Artifact Evaluated
NDSS
Available
Functional
Reproduced

PANDORA: Lightweight Adversarial Defense For Edge IoT Using Uncertainty-Aware Metric Learning

Avinash Awasthi*, Pritam Vediya*, Hemant Miranka†, Ramesh Babu Battula*, Manoj Singh Gaur‡
*Department of Computer Science and Engineering, Malaviya National Institute of Technology, Jaipur 302017, India.
†The LNM Institute of Information Technology, Jaipur 302031, India.
‡Indian Institute of Technology Jammu, Jammu and Kashmir 181221, India.

***Abstract*—The rapid augmentation of Internet of Things (IoT) devices that are resource-constrained in nature has significantly expanded the attack surface, exposed critical vulnerabilities in the network. As a result, traditional Intrusion Detection Systems (IDS), which rely on static, signature-based approaches, have become increasingly obsolete. Modern adversaries now employ sophisticated, automated, and often novel (zero-day) attacks that can easily bypass such conventional defenses. Moreover, the existing IDS models with machine learning often fail in real-world scenarios to handle challenges like concept drift and an inability to generalize to unseen threats. To address these gaps, we introduce PANDORA (Probabilistic Adversarial Network Defense Over Resource-constrained Architectures), a novel, end-to-end framework for detecting zero-day attacks on edge devices. PANDORA makes three key contributions: 1) It learns uncertainty-aware probabilistic embeddings to create robust representations of network traffic; 2) It introduces a novel Probabilistic Manifold Structuring and Distance (PMSD) Loss function that enables effective zero-shot generalization; and 3) It utilizes an efficient Mamba-Mixture of Experts (MoE) architecture for on-device deployment. To validate our approach, we also introduce the TTDFIoTIDS2025 dataset, a new, high-fidelity benchmark featuring complex, programmatically generated attacks. Our extensive evaluations demonstrate that PANDORA significantly outperforms state-of-the-art models, achieving an F1-score of 0.971 with just 10-shot adaptation on CICIDS2017. Critically, it achieves up to 99% accuracy in zero-shot detection under domain shift and, when deployed on a Raspberry Pi, maintains a low memory footprint of 24 MB and a throughput of up to 4.26 flows/sec, proving its practical viability for real-time edge security.**

I. Introduction

The continual drive for hyper connectivity, data, and automation used in billions of resource-constrained Internet of Things (IoT) devices has fundamentally reshaped the security landscape, powering everything from smart cities [1] and industrial automation (Industry 4.O, and Industry 5.O) [2] to connected healthcare [3]. This hyper-connectivity has dissolved traditional network perimeters, giving rise to a vast and heterogeneous attack surface composed of devices that are difficult to secure [4]–[7]. Compounding the challenge, these devices often rely on lightweight protocols like Message Queuing Telemetry Transport (MQTT) and Constrained Application Protocol (CoAP), rendering conventional endpoint protection ineffective [8]. Consequently, attackers now exploit this surface with sophisticated, automated attacks designed to evade static defenses, often employing novel or subtle variations of known threats such as Distributed Denial of Services (DDoS), amplification attacks and etc [9], [10].

To counter such threats, Network Intrusion Detection Systems (NIDS) have served as a foundational line of defense. This approach is based on AI-driven anomaly detection [11], [12]. By learning a model of normal behavior, anomaly-based systems can identify previously unseen threats as deviations from an established baseline. *Therefore, an effective, modern NIDS must combine the precision of signature-based classification for known threats with the adaptive intelligence required to detect entirely unknown attacks.*

Recent modern NIDS methodologies have increasingly adopted Machine Learning (ML) and Deep Learning (DL) algorithms and architectures [13]–[18]. Although existing modern NIDS models demonstrate high efficacy on static datasets, their performance often degrades significantly in dynamic, real-world network environments - a phenomenon known as *model decay*. A primary driver of this model decay is the inherent heterogeneity within attack categories, a challenge that can be formalized as the problem of *intra-class fragmentation*. A high-level threat category, C_k, is not always a monolithic entity with a single modal data distribution. Instead, its probability distribution, $P(x|y = C_k)$, is more accurately described as a mixture of N distinct sub-class distributions:

$$P(x|y = C_k) = \sum_{i=1}^{N} w_i \cdot P_i(x|y = C_{k,i}) \quad (1)$$

Where $C_{k,i}$ represents the i-th sub-class of C_k, and w_i are the mixture weights. Each sub-class $C_{k,i}$ possesses distinct statistical characteristics, such as a distinct mean $\mu_{k,i}$ and covariance $\Sigma_{k,i}$, dominant in separate feature dimensions. Consequently, a model trained primarily on samples from one mode of the distribution ($C_{k,i}$) will fail to generalize when

Network and Distributed System Security (NDSS) Symposium 2026
23-27 February 2026, San Diego, CA, USA
ISBN 979-8-9919276-8-0
https://dx.doi.org/10.14722/ndss.2026.240713
www.ndss-symposium.org

faced with a novel variant from a different mode ($C_{k,j}$ where $j \neq i$) that manipulates an alternative set of features.

Beyond the challenge of this internal diversity faced by an NIDS model, additional challenges are posed by external temporal and environmental pressures. The temporal pressure of concept drift describes the gradual evolution of benign and malicious traffic, slowly degrading a model's accuracy as feature relevance shifts [19]–[21]. A more severe temporal challenge is concept shift, an abrupt change defined by the emergence of a new class of threat. It very definition of a zero-day attack is a statically trained model that has no mechanism to recognize in the present NIDS scenarios [22]. Finally, the threat model environmental pressure of *domain shift* describes a model's inability to generalize when deployed in a new NIDS-related environment, such as a transition from *an enterprise testbed* to *a live IoT network* with different protocols and baseline feature distributions [23], [24]. Consequently, an effective IDS cannot be a static artifact, but it must be an adaptive system capable of handling the future multifaceted forms of *model decay*.

The most acute of these challenges is the concept shift accompanying the abrupt emergence of a zero-day attack. Conventional supervised models are not better for this task because, by design, they learn to map inputs to a fixed, predefined set of output classes when presented with a zero-day attack; such a model lacks the category unknown. It is forced to misclassify the new threat as one of the known classes it was trained on, leading to a critical security failure. To overcome this, the supervised learning approach for NIDS moves towards Zero-Shot Learning (ZSL) and its complete framework, a paradigm designed to classify and generalize for unseen classes. Instead of learning rigid decision boundaries, a ZSL system learns a rich embedding space where similarity corresponds to proximity, enabling it to generalize to unseen classes of threats [25]–[28].

We contend that the Manifold Hypothesis governs the success of any such ZSL system for NIDS; namely, that an effective model must learn to project different classes of network traffic into distinct, separable clusters, or manifolds, within this embedding space [29] [30]. However, current ZSL approaches for NIDS are fundamentally limited in their ability to structure this manifold effectively. They typically rely on two simplifying assumptions that create a brittle embedding space first they use deterministic embeddings, which represent a complex and variable network flow as a single point, failing to capture its inherent uncertainty; and second they employ fixed distance metrics like Euclidean distance, cosine similarity etc, which are sensitive to volumetric changes but often fail to distinguish the subtle, pattern-based signatures of stealthy attacks. This creates a clear research gap for a more robust learning framework to structure an uncertainty-aware, probabilistic manifold.

To address this research gap, we present *PANDORA* (Probabilistic Adversarial Network Defense Over Resource-constrained Architectures). This novel, end-to-end framework achieves proactive anomaly detection through three key architectural innovations: 1) uncertainty-aware probabilistic embeddings that model the variance of each network flow; 2) a new *Probabilistic Manifold Structuring and Distance (PMSD) Loss*; and 3) an efficient *Mamba-Mixture of Experts (MoE)* architecture that makes the fine-grained analysis needed to distinguish between subtle and complex attack variations computationally feasible on resource-constrained devices.

