})\})$$
- K2 Enrichment:
  $$L_{K2} = \bigcup_{l \in L_{K1}} \{\text{TopK2\_Similar}(l, M_{co})\}$$

  where $\text{TopK2\_Similar}(l, M_{co})$ returns the labels of the K2 columns in row l of M_{co} with the highest values.
- Final Prediction (at level l):

  $$\text{PrimaryLabel}(l)=\text{mode}(L_{K1}\cup L_{K2})$$

This structured, multi-stage process provides a mechanism for robust inference that is resilient to minor adversarial evasion tactics. An attacker might employ obfuscation techniques to slightly alter the traffic characteristics of a specific attack, such as an SQL Injection, potentially moving its V_{flow} vector just far enough in the feature space to avoid having other SQL Injection flows as its immediate nearest neighbors in the K1 step.[1] However, it is significantly more difficult for an attacker to hide the precursor activities that are part of their standard TTPs, such as initial

Port Scanning and Vulnerability Scanning. In such a scenario, the K1 step might classify the neighbors of the obfuscated flow as Vulnerability Scanning. The pre-computed M_{co} matrix, built from thousands of historical attacks, will encode the strong, frequently observed correlation between Vulnerability Scanning and SQL Injection.[31] The K2 step would then retrieve

SQL Injection as a highly co-occurring label and add it to the final prediction set. Consequently, even if the direct, primary evidence for the attack is weakened by evasion, the strong, secondary relational evidence allows the system to correctly infer the likely true nature of the threat, making the overall framework more robust.

Strategic Applications: Deploying HSA-K1K2NN for Enhanced IoT Network Security

The theoretical power of the HSA-K1K2NN algorithm finds its most compelling application in the domain of Internet of Things (IoT) security. The unique characteristics of IoT environments—massive scale, device heterogeneity, constrained resources, and a vast attack surface—present a perfect storm of challenges for conventional security systems. The specific design features of HSA-K1K2NN, namely its source-awareness and its ability to contextualize multi-stage attacks, are directly tailored to address these very challenges.

Use Case: Securing a Heterogeneous IoT Network

Consider a typical modern IoT deployment, such as a smart building or an industrial control system (ICS) network. Such an environment contains a diverse ecosystem of connected devices: HVAC sensors, security cameras, smart lighting systems, programmable logic controllers (PLCs), and employee workstations, all communicating on the same network.[6] Each of these device types has a distinct and predictable "normal" behavior. A security camera should be sending a steady stream of video data to a specific server, an HVAC sensor should be sending small, periodic temperature readings, and a PLC should be communicating with specific industrial equipment using specialized protocols.

The security objective in this environment is to detect a wide range of threats, from an individually compromised device that begins behaving erratically to a large-scale, coordinated botnet attack that leverages hundreds of devices to launch a DDoS attack or exfiltrate sensitive data.[34] This is precisely the scenario where HSA-K1K2NN is designed to excel.

Attacker and Device Profiling in Practice

The efficacy of the HSA-K1K2NN algorithm begins with the robust implementation of its Source-Awareness Module. This involves a practical, data-driven process of building detailed profiles for every entity on the network.

Building Device Profiles (Source-Awareness): The process of creating the profile vector (Vprofile) for each legitimate IoT device is critical. It begins with an initial baseline period where the IDS operates in a passive monitoring mode, capturing and analyzing all traffic originating from each new device to learn its normal operational patterns.[11] This learned baseline is then encapsulated in the device's profile vector. To make this process concrete and actionable, a specific set of features must be engineered from the raw network traffic. Drawing from extensive research on IoT traffic analysis, a powerful feature set can be constructed to build these profiles, as exemplified by applying it to a dataset like Bot-IoT.

Feature Name	Description	Feature Type	Relevance to Profiling	Relevant Snippets
Avg_Pkt_Size_Sent	Average size of packets sent by the source over a sliding time window.	Continuous	Differentiates devices sending small sensor readings from those streaming large video files.	40
Dest_IP_Entropy	Shannon entropy of unique destination IP addresses contacted by the source.	Statistical	A low entropy indicates communication with a few fixed servers (normal for many IoT devices), while high entropy suggests scanning behavior.	9

Port_Usage_Histogram	A vector representing the frequency distribution of destination ports used by the source.	Categorical	Fingerprints the specific applications or services a device is designed to use (e.g., port 80 for web, port 53 for DNS).	20
Protocol_Ratio_TCP/UDP	The ratio of TCP flows to UDP flows initiated by the source.	Continuous	Distinguishes between devices that require reliable, connection-oriented communication (TCP) and those that use lightweight, connectionless protocols (UDP).	10
Flow_Periodicity	The statistical regularity of communication intervals (e.g., mean and standard deviation of inter-flow times).	Statistical	Identifies devices with highly periodic "heartbeat" or reporting functions, common in sensor networks. Deviations can indicate anomalous activity.	41