A persistent practicality gap challenges NIDS research. Models are often evaluated on outdated and homogeneous datasets where threat classes are artificially separable. This allows a model to learn a simple, biased decision boundary that performs well in testing but fails to generalize to real-world complexity. Consequently, high accuracy scores are meaningless if the model is also too computationally expensive for the resource-constrained hardware where it is needed most [31]. To bridge this gap, we developed a modern dataset designed to exhibit complex intra-class fragmentation across heterogeneous IoT scenarios. We leverage this dataset within our physical IoT testbed to validate the practical, on-device efficacy of NIDS architectures.

- *Adversarial IoT Dataset:* We release a high-fidelity IoT intrusion dataset with subclass-level attacks and heterogeneous traffic.
- *PANDORA Framework:* We propose an edge-efficient architecture integrating uncertainty-aware embeddings, a new PMSD loss for zero-shot generalization, and a lightweight Mamba-MoE backbone.
- *Adaptation and Intelligence:* We enable zero-shot detection and few-shot prototype adaptation, offering contextual threat insights beyond binary alerts.
- *Real-World Validation:* We demonstrate end-to-end deployment on a resource-constrained IoT testbed, showing field-ready performance.
- Code and Dataset Availability: Code is available at *https://doi.org/10.5281/zenodo.17881774*.

The remainder of the paper is organized as follows: Section II introduces background concepts, Section III reviews related work, Section IV details the PANDORA architecture, Section V describes the TTDFIOTIDS2025 dataset, Section VI presents the evaluation, Section VII provides discussion, and Section VIII concludes the paper.

II. Background

Our methodology is built upon synthesizing three advanced concepts representing data not as points but as distributions, learning a generalizable similarity metric, and using highly efficient neural architectures for feature encoding.

A. Probabilistic Embeddings and Manifolds

Traditional neural network encoders and layers map high-dimensional inputs like network flows and attack threat vectors to a single, deterministic vector in a latent space for classification, which inherently assumes that each input can be represented with perfect certainty [32]–[34]. This assumption is a poor fit for network traffic because flows belonging to the same class can exhibit significant statistical differences due to

variations in protocol implementation, attack tools, or network conditions.

Probabilistic embeddings address this limitation by representing each input not as a single point, but as a full probability distribution. This inherent uncertainty in the data is addressed and quantified in our work, where an encoder network, with Wasserstein distance f_ϕ, maps an input feature vector x to the parameters of a Gaussian distribution in the latent space Z. This mapping can be expressed as:

$$f_\phi(x) \rightarrow (\mu, \sigma^2) \tag{2}$$

Here, the mean vector (μ) captures the central characteristics of the flow, while the variance vector (σ^2) explicitly models its uncertainty or ambiguity. This provides a richer, more robust representation. The geometric space these distributions inhabit is known as a *probabilistic manifold*, and learning to structure this manifold is a core objective of our framework.

B. Meta Learning for Zero Shot Classification

Zero-shot classification aims to identify instances of a class that were never seen during training. This is impossible for conventional anomaly-based intrusion detection systems to identify, which learn to map inputs to a fixed set of predefined labels.

While SOTA unsupervised anomaly detection methods such as autoencoders and isolation forests [35], [36] can identify outliers with labelled data. However, they typically provide binary decisions, namely normal and anomalous, and lack the semantic capability to distinguish between specific types of novel attacks. Furthermore, they cannot rapidly adapt to categorise these new threats once identified.

Meta-learning offers a solution through a paradigm known as metric-learning [37]–[40]. A prime example is the prototypical network framework. Instead of learning to classify, the model learns a generalizable similarity metric within an embedding space. Training is conducted in an episodic manner. Each episode gives the model a small, labeled support set and an unlabeled query set. It computes a single prototype for each class in the support set, typically by averaging the embeddings of its samples. Query samples are then classified by assigning them the label of the nearest class prototype [41], [42]. This process teaches the model to create an embedding space where proximity corresponds to semantic similarity, a concept that naturally generalizes to new, unseen classes.

C. Interpretable Feature Attention

While neural network and transformer-based model architectures are robust, they are often criticised for being black boxes, making it difficult to understand their decision-making process and developing a lack of trust [43]–[45]. To develop this trust, feature attention is a mechanism that provides a direct form of interpretability by not treating all input features equally. Adding an attention layer learns a set of weights that correspond to the importance of each feature.

D. Advanced Sequential Architectures Design Rationale

The Transformer architecture, while powerful, suffers from quadratic computational complexity ($O(n^2)$) concerning sequence length, making it inefficient for the high-dimensional feature sets standard in NIDS. Mamba, a recent State-Space Model (SSM), overcomes this with linear-time complexity ($O(n)$) [46], [47]. It uses an input-dependent selection mechanism to efficiently model long-range dependencies, making it more suitable for analyzing complex feature vectors and understanding feature heterogeneity. Mixture of Experts (MoE) is a technique for scaling a model's capacity, that is, the number of parameters, without a proportional increase in computational cost, which replaces a single, monolithic network layer with a collection of smaller expert networks and a gating network that routes each input to a small subset of these [48]. This fosters expert specialization, allowing different parts of the model to learn distinct patterns, a feature well-suited to network attacks' diverse and multi-modal nature. Furthermore, based on the modality of the data, gating is performed to activate only the layers required for decision-making for the particular modality and packets, thereby drastically reducing computational efficiency.

III. Related Work

This section reviews the literature across two critical axes for NIDS development. This establishes the context for our work and motivates our novel contributions in addressing these identified gaps.

A. From static classification to zero shot learning

The application of standard Machine Learning (ML) and Deep Learning (DL) models to IoT-based NIDS has been extensively studied and demonstrated high accuracy and a lower false positive rate in classifying attacks within static datasets [49]–[52]. However, this traditional supervised learning approach is fundamentally brittle in real-world security operations. These models are static classifiers, learning fixed decision boundaries for a predefined set of threats. This creates two critical failure modes. First, they are inherently incapable of handling zero-day attacks (concept shift), as they can only misclassify a zero-day threat as one of the known classes they were trained on. Second, their performance degrades over time even on known attacks due to concept drift, as adversaries continually evolve their techniques and network behaviors change.

To overcome this static learning problem, a more adaptive paradigm is required. Recent work, such as *MATEEN*, has focused on building online learning frameworks to handle concept drift in benign traffic, using ensembles of autoencoders to adapt to evolving network behaviors [53]. While effective for adapting to known classes, this approach does not explicitly address the zero-day problem. For that challenge, meta-learning is a more suitable approach [54], [55]. Instead of learning *what* to classify, a meta-learning model learns a generalizable similarity metric, enabling it to recognize and adapt to new classes from a few examples. This makes it a natural fit

for Zero-Shot Learning (ZSL) and Few-Shot Learning (FSL) in the data-scarce security domain. Frameworks based on Model-Agnostic Meta-Learning (MAML) have been proposed to enable rapid adaptation to new threats [56], [57]. However, continual adaptation can lead to catastrophic forgetting, where a model's performance on previously learned attacks degrades; recent systems address this by preserving and recalling older models as needed during online learning [55].

The most similar work is based on prototypical networks, which explicitly apply metric learning to few-shot intrusion detection [58]. This approach provides a crucial baseline, confirming that metric learning is a promising direction for NIDS. More advanced systems like *Helios* have evolved this concept, using Supervised Mixture Prototypical Learning (SMPL) to learn multiple prototypes per class with hardware-aware distance metrics for in-network deployment [59]. However, these methodologies and related meta-learning works reveal two fundamental limitations that *PANDORA* is designed to overcome. First, they rely on deterministic embeddings and fixed distance metrics, which can be insensitive to the subtle, pattern-based signatures of stealthy threats. Second, they operate almost exclusively on flow-level features, Overlooking critical payload and time-window-based contextual information that is essential for detecting sophisticated [60].

Our work directly addresses these identified gaps. We move beyond the deterministic representations used in prior models by introducing *Uncertainty-Aware Probabilistic Embeddings*, which model each flow as a full distribution. Furthermore, we replace the fixed metric with our novel *Probabilistic Manifold Structuring and Distance (PMSD)* Loss. This hybrid objective uses a more appropriate distributional distance for classification and globally structures the embedding manifold, creating a more robust and generalizable framework for accurate *zero-shot detection*.

B. Advanced Architectures for Network Traffic Analysis

The effectiveness of any learning based intrusion detection system is heavily dependent on the power of its underlying encoder architecture to model complex relationships within network traffic features. Early models evolved from simple feed-forward networks to sequential models like LSTMs, CNNs, etc [61]. With its self-attention mechanism, the advent of the transformer architecture marked a significant leap in performance, demonstrating the ability to model global relationships between features for early and accurate intrusion detection [62] [63]. However, self-attention-based transformers are powerful but inefficient for high-dimensional intrusion detection tasks. Recent innovations address this with state space models like Mamba, which efficiently capture long-range dependencies, and Mixture-of-Experts (MoE) architectures, which enhance model capacity with minimal computational cost. However, these approaches alone struggle with the diverse feature types in complex network environments. PANDORA introduces a novel Mamba-MoE integration that combines Mamba's sequential modeling strength with MoE's specialization. This synergy enables efficient, high-capacity analysis suitable for complex traffic on resource-limited devices setting it apart from earlier, less integrated approaches.

A primary criticism of deep learning NIDS is their black box nature, a challenge often addressed post-hoc with resource-intensive explainability models like SHAP and [64] [65] [66]. In contrast, our architecture addresses this challenge by design, directly integrating an *Interpretable Feature Attention* mechanism at the input layer. This provides a robust and computationally efficient method for feature analysis immediately after training. While other works, such as R1DIT, have used attention for malware traffic classification [67], our approach is distinct. It is a dedicated, interpretable layer that precedes the powerful Mamba-MoE encoder. This architectural decision provides us with the opportunity to offer an immediate view into the model's first decision-making process by changing the importance of features dynamically before the complicated sequential analysis takes place. This potentiality is pivotal for grasping the general, class-based verdicts and scrutinizing the features that participate in the differentiation between the attack sub-classes, which is a very detailed analysis not found in the SOTA studies.

IV. The PANDORA Framework: System Design

This section details the architecture of PANDORA, an end-to-end framework designed for zero-shot detection and few-shot adaptation to zero-day attacks in heterogeneous network environments, making it suitable for resource-constrained device deployment. Our design is guided by five core principles: tackling *feature heterogeneity* and *data heterogeneity*, ensuring *interpretability* and *uncertainty-awareness*, and maintaining *computational efficiency*.

A. PANDORA Architecture Overview

PANDORA's pipeline begins by processing an incoming network flow, which is mathematically represented as a D-dimensional feature vector $\boldsymbol{x} \in \mathbb{R}^D$. To contend with the significant feature heterogeneity present in network traffic, where different attack types manifest across different feature sets, we partition $\boldsymbol{x}$ into two logical modalities: a *Temporal* feature vector $\boldsymbol{x}_t \in \mathbb{R}^{D_t}$ and a *Volumetric* feature vector $\boldsymbol{x}_v \in \mathbb{R}^{D_v}$, such that $D = D_t + D_v$ as illustrated in Fig. 1, each vector modality is then fed into a dedicated *probabilistic encoder*. An encoder applies an *interpretable attention mechanism* before processing the data through *Mamba-MoE blocks*. The encoders transform each modality into a probabilistic embedding, parameterized by a mean vector and a variance vector $(\boldsymbol{\mu}_t, \boldsymbol{\sigma}_t^2)$ for the temporal features and $(\boldsymbol{\mu}_v, \boldsymbol{\sigma}_v^2)$ for the volumetric features. A Cross-Attention Fusion module then intelligently merges these representations to produce a final, fused embedding $(\boldsymbol{\mu}_f, \boldsymbol{\sigma}_f^2)$. This final, uncertainty-aware representation is subsequently used within our episodic meta-learning framework, which is optimized by our novel PMSD Loss function to produce either a classification for a known threat or an actionable alert for a zero-day attack. Furthermore, we detail its three core stages: the multi-modal probabilistic encoders that generate uncertainty-aware embeddings, the

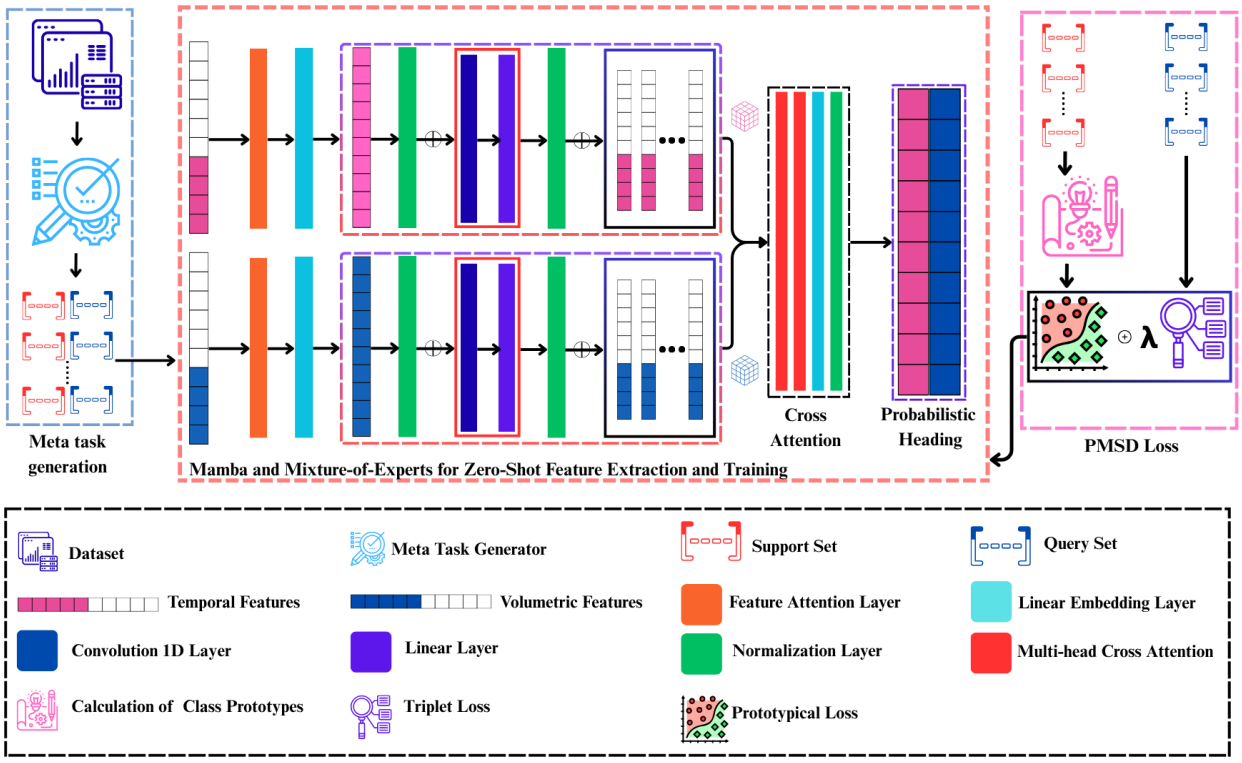

Fig. 1. The PANDORA System Architecture. The diagram illustrates the end-to-end data flow, from the initial partitioning of features into Temporal and Volumetric modalities, through the parallel Probabilistic Encoders and Cross-Attention Fusion, to the final classification within the episodic meta-learning framework.

PMSD loss function that structures the embedding space, and the zero-shot inference framework that provides actionable intelligence.

B. Multi-Modal Probabilistic Encoders

The core of PANDORA's representation learning is its multi-encoder architecture, which transforms the partitioned feature vectors into rich, uncertainty-aware probabilistic embeddings. This process, detailed in Algorithm 1 added in Appendix 1, consists of four key stages for each modality. First, an interpretable attention mechanism weighs the input features. Second, a series of Mamba-MoE blocks encode the attended features to capture complex patterns efficiently. Third, a probabilistic head maps the final representation to a mean and variance. Finally, a cross-attention module fuses the embeddings from both modalities into a single, unified representation. In comparison to traditional methods such as simple concatenation or averaging, cross-attention enables each modality to selectively attend to the most relevant features of the other, resulting in a more discriminative, unified embedding.