Attacker Profiling: In parallel, profiles are created for known malicious actors. When an attack is identified in the training data (e.g., a labeled attack from the Bot-IoT or CIC-IDS2017 datasets), the traffic characteristics and behavioral patterns of the attacking source(s) are aggregated into an attacker profile. This allows the system to recognize not just specific attacks, but also the digital fingerprints of the actors or botnets that perpetrate them.

Detecting Multi-Stage Attacks from Compromised Devices

The true strategic advantage of HSA-K1K2NN becomes apparent when analyzing a realistic, multi-stage attack scenario within an IoT network. Consider the following chain of events:

1. **Initial Compromise:** An attacker successfully compromises a device with a weak security posture, such as a smart lightbulb with default credentials.
2. **Internal Reconnaissance:** The compromised lightbulb, now part of a botnet, begins to scan the internal network to discover other devices and potential vulnerabilities. This constitutes a Probe attack.
3. **Lateral Movement:** The attacker uses information from the scan to exploit a vulnerability on a more critical asset, such as a file server on the same network, gaining unauthorized access. This is an R2L attack.
4. **Coordinated Attack:** Finally, the attacker commands the lightbulb, along with other compromised devices in the botnet, to flood an external target with traffic, participating in a DDoS attack.

An HSA-K1K2NN-based IDS would analyze this sequence as follows:

- When the lightbulb begins its internal scan, this traffic is a stark deviation from its established profile (Vprofile), which would indicate only periodic communication with a specific control server. This deviation makes the augmented input vector X for these flows highly anomalous.
- At Level 2 of the hierarchy, the K1 step would likely classify this traffic as a Probe attack, based on its similarity to other network scanning traffic in the training data.
- Crucially, the K2 step would then activate. By consulting the attack co-occurrence matrix (Mco), the algorithm would find that Probe attacks have a high statistical correlation with subsequent R2L and DDoS attacks.
- The system would therefore generate an enriched alert, flagging the traffic as a Probe attack but also warning the security analyst that there is a high probability of subsequent lateral movement and participation in a DDoS attack. This provides not just detection, but predictive intelligence about the attacker's likely intent and next steps.

Addressing the "Cold-Start" Problem for New Devices

A significant challenge for any IDS that relies on behavioral profiling is the "cold-start" problem: how to handle a new device that has just been added to the network, for which no historical data and thus no behavioral profile exists.[42] Pure anomaly detection systems are particularly vulnerable here, as any traffic from the new device is, by definition, a deviation from the known baseline.

HSA-K1K2NN offers a more graceful solution to this problem. When a new, unprofiled device connects:

1. **Initial Profiling:** If possible, an initial, generic Vprofile can be assigned based on metadata, such as the device's manufacturer (derived from its MAC address OUI) or its declared device type (e.g., "IP Camera").
2. **Hybrid Inference:** In the absence of a robust profile, the algorithm can be configured to temporarily down-weight the profile component of the distance metric and rely more heavily on the traffic features (Vflow).
3. **Leveraging Global Knowledge:** Even without a specific source profile, the K2 step remains fully functional. The system can still use its global knowledge of attack co-occurrence patterns to identify suspicious sequences of behavior.
4. **Incremental Learning:** As the new device operates, the Source-Awareness Module collects its traffic data and incrementally builds its unique behavioral profile. After a sufficient learning period, the profile becomes robust, and the algorithm transitions to its fully source-aware operational mode.

This hybrid approach, which combines instance-based learning with a pre-existing knowledge model (Mco), allows the system to provide meaningful protection even for new devices, mitigating the critical vulnerability window inherent in the cold-start problem.

The output of the HSA-K1K2NN framework transcends that of a traditional classifier. Instead of generating a stream of discrete, uncorrelated alerts (e.g., "Alert: Port Scan from 192.168.1.101," followed minutes later by "Alert: Exploit Detected from 192.168.1.101"), it produces a holistic "threat package." Upon detecting the initial port scan, the system might generate a single, enriched alert: Primary Classification: Probe (Port Scan), Associated High-Probability Threats: {R2L, DoS}, Source Context: IoT Camera (Anomalous Activity - High Deviation from Profile). This single output provides a complete narrative for the security operations center (SOC). It communicates not only what has already happened, but what is statistically likely to happen next, and it identifies the compromised asset. This level of decision intelligence enables a shift from reactive to proactive security. It allows automated security orchestration, automation, and response (SOAR) platforms to implement more intelligent, risk-based responses. For example, a simple Probe alert with no strong K2 associations might trigger heightened monitoring, whereas an alert with a chain of high-severity associated labels like R2L and Data Exfiltration could trigger an immediate, automated network quarantine of the offending device, effectively neutralizing the threat before it can escalate.