1) Interpretable Feature Attention: Before encoding, each modality's feature vector, denoted generally as $\boldsymbol{x}_m$ (where $m \in \{t, v\}$ for temporal and volumetric respectively), is passed through a feature attention mechanism. This module enhances interpretability and allows the model to dynamically focus on the most salient features for a given input. The layer learns a *modality-specific*, learnable parameter vector $\boldsymbol{w}_m \in \mathbb{R}^{D_m}$, where D_m is the modality's dimensionality. This vector is transformed into an attention weight vector $\boldsymbol{\alpha}_m \in \mathbb{R}^{D_m}$ via a softmax function:

$$\alpha_{m,i} = \frac{\exp(w_{m,i})}{\sum_{j=1}^{D_m} \exp(w_{m,j})} \tag{3}$$

This weight vector, representing feature importance, is then applied to the input features $\boldsymbol{x}_m$ using element-wise multiplication (Hadamard product $\odot$ [68]) to produce an attended feature vector $\boldsymbol{x}'_m$:

$$\boldsymbol{x}'_m = \boldsymbol{\alpha}_m \odot \boldsymbol{x}_m \tag{4}$$

2) The Mamba-MoE Encoder Block: o efficiently capture complex dependencies while maintaining a low computational footprint, we construct our encoders from a series of MambaMoEBlocks. The attended feature vector from the previous stage, x'_m, serves as the initial input, which is first projected by an embedding layer to form the initial hidden state, h_0. This state is then processed sequentially through the stack of Mamba-MoE blocks.

Within each block, the input representation h_i is transformed by the Mamba and MoE layers to produce the block's final output, h_{i+1}. Specifically, the MoE layer processes an

intermediate representation to produce its output, y, as a weighted sum of expert outputs:

$$y = \sum_{i=1}^{N} G(h)_i \cdot E_i(h) \tag{5}$$

This intermediate output y is then combined with the output from the Mamba layer and a residual connection to the original input h_i to form the block's final output, h_{i+1}. This complete representation is then passed to the next block. The final hidden state from the last block in the stack is denoted as h_m, which is the final, refined feature representation for the modality. This representation is subsequently passed to the Probabilistic Embedding Head. This architecture enables PANDORA to learn a large number of specialized features without incurring a proportional increase in computational cost, as only the relevant experts are activated for any given input.

3) Probabilistic Embedding Head: The final step of the encoding process is to convert the deterministic feature representation into a probability distribution. After passing through the stack of Mamba-MoE blocks, the final hidden state, $\boldsymbol{h}_m$, is fed into two separate linear heads, $\texttt{fc}_{\mu}$ and $\texttt{fc}_{\log \sigma^2}$. These heads project $\boldsymbol{h}_m$ into the parameters of a Gaussian distribution: a mean vector $\boldsymbol{\mu}_m$, which represents the most likely location of the embedding, and a log-variance vector $\log(\boldsymbol{\sigma}_m^2)$, which represents its uncertainty. Together, these vectors parameterize the final distribution, from which a specific embedding $\boldsymbol{z}_m$ can be sampled: $\boldsymbol{z}_m \sim \mathcal{N}(\boldsymbol{\mu}_m, \text{diag}(\boldsymbol{\sigma}_m^2))$.For subsequent loss calculations, the framework utilizes the full distribution parameters (μ_m and σ_m^2) directly, rather than a single stochastic sample z_m, to ensure training stability and a more robust similarity metric.

4) Cross-Attention Fusion: To create a single, unified representation, a `CrossAttentionFusion` module intelligently merges the probabilistic embeddings from the temporal $(\boldsymbol{\mu}_t, \log \boldsymbol{\sigma}^{2,t})$ and volumetric $(\boldsymbol{\mu}_v, \log \boldsymbol{\sigma}^{2,v})$ encoders. The fusion process first combines the mean vectors. We employ a bidirectional attention mechanism, which we denote as $\mathcal{A}(\mathbf{Q}, \mathbf{K})$, where $\mathbf{Q}$ is a query vector and $\mathbf{K}$ is a key vector. Each modality's mean vector acts as a query to attend to the other. The resulting context vectors are then concatenated (denoted by $\|$) and passed through a Feed-Forward Network (FFN) to produce the final, fused mean vector $\boldsymbol{\mu}_f$:

$$\boldsymbol{\mu}_f = \text{FFN}(\mathcal{A}(\boldsymbol{\mu}_t, \boldsymbol{\mu}_v) \parallel \mathcal{A}(\boldsymbol{\mu}_v, \boldsymbol{\mu}_t)) \tag{6}$$

Assuming the uncertainties of the two modalities are conditionally independent, the final fused variance is their sum. We implement this aggregation in log-space for numerical stability:

$$\log \boldsymbol{\sigma}^{2,f} = \log(\exp(\log \boldsymbol{\sigma}^{2,t}) + \exp(\log \boldsymbol{\sigma}^{2,v})) \tag{7}$$

C. Adaptive Metric Learning with PMSD Loss

PANDORA's ability to generalize to zero-day attacks is driven by its meta-learning framework, which uses zero-shot detection optimized by a novel loss function. Algorithm 1 added in Appendix A-B details the complete episodic training process.

1) Episodic Training Framework: We employ a *prototypical network framework* for training based on *Mamba-MoE*. In each training episode, the model is presented with a small, labeled *support set* $S = \{(\boldsymbol{x}_i^s, y_i^s)\}_{i=1}^{N \times K}$ and an unlabeled *query set* $Q = \{(\boldsymbol{x}_j^q, y_j^q)\}_{j=1}^{N \times Q}$. The model learns by creating a prototype $\boldsymbol{c}_k$ for each class k from the support set and then attempting to classify the query samples based on their similarity distance to these prototypes using the PMSD loss.

2) The PMSD Loss Function: We introduce the *Probabilistic Manifold Structuring and Distance (PMSD) Loss*, a hybrid objective designed to create a robust and well-structured embedding space, identify distance-based clusters effectively, and make adaptive boundary decisions. It is formulated as a weighted sum of two components. *The total loss $\mathcal{L}_{PMSD}$ is the sum of the Wasserstein distance loss $\mathcal{L}_{Wass}$ and the Triplet margin loss $\mathcal{L}_{Triplet}$, weighted by a hyperparameter λ.* However, fixed weights can be highly susceptible to manipulation because of their sensitivity.

$$\mathcal{L}_{\text{PMSD}} = \mathcal{L}_{\text{Wass}} + \lambda \cdot \mathcal{L}_{\text{Triplet}} \tag{8}$$

To eliminate the sensitivity associated with manually tuning the hyperparameter λ, we have several approaches, such as GradNorm [69] and DWA [70]. Furthermore, constraint-based optimisation methods, such as Augmented Lagrangian [71], effectively enforce hard boundaries for adversarial evasion, and we adopted and employed it in Homoscedastic Uncertainty [72]. This aligns with PANDORA's probabilistic nature, allowing the model to dynamically balance manifold structuring and classification based on task-inherent noise. By modelling inherent noise variance (σ^2), it acts as a learnable regularizer that automatically scales task weights based on precision, effectively balancing objectives without the overhead of gradient-based methods and no rigid boundaries. We formulate the *Probabilistic Manifold Structuring and Distance* (PMSD) Loss using a multi-task learning approach based on homoscedastic uncertainty. Rather than giving a fixed weight, we consider the loss balancing to be the process of maximizing the Gaussian likelihood of the model's uncertainty. We add noise parameters that can be adjusted, σ_{Wass} and $\sigma_{Triplet}$, that are linked to the Wasserstein classification loss and the Triplet structuring loss, respectively. The goal is restructured in such a way that the different losses are weighted automatically during backpropagation:

$$\begin{aligned}\mathcal{L}_{PMSD} = \frac{1}{2\sigma_{Wass}^2}\mathcal{L}_{Wass} + \frac{1}{2\sigma_{Triplet}^2}\mathcal{L}_{Triplet} \\ + \log(\sigma_{Wass}) + \log(\sigma_{Triplet})\end{aligned} \tag{9}$$