Empirical Validation: A Proposed Framework for Performance Evaluation

The conceptual and architectural soundness of the HSA-K1K2NN algorithm must be substantiated through rigorous empirical evaluation. A robust experimental framework is required to validate its performance, quantify the benefits of its novel components, and benchmark it against the current state of the art in intrusion detection. This section outlines such a framework, detailing the choice of datasets, experimental setup, performance metrics, and baseline models necessary for a comprehensive and credible assessment.

Benchmark Datasets

The selection of appropriate datasets is paramount for a meaningful evaluation. The chosen datasets must be modern, large-scale, and contain the specific types of traffic and attack scenarios that HSA-K1K2NN is designed to address. To this end, a dual-dataset strategy is proposed, with each dataset chosen to specifically test the core innovations of the algorithm.

- **CIC-IDS2017:** This dataset, developed by the Canadian Institute for Cybersecurity, is an ideal choice for evaluating the hierarchical classification and multi-stage attack detection capabilities of the algorithm.[37] Its key strengths include:
    - **Attack Diversity:** It contains a wide variety of modern, realistic attacks, including Brute Force, DoS, DDoS, Web Attacks, Infiltration, and Botnets, providing a comprehensive testbed for the multi-level classification hierarchy.[37]
    - **Source Identifiers:** The dataset includes labeled source and destination IP addresses, which are essential for training and testing the source-awareness module.[37]
    - **Multi-Stage Scenarios:** The dataset explicitly includes documented multi-stage attacks, such as an Infiltration attack that involves an initial exploit followed by a port scan, providing the ground truth needed to validate the effectiveness of the K2 label enrichment step.[37]
    - **Realistic Traffic:** It features realistic background traffic generated by profiling the behavior of human users, ensuring that the models are tested in a non-sterile environment.[37]
- **Bot-IoT:** This dataset is specifically designed to reflect the security challenges of IoT

environments, making it the perfect choice for validating the device profiling and source-awareness components of HSA-K1K2NN.[41] Its key attributes are:

- **IoT-Centric Attacks:** It includes attacks commonly perpetrated by and against IoT devices, such as various forms of DoS and DDoS, reconnaissance scans, and data exfiltration.[10]
- **Realistic Network Topology:** The data was generated in a lab environment that realistically simulates an IoT network, incorporating a mix of normal and botnet traffic from various IoT devices.[47] This provides the heterogeneous device environment required to properly build and test the device profiling module.
- **Large Scale:** With over 72 million records in its full version, the dataset is well-suited for testing the scalability and performance of the HNSW-accelerated search mechanism.[41]

This dual-dataset strategy is not redundant; it is a deliberate design choice to ensure that each novel architectural contribution of HSA-K1K2NN—its hierarchical structure, its multi-stage attack detection, and its source-awareness—is rigorously and independently validated against a benchmark that is optimally suited for that specific task.

Experimental Setup and Preprocessing

The experimental design will follow best practices in machine learning research to ensure reproducibility and fairness of comparison.

1. **Data Splitting:** The datasets will be split into training, validation, and testing sets. To simulate a real-world deployment scenario, a temporal split will be used, where the model is trained on data from earlier days and tested on data from later days. This prevents data leakage and ensures the model is evaluated on its ability to generalize to unseen future traffic.
2. **Feature Engineering:** The preprocessing pipeline will involve two main stages. First, a tool like CICFlowMeter will be used to extract the network flow vectors (Vflow) from the raw PCAP files. Second, the training data will be used to construct the source profile vectors (Vprofile) and the attack co-occurrence matrix (Mco), as described in Section 4. These learned models will then be applied to the test set.
3. **Implementation:** The algorithm will be implemented in a standard data science environment, such as Python, utilizing libraries like scikit-learn for the core classification logic and specialized libraries like FAISS or hnswlib for the high-performance HNSW indexing. Hyperparameter tuning (for K1, K2, and HNSW parameters) will be performed using the validation set.

Performance Metrics

Evaluating a hierarchical, multi-label classifier requires a more sophisticated set of metrics than a standard binary or multi-class problem.