In this case, the first couple of terms penalize the model error it gives out in accordance with the variance (precision) while the logarithmic parts serve to limit the variance (σ^2) to a non-infinite value which is more or less like negative growth or regularization. During the training, if the Triplet loss at first is high (showing a great difficulty in structuring

the manifold), then the model will pick a higher $\sigma_{Triplet}$ to reduce its gradient effect. As the manifold gets stable, $\sigma_{Triplet}$ is lower, and the model is now ready for the coarse-fine classification through $\mathcal{L}_{Wass}$ with the help of a sharper σ. This way, it is automatically and optimally accomplished that the global manifold structure is not only maintained but even enhanced with local classification accuracy.

a) Distance Component ($\mathcal{L}_{Wass}$): This component is the primary classification loss, computed over the query set. It is based on the negative log-probability of a query sample $\boldsymbol{x}_j$ belonging to its true class y_j. The probability is derived from a softmax function applied over the negative Wasserstein distances between the query sample's embedding, $f_\phi(\boldsymbol{x}_j)$, and each of the N class prototypes, $\boldsymbol{c}_k$.

$$\mathcal{L}_{\text{Wass}} = -\log \frac{\exp(-d(f_\phi(\boldsymbol{x}_j), \boldsymbol{c}_{y_j}))}{\sum_{k=1}^{N} \exp(-d(f_\phi(\boldsymbol{x}_j), \boldsymbol{c}_k))} \tag{10}$$

The distance function d is the squared *2-Wasserstein distance* between two Gaussian distributions, $p_1 \sim \mathcal{N}(\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1)$ and $p_2 \sim \mathcal{N}(\boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2)$. Here, $\boldsymbol{\mu}$ represents the mean vector, $\boldsymbol{\Sigma}$ is the covariance matrix, and $\mathrm{Tr}(\cdot)$ is the trace operator of a matrix.

$$W_2^2(p_1, p_2) = \|\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2\|_2^2 + \mathrm{Tr}(\boldsymbol{\Sigma}_1 + \boldsymbol{\Sigma}_2 - 2(\boldsymbol{\Sigma}_2^{1/2}\boldsymbol{\Sigma}_1\boldsymbol{\Sigma}_2^{1/2})^{1/2}) \tag{11}$$

This offers a true distributional distance, enabling more reliable similarity assessments by accounting for both the mean and the variance of the embeddings. Whereas Euclidean distance and Cosine Similarity only look at the actual values of a point estimate.

b) Structuring Component ($\mathcal{L}_{Triplet}$): This component acts as a regularization loss to structure the embedding manifold. It uses the standard triplet margin loss, which is applied to the mean vectors of the embeddings. For a given query sample's mean embedding (anchor, $\boldsymbol{\mu}_a$), the loss encourages it to be closer to the mean embedding of its correct class (positive, $\boldsymbol{\mu}_p$) than to the mean embedding of any other class (negative, $\boldsymbol{\mu}_n$), by at least a margin m.

$$\mathcal{L}_{\text{Triplet}} = \max(\|\boldsymbol{\mu}_a - \boldsymbol{\mu}_p\|_2^2 - \|\boldsymbol{\mu}_a - \boldsymbol{\mu}_n\|_2^2 + m, 0) \tag{12}$$

This encourages *intra-class compactness* (samples of the same class are close) and *inter-class separability* (samples of different classes are far apart), which cannot be done solely by Wasserstein distance. By combining these losses, PANDORA achieves probabilistic manifold structuring. The Wasserstein component enhances local classification accuracy by considering distributional properties, while the triplet loss imposes a globally consistent structure essential for robust zero-shot generalization.

D. Zero-Shot Detection and Few-Shot Adaptation

1) Zero-Shot Detection: During inference while deployment in section V.E, PANDORA's primary goal is to distinguish known behaviors from novel, unseen threats. When a new network flow $\boldsymbol{x}_{\text{new}}$ arrives, it is first passed through the encoder f_ϕ to generate its probabilistic embedding $(\boldsymbol{\mu}_{\text{new}}, \boldsymbol{\sigma}^2_{\text{new}})$.

The model then calculates its distance to all known class prototypes $\{\boldsymbol{c}_k\}_{k=1}^N$. The minimum of these distances is defined as the `Novelty Score`. This score represents the degree of dissimilarity between the new sample and the closest known class.

$$\text{NoveltyScore} = \min_k d(f_\phi(\boldsymbol{x}_{\text{new}}), \boldsymbol{c}_k) \tag{13}$$

Instead of using a strict, preset constant that frequently results in a large number of false positives, PANDORA applies an *adaptive soft threshold* τ_{soft} that is computed based on the statistical characteristics of the learned manifold. In the validation stage, we determine the empirical distribution $\mathcal{D}_{\text{val}}$ of the Wasserstein distances between benign samples and their corresponding prototypes. We take the 95$^{\text{th}}$ percentile (Q_{95}) of this distribution to be τ_{soft}:

$$\tau_{\text{soft}} = Q_{0.95}(\mathcal{D}_{\text{val}}) + \epsilon \tag{14}$$

This establishes a statistical confidence boundary: scores below τ_{soft} are accepted as natural variance within known classes, while scores exceeding it reject the null hypothesis of "known behavior." The detection rule is thus:

- If NoveltyScore $> \tau_{\text{soft}}$, the sample is considered anomalous and flagged as a potential zero-day attack.

This binary decision—benign vs. anomalous/unknown—is the core of the zero-shot detection mechanism. For flagged attacks, the model can provide additional context to aid cybersecurity experts. It identifies the label of the closest known prototype, even though the sample crossed the novelty threshold. This label is not a definitive classification but rather a hint for analysts.

$$\text{SuggestedLabel} = k^* = \arg\min_k d(f_\phi(\boldsymbol{x}_{\text{new}}), \boldsymbol{c}_k) \tag{15}$$

2) Few-Shot Adaptation: After identifying a zero-day threat, if a few labeled samples of this new attack become available, the model can rapidly adapt. This is achieved through a fine-tuning process outlined in Algorithm 2, which incrementally incorporates the new class information into the model.

V. The TTDFIoTIDS2025 Dataset Creation

Developing supervised and unsupervised learning methods is based on NIDS, which fundamentally depends on the quality and realism of the datasets used for training and evaluation. Public datasets like CICIDS2017 [73], Bot-IoT [74], and the more recent CICIoT2023 [75] have been foundational, providing the community with standardized benchmarks that have enabled reproducible research. The initial methodology for learning for ML-based and DL-based models was done using these datasets that successfully captured the network traffic of their time [76]–[79]. However, as the threat landscape evolves, these datasets become increasingly outdated and insufficient for validating modern, adaptive NIDS. A critical analysis reveals several shared limitations. They often rely on tool-based attack generation developed long ago, where publicly available

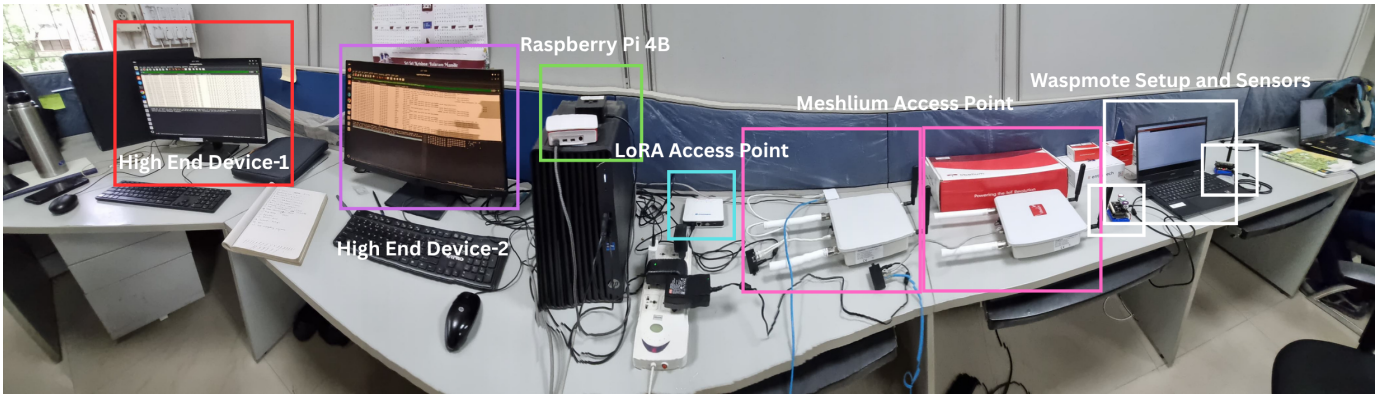

Fig. 2. An experimental testbed for Internet of Things (IoT) network analysis. The setup includes two **High-End PCs** for monitoring and processing, a **Raspberry Pi 4B**, a **LoRa Access Point**, a **Meshlium Access Point**, and a **Waspmote** sensor node. This configuration is designed for developing, testing, and evaluating various IoT communication protocols and security scenarios.

tools are used to create malicious traffic. This approach often produces predictable, easily finger printable attack signatures. Furthermore, many rely on virtualized environments, failing to capture the data heterogeneity of physical IoT hardware and protocols. Finally, attacks are often executed sequentially, which does not reflect the reality of modern, concurrent attack campaigns done by the adversaries. To address these gaps, we developed the *TTDFIoTIDS2025* dataset, a new benchmark designed to evaluate NIDS against modern, complex threat. Furthermore, We compare the existing benchmark dataset with our dataset.

- *Enhanced Realism:* Unlike the simplistic, centralized topology of testbeds like CICIoT2023, CICIDS2017, our setup uses a `heterogeneous wireless infrastructure` (multiple APs, LoRa, mesh) that more accurately reflects complex, real-world IoT environments.
- *Advanced Attack Generation:* We move beyond the predictable, tool-based attacks found in datasets like CICIDS2017 by employing custom, programmatic methods that better mimic sophisticated adversarial behavior.
- *Complex Attack Scenarios:* In contrast to the isolated attacks or tool-based attacks seen in other datasets, our dataset has features of concurrent, multi-threaded attacks from multiple sources to simulate modern threat campaigns.
- *Greater Threat Granularity:* While existing datasets have broad attack categories, seven in CICIoT2023 and 33 at the sub-class level, our dataset provides a richer taxonomy with nine classes and 63 `sub-classes`, enabling a more fine-grained evaluation of detection models.

A. Heterogeneous IoT Testbed and Data Capture

Our dataset was constructed within a physical, heterogeneous IoT testbed. to ensure a high degree of realism and overcome the limitations of prior works. The network was designed to reflect the complexity of modern smart IoT environments illustrated in Fig. 2

- *IoT Edge Nodes:* Three Raspberry Pi 4B devices were deployed as intelligent edge nodes and equipped with advanced sensor endpoints.
- *Wireless Infrastructure:* Two distinct heterogeneous access points providing standard Wi-Fi connectivity.
- *Mesh and LoRaWAN Networks:* Two 2.4GHz Meshlium devices by Libelium and a LoRa gateway provided connectivity for specialized IoT devices.
- *Sensor Endpoints:* A Waspmote device with advanced agricultural sensors communicates through the mesh network.

All devices were connected wirelessly, creating a complex and realistic heterogeneous environment. The testbed included two high-end systems (Intel i7 with 16GB and 47GB RAM) acting as a dedicated attacker and a centralized packet collection server. Crucially, our setup generated external attacks and traffic between the edge nodes (e.g., RPi to RPi), simulating realistic scenarios of compromised devices within the local network.

Raw network packets were stored as PCAP files captured using `tcpdump` and `Wireshark`. We then employed a dual-tool feature extraction process, using both *CICFlowMeter* and *dpkt* to generate a comprehensive set of 83 flow-based and packet-level features, providing a richer and more detailed view of the network traffic than single-tool approaches for feature extraction.

B. Programmatic and Concurrent Attack Generation

We created a Modular Attack Framework in Python to address the drawbacks of static tools with fixed signatures. In our approach, each attack is implemented as an independent, configurable module. Each module is designed around a specific goal (for example, suffering from a DoS attack versus creating and injecting PoCs into a specific application) and is capable of attacking multiple layers of the network stack (Layers 2–7), IoT devices, message brokers, and application layer services. The primary goal of our development was to create a scalable environment that could generate more than

sixty different attack types programmatically, including AI-driven attacks that were programmed rather than manually scripted.

Each incident shares a similar structure through target selection, Payload/Event Generation, Packet/Message Creation, and Flexible Execution Engine. The architecture of all these attacks provides flexibility in how each attack is executed; for example, the attack may take advantage of randomised message headers or utilise a sequence of messages preferred by the Protocol, while the AI-based variant allows payloads to change while executing. Execution backends also allow execution in various modes, such as single-slave, thus allowing the scale of the attacks from independent single-vector events to complex and concurrent multi-vector campaigns. As well as allowing for combining noisy volumetric floods with stealthy application-layer exploits, the architecture closely mimics real-life adversaries, allowing for rich interference traffic for the evaluation of Network Intrusion Detection Systems (NIDS). Further design information about all of the architectural modules as well as full attack definitions, and Execution Logs for all modules are contained within the Appendix A-C.

C. Dataset Profile and Evaluation Strategy

The resulting TTDFIoTIDS2025 dataset is a large-scale collection of over 4.7 million network flows. It features a comprehensive and granular taxonomy of attacks, encompassing over *63 distinct implemented attack sub-classes*. After rigorous data cleaning and merging classes with insufficient samples for robust model training, the final dataset contains 41 distinct malicious sub-classes grouped into nine major categories. This fine-grained ground truth enables a much deeper subclass analysis than previously possible.

We created a particularly challenging subset for our evaluation by merging several related subclasses. This was done intentionally to create more complex, heterogeneous macro-classes that exhibit high intra-class variance. This strategy tests a model's ability to learn a robust representation of a broad threat category rather than simply memorizing the distinct patterns of its sub-classes. The final statistics for the whole dataset and the subset used for evaluation are provided in Table I.

TABLE I
TTDFIoTIDS2025 Dataset Statistics

Statistic	Full Dataset	Evaluation Subset
Total Flows	4,742,843	20,00,000
Benign Flows	183,179	50,000
Malicious Flows	4,559,664	15,00,000
Number of Features	83	66
Major Attack Categories	9	7
Attack Sub-classes	63	30

VI. Evaluation

This section presents and provide a comprehensive empirical evaluation of the PANDORA framework. Our experiments are designed to answer four key questions: (1) Is our new TTDFIoTIDS2025 a more challenging and realistic benchmark than existing datasets? (2) How does PANDORA perform on known and zero-day threat detection compared to state-of-the-art models across multiple network environments? (3) Can we empirically justify PANDORA's novel architectural components? (4) Is the framework practically deployable in a real-time, resource-constrained environment?

A. Experimental Hardware Setup and Training Parameters

- *Training Environment*: Model training was performed on a workstation equipped with an NVIDIA RTX 3070 (8GB VRAM), 64GB RAM, and running Ubuntu 24.04.
- *On-Device Deployment*: The intrusion detection system (IDS) deployment and adaptation loop were tested on a Raspberry Pi 4B with 8GB of RAM.
- *Implementation:* In order to guarantee a strict comparison, we picked PTN-IDS [58] as our leading baseline from the SOTA. Besides, the baseline model structure is taken as transformers employed in [58], [62]. Because no official public repository is available for this particular model, we created an accurate copy of the architecture that fully conforms to the hyperparameters and design specifications provided in the original publication. This means that our performance comparison isolates the algorithmic improvements of PANDORA instead of the differences in implementation.

The architectural and training hyperparameters for the PANDORA model are detailed in Table II. These parameters were chosen to maintain a minimal model footprint, making it suitable for training and deployment on resource-constrained devices.