- **Level 1 (Binary Classification):** For the root node (Benign vs. Attack), standard performance metrics will be used:
  - **Accuracy:** Overall correctness of predictions.
  - **Precision:** The proportion of predicted attacks that were actual attacks.
  - **Recall (Detection Rate):** The proportion of actual attacks that were correctly identified.
  - **F1-Score:** The harmonic mean of Precision and Recall.
  - **False Positive Rate (FPR):** The proportion of benign traffic incorrectly classified as an attack. This is a critical metric, as a low FPR is essential for operational viability.[49]
  - **ROC-AUC:** The area under the Receiver Operating Characteristic curve, which measures the model's ability to discriminate between the two classes across all possible thresholds.[52]
- **Levels 2 & 3 (Multi-Class/Multi-Label Classification):**
  - **Macro/Micro/Weighted Averages:** For Precision, Recall, and F1-score, averaged versions will be used to provide a comprehensive view of performance across all classes, especially in the presence of class imbalance (where some attack types are much rarer than others).[32]
  - **Jaccard Similarity Score:** This metric is essential for evaluating the multi-label output of the K1-K2 process. It will be calculated on a per-session basis, comparing the set of predicted labels (from Lfinal) with the set of ground truth labels for that session. This directly measures the algorithm's ability to correctly identify both the primary attack and its co-occurring companions.[30]

Baseline Models for Comparison

To rigorously establish the value of HSA-K1K2NN, its performance will be compared against a suite of strong baseline models, including both established machine learning algorithms and ablated versions of the proposed algorithm itself.

- **State-of-the-Art Classifiers:**
  - **Random Forest (RF) and XGBoost:** These ensemble tree-based models are consistently reported in the literature as top-performing classifiers on both the

CIC-IDS2017 and Bot-IoT datasets, making them the primary benchmarks to outperform.[38]

  - **Deep Learning Models:** A representative deep learning model, such as a Multi-Layer Perceptron (MLP) or a hybrid CNN-LSTM architecture, will be included, as these have also demonstrated state-of-the-art results.[40]
- **Ablation Study:** To isolate and quantify the contribution of each novel component of the HSA-K1K2NN architecture, it will be compared against several simplified versions:
  1. **Standard KNN:** A baseline flat KNN classifier to establish the performance of the core algorithm without any enhancements.
  2. **Hierarchical KNN (H-KNN):** A version with the hierarchical structure but using a standard KNN at each node (no source-awareness or K2 step). This will measure the benefit of the hierarchical decomposition alone.
  3. **Source-Aware KNN (SA-KNN):** A flat KNN classifier that uses the augmented source-aware input vector (X). This will measure the benefit of the source-awareness module in isolation.
  4. **Hierarchical K1K2NN (H-K1K2NN):** The full algorithm but without the source-awareness module (using only Vflow as input). This will measure the combined benefit of the hierarchy and the K2 step.

This comprehensive evaluation framework, particularly the ablation study, will provide clear, quantitative evidence of the value added by each architectural innovation in the HSA-K1K2NN design.

To provide a clear performance target, the following table summarizes the reported state-of-the-art results from various studies using the CIC-IDS2017 dataset. The goal of the proposed HSA-K1K2NN algorithm will be to exceed these benchmarks, particularly in terms of F1-Score and False Positive Rate.

Algorithm	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)	Source Citation
XGBoost	99.11	-	-	-	[56]
Random Forest	99.00	-	-	-	[38]
K-Nearest Neighbors	99.00	-	-	-	[38]
XGBM	99.86	-	-	-	[15]

(with feature selection)					
CZOLSTM (Deep Learning)	99.83	99.83	99.82	99.82	15
Hybrid CNN-LSTM	99.84	-	-	-	54

This table establishes the gold standard of performance on this widely used benchmark. Any novel algorithm must demonstrate a competitive, if not superior, performance against these figures to be considered a meaningful advancement in the field. While achieving high accuracy is important, the primary hypothesis is that HSA-K1K2NN will offer a superior balance of high detection rates (Recall) and low false alarms (low FPR), coupled with the richer, more contextually valuable output provided by the K2 inference step.

Analysis and Future Directions: Strengths, Limitations, and the Path Forward for HSA-K1K2NN

The development of the Hierarchical & Source-Aware K1K2NN (HSA-K1K2NN) algorithm represents a principled effort to engineer an intrusion detection system that is structurally and contextually aligned with the complexities of the modern threat landscape. While the proposed framework holds significant promise, a critical and balanced analysis is necessary to understand its anticipated strengths, acknowledge its potential limitations, and chart a course for future research and development.

Anticipated Strengths

The unique synthesis of hierarchical classification, source-awareness, and multi-stage inference is expected to yield several key advantages over existing IDS models.

- **Enhanced Accuracy for Multi-Stage Attacks:** The K2 label enrichment mechanism is the algorithm's most significant innovation for threat intelligence. By leveraging a

pre-computed model of attack co-occurrence, the system can infer the presence of related attack techniques, even if they are not directly observed in a flow's immediate neighborhood. This should lead to a marked improvement in the detection rate of coordinated, multi-stage campaigns, which are often missed by classifiers that treat each flow as an independent event.
- **Reduced False Positives:** The source-awareness module provides a powerful tool for disambiguation. By establishing a detailed behavioral baseline for each device on the network, the algorithm can more accurately distinguish between activity that is truly anomalous (e.g., a printer attempting to SSH into a server) and activity that is merely unusual but legitimate (e.g., a system administrator running a network scan). This source-specific context is hypothesized to significantly lower the false positive rate, a critical factor for the operational usability of any IDS.
- **Improved Interpretability and Decision Support:** The output of the HSA-K1K2NN algorithm is inherently more interpretable than that of a flat classifier. Instead of a single, context-free label, it provides a hierarchical diagnosis (e.g., Attack -> Probe -> Port Scan) and a set of associated, likely future threats. This structured "threat package" provides a richer narrative to security analysts, enabling them to understand not just what happened, but also its potential implications, thereby facilitating faster and more effective incident response.
- **Scalability for Large-Scale Networks:** By building the nearest neighbor search at each node of the hierarchy on a high-performance approximate nearest neighbor (ANN) index like HNSW, the algorithm is designed for scalability. The logarithmic search complexity ensures that the system can handle the massive volumes of traffic characteristic of modern enterprise and IoT networks without a prohibitive degradation in performance.[24]

Potential Limitations and Challenges

Despite its promising architecture, the HSA-K1K2NN framework is not without its challenges and potential limitations that must be acknowledged and addressed in future work.

- **Computational and Architectural Complexity:** The algorithm is, by design, more complex than a standard classifier. The overhead of the hierarchical structure, combined with the need to build and maintain a database of source profiles and a large attack co-occurrence matrix, introduces significant computational and data management complexity. The training phase, in particular, would be more resource-intensive than that of a single, monolithic model.
- **Dependency on Training Data Quality:** The performance of the algorithm is heavily contingent on the quality, volume, and diversity of the training data. The source-awareness module requires sufficient data from each device to build a reliable profile. Similarly, the attack co-occurrence matrix (Mco) is only as good as the examples

of multi-stage attacks present in the training set. A sparse or biased training dataset could lead to incomplete profiles and a weak co-occurrence model, thereby degrading the performance of the K2 inference step.

- **Hyperparameter Tuning:** The framework introduces several new and sensitive hyperparameters that require careful tuning, including K1 and K2 at each level of the hierarchy, as well as the construction and search parameters for the underlying HNSW graphs. Finding the optimal configuration for these parameters will likely require extensive experimentation and may vary significantly between different network environments and datasets.
- **The "Frozen Start" Problem:** While the algorithm includes a strategy to mitigate the "cold-start" problem for new devices of a known type, it may still struggle with the "frozen start" problem.[44] This occurs when a completely novel type of device, with a behavioral pattern unlike anything seen in the training data, is introduced to the network. In such cases, building an initial profile is difficult, and the system may be temporarily vulnerable until enough data can be collected to characterize the new device's normal behavior.

Future Research Directions

The HSA-K1K2NN framework should be viewed not as a final solution, but as a foundational architecture upon which further research can build. Several promising avenues exist for extending and enhancing its capabilities.

- **Online and Adaptive Learning:** The current proposal describes a model trained in an offline batch mode. A critical next step is to develop mechanisms for online learning, allowing the system to update its source profiles and the attack co-occurrence matrix in real-time as new traffic is observed and new threats are identified. This would enable the IDS to adapt dynamically to changes in the network environment and the evolving tactics of adversaries.
- **Automated Hierarchy Generation:** The proposed three-level hierarchy is based on a standard, predefined attack taxonomy. Future research could explore the use of unsupervised learning or clustering techniques to automatically discover the optimal hierarchical structure directly from the data itself. This could reveal non-obvious relationships between attack types and lead to a more data-driven and potentially more effective classification structure.
- **Hardware Acceleration:** To meet the stringent latency requirements of real-time intrusion detection in high-throughput networks (e.g., 100 Gbps and beyond), research into hardware acceleration is essential. The core computational bottlenecks of the algorithm—the high-dimensional distance calculations and graph traversal—are well-suited for parallelization on specialized hardware such as Graphics Processing Units