TABLE II
PANDORA Architectural and Training Hyperparameters

Architectural Parameters		Training Parameters	
Parameter	**Value**	**Parameter**	**Value**
d_{model} (Embedding Dim)	64	Optimizer	Adam
num_blocks (Mamba-MoE)	1	Learning Rate	1×10^{-4}
$num\_experts$ (per MoE)	2	Triplet Margin (m)	1.0
n_{heads} (Fusion Attention)	2	Batch Size	256
$dropout\_rate$	0.5		

Our evaluation is conducted across three distinct datasets to test performance, generalization, and robustness in NIDS. The primary dataset for all core performance, state-of-the-art comparison, and ablation studies is our own TTDFIoTIDS2025, as detailed in Section 5. We use CICIoT2023 for direct comparison with a recent state-of-the-art benchmark. To rigorously test for domain shift, we also evaluate models trained on our IoT dataset against the widely used, non-IoT CICIDS2017 dataset.

B. Validating the TTDFIDSIOT2025 Dataset

Before evaluating our model, we empirically validate our primary claim from Section 5: that the TTDFIoTIDS2025 dataset is a more challenging and realistic benchmark than

existing state-of-the-art datasets. To visually compare the complexity of our dataset against a modern benchmark, we performed a t-SNE dimensionality reduction on stratified samples from both TTDFIoTIDS2025 and CICIoT2023. The proposed dataset evaluation classes are already defined in Section 5, and twenty-three classes that had a minimum of 2500 raw samples for the visualization were taken for CICIoT2023. The results, shown in Fig. 3, clearly confirm our dataset's increased complexity. The t-SNE projection for CICIoT2023 (top) shows relatively well-defined, separable clusters for its various attack classes, and some classes only overlap, with fewer samples. In stark contrast, the projection for TTDFIoTIDS2025 (bottom) reveals a much more entangled feature space. The clusters are less distinct, exhibit significant overlap, and are more fragmented, visually demonstrating the high degree of intra-class variance and feature heterogeneity we engineered through our programmatic, concurrent attack generation. This inherent complexity makes it a significantly more complex challenge for any machine learning model to learn effective decision boundaries.

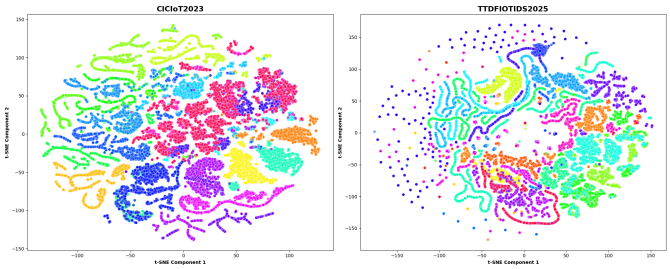

Fig. 3. t-SNE visualization of the feature space for CICIoT2023 and our proposed TTDFIoTIDS2025

C. PANDORA Performance Analysis

In this section, we present a comprehensive performance analysis of the PANDORA framework. To validate the specific contribution of our uncertainty-aware approach, we prioritize a direct comparison against **PTN-IDS** [58]. As the primary existing framework applying Prototypical Networks to the NIDS domain, PTN-IDS serves as the most methodologically relevant baseline. By comparing PANDORA against this specific architecture, we can attribute performance gains directly to our novel PMSD loss and uncertainty modeling, rather than differences in the underlying neural backbone. We also benchmark against MAML-NIDS [30] to provide context within the broader meta-learning landscape Our primary goal is to establish a direct comparison against the state-of-the-art PTN-IDS on the well-established CICIDS2017 dataset, demonstrating PANDORA's superior performance in few-shot adaptation under various scenarios.

1) Direct Comparison with PTN-IDS on CICIDS2017: To ground our evaluation, we replicated the three few-shot classification scenarios from the PTN-IDS paper. In these scenarios, a model is trained on a set of source attacks and then adapted with a few shots of previously unseen target attacks. We compare the performance of the original PTN-IDS results against our PANDORA framework.

The results, summarized in Table III, clearly demonstrate PANDORA's superior adaptation capabilities in simpler scenarios and highlight the challenges of few-shot learning in complex, multi-class environments. In Scenario 1 (Sc.1), where only a single, relatively distinct class (DDoS) is unseen, PANDORA significantly outperforms the PTN-IDS approach across all k-shot values, achieving an F1-score of 0.9811 with just 10 shots.

However, in the more challenging Scenarios 2 and 3, where multiple, more heterogeneous classes (Web Attack, DoS, PortScan) are introduced as unseen threats, PANDORA's 1-shot performance is lower than the baseline. This is expected because PANDORA is based on ZSL and is trying to adapt to a single image during the test. While PTN IDS is based entirely on a few-shot learning paradigm, increasing the number of samples in PANDORA outperforms PTN IDS in Scenarios 2 and 3 for 5-shot and 10-shot adaptation.

TABLE III
SOTA COMPARISON ON CICIDS2017 (FEW-SHOT ADAPTATION, ACC = ACCURACY, F1 = F1-SCORE)

Shots	Metric	Sc.1 (n=1)		Sc.2 (n=2)		Sc.3 (n=3)	
		PTN	**PANDORA**	PTN	**PANDORA**	PTN	**PANDORA**
1-shot	Acc	0.792	**0.890**	0.701	0.680	0.635	0.576
	F1	0.765	**0.914**	0.680	0.586	0.592	0.554
5-shot	Acc	0.910	**0.961**	0.830	**0.897**	0.795	**0.835**
	F1	0.907	**0.967**	0.823	**0.879**	0.779	**0.831**
10-shot	Acc	0.931	**0.969**	0.845	**0.908**	0.819	**0.860**
	F1	0.930	**0.981**	0.837	**0.894**	0.808	**0.879**

The impact of the distance metric is further analyzed in Table IV. The results show that while euclidean distance is the best performer among standard metrics, our PMSD Loss (which uses Wasserstein distance) provides a significant advantage. In the simple Scenario 1 (Sc.1) , PANDORA achieves an F1-score of 0.964, outperforming Euclidean distance by over 6%. This advantage becomes even more pronounced in the complex Scenario 2, where PANDORA's F1-score of 0.8790 is over 5.5% higher than the next best metric. This empirically validates our claim that using a true distributional distance is crucial for accurately modeling the similarity between complex, probabilistic traffic embeddings.

TABLE IV
COMPARISON OF DISTANCE FUNCTIONS IN PTN-IDS VS PANDORA (5-SHOT, CICIDS2017)

Dist.	Sc.1 (n=1)		Sc.2 (n=2)		Sc.3 (n=3)	
	Acc	F1	Acc	F1	Acc	F1
Euclidean	0.910	0.907	0.830	0.823	0.795	0.779
Manhattan	0.886	0.878	0.813	0.804	0.780	0.768
Cosine	0.910	0.905	0.729	0.696	0.769	0.756
Wasserstein	**0.961**	**0.967**	**0.897**	**0.879**	**0.841**	**0.821**

We further assess PANDORA's robustness to domain shift, a key requirement for real-world NIDS. Trained on CICIDS2017

and evaluated on CICIDS2018, PANDORA shows strong cross-domain generalization (Table V). The baseline model fails completely, misclassifying almost all attacks as *Benign*. A standard PTN-IDS with 5-shot finetuning performs better but still struggles with several classes, achieving an accuracy of 0.95.

In contrast, PANDORA's zero-shot variant—evaluated on the target domain without any finetuning—achieves near-perfect recall on *DDoS*, *BruteForce*, and *Bot*, and a recall of 0.92 on *Benign*, resulting in an overall accuracy of 0.99. This demonstrates the inherent robustness of PANDORA's probabilistic manifold and its ability to provide effective protection in unseen environments without retraining.