(GPUs) or Field-Programmable Gate Arrays (FPGAs).[1]

- **Integration with Federated Learning:** A major challenge in building source profiles, especially in multi-tenant or privacy-sensitive environments, is the need to centralize traffic data. Federated Learning (FL) offers a powerful solution to this problem.[57] A future version of the HSA-K1K2NN system could be designed where lightweight profiling models are trained locally on edge devices or gateways, without the raw traffic ever leaving its local domain. Only the model updates (the learned profiles) would be sent to a central server for aggregation. This would enable the creation of robust, source-aware models in a distributed, privacy-preserving manner, making the system far more applicable to real-world IoT and enterprise deployments.

Ultimately, the most significant long-term challenge for the operational deployment of a system like HSA-K1K2NN lies not in the refinement of the classification algorithm itself, but in solving the "data logistics" problem. The power of the algorithm is derived from two large, dynamic knowledge bases: the comprehensive set of source profiles and the rich attack co-occurrence matrix. In any real network, devices are constantly being added, removed, or updated, and attacker TTPs are in a perpetual state of evolution. This means that both knowledge bases are subject to concept drift and will become stale over time. A static, one-time training process is therefore insufficient for long-term viability. The system requires a robust, automated MLOps (Machine Learning Operations) pipeline capable of continuous data ingestion, real-time profile updates, and periodic, intelligent retraining of the relational attack model. This shifts the long-term research focus from simply improving classification accuracy by a few percentage points to architecting a sustainable, adaptive, and continuously learning security ecosystem. The integration of techniques like federated learning is a direct response to this challenge, representing a path toward a system that can evolve alongside the threats it is designed to defeat.

Works cited

1. Network Intrusion Detection Using a Hardware-Based Restricted Coulomb Energy Algorithm on a Cognitive Processor - AAAI, accessed August 17, 2025, https://cdn.aaai.org/ocs/10355/10355-46094-1-PB.pdf
2. An Enhanced Intrusion Detection Model Based on Improved kNN in WSNs - MDPI, accessed August 17, 2025, https://www.mdpi.com/1424-8220/22/4/1407
3. Use of K-Nearest Neighbor classifier for intrusion detection, accessed August 17, 2025, https://www.eecis.udel.edu/~cavazos/cisc850-spring2017/papers/KNN.pdf
4. Improvement of K-nearest Neighbors (KNN) Algorithm for Network Intrusion Detection Using Shannon-Entropy - Journal of Communications, accessed August 17, 2025, https://www.jocm.us/uploadfile/2021/0720/20210720030454657.pdf
5. A Hybrid Framework for Intrusion Detection in Healthcare Systems Using Deep Learning, accessed August 17, 2025, https://pmc.ncbi.nlm.nih.gov/articles/PMC8790147/
6. Machine Learning-Based Intrusion Detection Methods in IoT Systems: A

Comprehensive Review - MDPI, accessed August 17, 2025, https://www.mdpi.com/2079-9292/13/18/3601
7. K Nearest Neighbor Based Model for Intrusion Detection System - ResearchGate, accessed August 17, 2025, https://www.researchgate.net/publication/364119665_K_Nearest_Neighbor_Based_Model_for_Intrusion_Detection_System
8. Ensemble Neural Network and K-NN Classifiers for Intrusion Detection - INTERNATIONAL JOURNAL OF COMPUTER SCIENCE AND INFORMATION TECHNOLOGIES, accessed August 17, 2025, https://www.ijcsit.com/docs/Volume%205/vol5issue02/ijcsit20140502343.pdf
9. (PDF) A Survey on IoT Profiling, Fingerprinting, and Identification - ResearchGate, accessed August 17, 2025, https://www.researchgate.net/publication/360976914_A_Survey_on_IoT_Profiling_Fingerprinting_and_Identification
10. Description of the Bot-IoT dataset. | Download Scientific Diagram - ResearchGate, accessed August 17, 2025, https://www.researchgate.net/figure/Description-of-the-Bot-IoT-dataset_tbl1_373206288
11. Mastering Device Profiling in IoT - Number Analytics, accessed August 17, 2025, https://www.numberanalytics.com/blog/mastering-device-profiling-in-iot
12. Mitigating Adversarial Attacks against IoT Profiling - MDPI, accessed August 17, 2025, https://www.mdpi.com/2079-9292/13/13/2646
13. arxiv.org, accessed August 17, 2025, https://arxiv.org/html/2403.13013v1
14. [2403.13013] Hierarchical Classification for Intrusion Detection System: Effective Design and Empirical Analysis - arXiv, accessed August 17, 2025, https://arxiv.org/abs/2403.13013
15. Evaluation performances on CIC-IDS2017 dataset (a) accuracy, precision - ResearchGate, accessed August 17, 2025, https://www.researchgate.net/figure/Evaluation-performances-on-CIC-IDS2017-dataset-a-accuracy-precision-recall-and_fig3_370496293
16. k-nearest neighbors algorithm - Wikipedia, accessed August 17, 2025, https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm
17. What is the k-nearest neighbors algorithm? - IBM, accessed August 17, 2025, https://www.ibm.com/think/topics/knn
18. Mathematical explanation of K-Nearest Neighbour - GeeksforGeeks, accessed August 17, 2025, https://www.geeksforgeeks.org/machine-learning/mathematical-explanation-of-k-nearest-neighbour/
19. K-NN Algorithm and it's Mathematical Implementation | by Priyanka Parashar - Medium, accessed August 17, 2025, https://medium.com/@priyankaparashar54/k-nn-and-its-mathematical-implementation-67129fdc0480
20. ML-4: K-Nearest Neighbors Algorithm in Cybersecurity: Detecting Malware - Medium, accessed August 17, 2025, https://medium.com/@amuldhungel01/ml-4-k-nearest-neighbors-algorithm-in-c

ybersecurity-detecting-malware-d559a7ed3822
21. Intrusion Detection Systems, Issues, Challenges, and Needs - Atlantis Press, accessed August 17, 2025, https://www.atlantis-press.com/journals/ijcis/125951139/view
22. Deep Dive on KNN: Understanding and Implementing the K-Nearest Neighbors Algorithm, accessed August 17, 2025, https://arize.com/blog-course/knn-algorithm-k-nearest-neighbor/
23. A Fast Exact k-Nearest Neighbors Algorithm for High Dimensional ..., accessed August 17, 2025, https://pmc.ncbi.nlm.nih.gov/articles/PMC3255306/
24. Efficient and robust approximate nearest neighbor search using Hierarchical Navigable Small World graphs - arXiv, accessed August 17, 2025, https://arxiv.org/abs/1603.09320
25. What is a Hierarchical Navigable Small World (HNSW) graph index, and how does it organize vectors to enable efficient approximate nearest neighbor search? - Milvus, accessed August 17, 2025, https://milvus.io/ai-quick-reference/what-is-a-hierarchical-navigable-small-world-hnsw-graph-index-and-how-does-it-organize-vectors-to-enable-efficient-approximate-nearest-neighbor-search
26. Hierarchical Navigable Small Worlds (HNSW) | Pinecone, accessed August 17, 2025, https://www.pinecone.io/learn/series/faiss/hnsw/
27. Hierarchical navigable small world - Wikipedia, accessed August 17, 2025, https://en.wikipedia.org/wiki/Hierarchical_navigable_small_world
28. What is a Hierarchical Navigable Small World - MongoDB, accessed August 17, 2025, https://www.mongodb.com/resources/basics/hierarchical-navigable-small-world
29. K1K2NN: A novel multi-label classification approach based on neighbors for predicting COVID-19 drug side effects - PubMed, accessed August 17, 2025, https://pubmed.ncbi.nlm.nih.gov/38579549/
30. Multi-label classification performance metrics - Mastering Machine Learning with scikit-learn - Second Edition [Book] - O'Reilly Media, accessed August 17, 2025, https://www.oreilly.com/library/view/mastering-machine-learning/9781788299879/87b63eb8-f52c-496a-b73b-42f8aef549fb.xhtml
31. Jaccard Index Formula – Measuring Set Similarity in Classification and Clustering | Uplatz Blog, accessed August 17, 2025, https://uplatz.com/blog/jaccard-index-formula-measuring-set-similarity-in-classification-and-clustering/
32. jaccard_score — scikit-learn 1.7.1 documentation, accessed August 17, 2025, https://scikit-learn.org/stable/modules/generated/sklearn.metrics.jaccard_score.html
33. Hierarchical Intrusion Detection Using Machine Learning and Knowledge Model - MDPI, accessed August 17, 2025, https://www.mdpi.com/2073-8994/12/2/203
34. Anatomy of attacks on IoT systems: review of attacks, impacts and ..., accessed August 17, 2025, https://www.oaepublish.com/articles/jsss.2022.07
35. Threat modelling in Internet of Things (IoT) environments using dynamic attack graphs, accessed August 17, 2025,

https://www.frontiersin.org/journals/the-internet-of-things/articles/10.3389/friot.2024.1306465/full
36. Intrusion Detection System Using Machine Learning Algorithms - GeeksforGeeks, accessed August 17, 2025, https://www.geeksforgeeks.org/machine-learning/intrusion-detection-system-using-machine-learning-algorithms/
37. Intrusion detection evaluation dataset (CIC-IDS2017) - University of New Brunswick, accessed August 17, 2025, https://www.unb.ca/cic/datasets/ids-2017.html
38. Features in cic ids 2017 dataset | Download Scientific Diagram - ResearchGate, accessed August 17, 2025, https://www.researchgate.net/figure/Features-in-cic-ids-2017-dataset_tbl1_343850781
39. IoT, Anomaly Detection, Machine Learning, K-Nearest Neighbors, Random Forest, Real-Time Detection - ARTEII, accessed August 17, 2025, https://international.arteii.or.id/index.php/IJIES/article/download/50/55/230
40. CIC-IDS-2017-MLPs-v.1.0 - Kaggle, accessed August 17, 2025, https://www.kaggle.com/code/nolovelost/cic-ids-2017-mlps-v-1-0
41. The Bot-IoT Dataset | UNSW Research, accessed August 17, 2025, https://research.unsw.edu.au/projects/bot-iot-dataset
42. The Cold-Start Problem In ML Explained & 6 Mitigating Strategies, accessed August 17, 2025, https://spotintelligence.com/2024/02/08/cold-start-problem-machine-learning/
43. Recommendation Systems • Cold Start - aman.ai, accessed August 17, 2025, https://aman.ai/recsys/cold-start/
44. A Machine Learning Approach for Solving the Frozen User Cold-Start Problem in Personalized Mobile Advertising Systems - MDPI, accessed August 17, 2025, https://www.mdpi.com/1999-4893/15/3/72
45. Network Intrusion dataset(CIC-IDS- 2017) - Kaggle, accessed August 17, 2025, https://www.kaggle.com/datasets/chethuhn/network-intrusion-dataset
46. An Intrusion Detection System Using BoT-IoT - MDPI, accessed August 17, 2025, https://www.mdpi.com/2076-3417/13/9/5427
47. The BoT-IoT Dataset - openargus, accessed August 17, 2025, https://openargus.org/argus-ml?view=article&id=35:the-bot-iot-dataset&catid=2
48. The BoT-IoT Dataset - Impact Cyber Trust, accessed August 17, 2025, https://www.impactcybertrust.org/dataset_view?idDataset=1296
49. Intrusion Detection Systems: A State-of-the-Art Taxonomy and Survey - ResearchGate, accessed August 17, 2025, https://www.researchgate.net/publication/365479082_Intrusion_Detection_Systems_A_State-of-the-Art_Taxonomy_and_Survey
50. Evaluation Metrics for Intrusion Detection Systems - A Study-2014 | PDF - Scribd, accessed August 17, 2025, https://www.scribd.com/document/835010476/Evaluation-Metrics-for-Intrusion-Detection-Systems-A-Study-2014
51. Network Intrusion Detection Systems Performance Metrics - ResearchGate,

accessed August 17, 2025, https://www.researchgate.net/figure/Network-Intrusion-Detection-Systems-Performance-Metrics_tbl1_340933690
52. Evaluation of Intrusion Detection Systems - PMC, accessed August 17, 2025, https://pmc.ncbi.nlm.nih.gov/articles/PMC4844520/
53. Classifier Chain Ensemble: Multilabel Classification Techniques - LabEx, accessed August 17, 2025, https://labex.io/tutorials/ml-classifier-chain-ensemble-49079
54. Performance comparison with state of the art using the CICIDS2017 dataset., accessed August 17, 2025, https://www.researchgate.net/figure/Performance-comparison-with-state-of-the-art-using-the-CICIDS2017-dataset_tbl8_374931842
55. Hybrid Machine Learning Models for Intrusion Detection in IoT: Leveraging a Real-World IoT Dataset - arXiv, accessed August 17, 2025, https://arxiv.org/html/2502.12382v1
56. Evaluation of DoS/DDoS Attack Detection with ML Techniques on CIC-IDS2017 Dataset - SciTePress, accessed August 17, 2025, https://www.scitepress.org/Papers/2023/116057/116057.pdf
57. Comparative Review of the Intrusion Detection Systems Based on Federated Learning: Advantages and Open Challenges - MDPI, accessed August 17, 2025, https://www.mdpi.com/1999-4893/15/7/247