TABLE V
PERFORMANCE UNDER DOMAIN SHIFT (TRAIN: CICIDS2017, TEST: CICIDS2018)

Method	**Ben**	**DDoS**	**BF**	**Bot**	**Web**	**DoS**	**OA**
Baseline	0.995	0.040	0.000	0.000	0.000	0.000	0.172
PTN-IDS 5-shot	0.707	0.870	0.540	0.569	0.765	0.555	0.668
+ Fine-Tuning	0.826	1.000	0.994	0.959	0.981	0.945	0.951
PANDORA 1-shot	0.225	0.254	1.000	0.872	0.310	0.867	0.565
PANDORA 5-shot	0.321	0.424	1.000	0.864	0.512	0.838	0.669
+Zero Shot	**0.987**	**1.000**	**1.000**	**1.000**	**0.992**	**0.989**	**0.991**

2) Qualitative and Quantitative SOTA Comparison of End-to-End Capabilities: While direct performance metrics such as F1-score are important, they do not fully capture the comprehensive capabilities required for modern intrusion detection systems (NIDS). Table VI provides both qualitative and quantitative comparisons of recent SOTA frameworks [26], [39], [57]–[59], [61], [80] such as across eight critical dimensions. The models are chosen because they belong to meta-learning strategies, prototypical networks, and transformers. The eight dimensions include support for zero-shot detection, few-shot adaptation, edge deployment, handling of complex datasets, and feature attention mechanisms—elements essential for building practical, adaptable, and future-proof security systems.

The F1-scores reported represent each model's best performance in known attack scenarios, for PANDORA, on TTDFIDSIoT2025,CICIoT2023 and CICIDS2017 `0.8203, 0.9182, and 0.9983` in known attack scenarios and zero shot scenarios we achieved `0.7982,0.8983,0.9690`, respectively. These values demonstrate PANDORA's versatility across varying operational settings and its superior generalization, especially in challenging zero-shot environments. A detailed breakdown of these deployment-specific results is discussed in Section E.

This comparative analysis highlights that while many existing models target isolated challenges in NIDS, such as classification or few-shot learning, *PANDORA stands alone in offering a holistic, end-to-end framework*. It encompasses the entire lifecycle: from training on complex, real-world datasets to performing real-time detection on constrained edge devices. Its architecture enables rapid adaptation to emerging threats and robust zero-day attack identification—key capabilities that are lacking in traditional models. .

D. Ablation Studies: Justifying PANDORA's Architecture

To empirically validate our architectural choices, we conducted a series of ablation studies on the TTDFIoTIDS2025 dataset. These experiments isolate the contribution of each novel component of the PANDORA framework, from our PMSD Loss function to the Mamba-MoE encoder and the interpretable feature attention mechanism. The ablation studies were done on 5 baseline epochs.

1) Impact of the PMSD Loss (Distance Metric Analysis): To evaluate its impact, we compared the full PANDORA model against variants using only one component of the loss, as well as a baseline using the standard Euclidean distance.

The results, presented in Table VII, are particularly revealing. On the simpler, more established datasets (CICIDS2017 and CICIoT2023), the standard Euclidean distance performs competitively. This is expected, as the classes in these datasets have relatively well-defined, unimodal distributions that can be effectively separated by a simple distance metric. However, on our more complex TTDFIoTIDS2025 dataset, which is characterized by high intra-class fragmentation, the performance of the Euclidean distance model drops significantly, achieving a Macro F1-score of only 0.6488.

In contrast, our full PMSD Loss demonstrates robust performance across all datasets, achieving the highest Macro F1-score (0.7001) on the challenging TTDFIoTIDS2025 benchmark. This empirically validates our central claim: for modern, complex network environments with heterogeneous and multi-modal attack classes, a simple distance metric is insufficient. The combination of a true distributional distance (Wasserstein) and a manifold structuring term (Triplet Loss) is necessary to learn a robust and generalizable representation.

2) Impact of Mamba-MoE Architecture: Table VIII demonstrates the clear superiority of the Mamba-MoE approach. While the Transformer-based model has slightly fewer parameters and a marginally faster inference time, it suffers a catastrophic drop in performance, with its Macro F1-score falling by over 30 percentage points on the CICIDS2017 dataset and by over 45 points on the more complex IoT datasets. This highlights the inability of the standard Transformer's self-attention mechanism to effectively model the complex, long-range dependencies in high-dimensional network traffic data. In contrast, the Mamba-MoE architecture's combination of linear-time efficiency and expert specialization proves to be far more effective, providing a significant performance gain for a negligible increase in computational cost.

3) Model Interpretability (Feature Attention Analysis): The top 10 most important features for each modality and dataset are presented in Table XIV, added in the Appendix.

The results provide valuable insights into the model's decision-making process. On the CICIDS2017 dataset, the model learns to prioritize a mix of temporal features, such as *Bwd IAT Min* and *Flow Duration*, and volumetric features, like *Bwd Packet Length Min* and *SYN Flag Count*. However,

TABLE VI
QUALITATIVE AND QUANTITATIVE SOTA COMPARISON OF NIDS FRAMEWORK CAPABILITIES.

Model	Best F1-Score	Zero-Shot	Few-Shot Adaptation	Edge Deploy.	Complex Dataset	Feat. Attn.	Dataset(s) Used	Classes Eval.
RF	0.7823	✗	✗	✓	✗	✗	Various	Various
MAML-NIDS [26]	0.9219	✗	✓	✗	✗	✗	CICIDS2017	5 Major
MAML-NIDS Online [55]	0.9285	✗	✓	✗	✗	✗	CICIDS2017	5 Major
Transformer-NIDS [61]	0.9310	✗	✗	✗	✗	✗	NSL-KDD	5 Major
PTN-IDS [58]	0.9310	✗	✓	✗	✗	✗	CICIDS2017	7 Major
MASiNet [57]	0.9412	✗	✓	✗	✗	✗	UNSW-NB15, NSL-KDD	10 Major
Helios [59]	0.9578	✗	✗	✗	✗	✗	CICIoT2023 TON-IoT	12 Sub-classes
MTH-IDS [80]	0.9630	✗	✗	✗	✗	✗	CICIDS2017	20 Sub-classes
PANDORA Zero Shot (Ours)	**0.7982,0.8983,0.9690**	✓	✓	✓	✓	✓	**TTDF, CICIoT2023, CICIDS2017**	**41,23,7 Sub Classes**
PANDORA Known (Ours)	**0.8203, 0.9182, and 0.9983**	✓	✓	✓	✓	✓	**TTDF, CICIoT2023, CICIDS2017**	**41,23,7 Sub Classes**

TABLE VII
ABLATION STUDY OF THE PMSD LOSS FUNCTION ACROSS DATASETS

Config	CIC17		CICIoT23		TTDF25	
	F1	AUC	F1	AUC	F1	AUC
Full PMSD	**0.859**	**0.991**	**0.822**	**0.990**	**0.700**	**0.986**
Wasserstein only	0.815	0.987	0.654	0.974	0.671	0.983
Triplet only	0.826	0.987	0.678	0.976	0.678	0.983
Euclidean only	0.842	0.989	0.821	0.986	0.649	0.983

TABLE VIII
ABLATION OF ENCODER ARCHITECTURE ACROSS DATASETS

Dataset	Config	Params	Time (ms)	F1	AUC
CIC17	**Mamba-MoE**	196,938	**0.0179**	**0.836**	**0.991**
	Transformer	183,552	0.1595	0.509	0.927
CICIoT23	**Mamba-MoE**	195,378	**0.0181**	**0.813**	**0.985**
	Transformer	183,552	0.1165	0.356	0.919
TTDF25	**Mamba-MoE**	195,378	**0.0176**	**0.813**	**0.985**
	Transformer	183,552	0.1669	0.356	0.919

when trained on the more modern IoT datasets, the model's focus shifts. On CICIoT2023, it learns that the simple *Rate* and *Duration* features are highly indicative of an attack, while on our more complex TTDFIoTIDS2025 dataset, it prioritizes more nuanced features like *Bwd Blk Rate Avg* and *Down Up Ratio*. This demonstrates that the attention mechanism is not static; it successfully learns to adapt its focus to the most salient features of a given network environment, providing a direct and transparent view into its reasoning.

E. Real-Time On-Device Performance

A key limitation of many academic NIDS is the practicality gap—models that perform well offline often become prohibitively expensive for real-time use on constrained devices. To address this, we evaluated a quantized version of PANDORA directly on a Raspberry Pi, demonstrating its suitability for edge-level security deployment.

Deployment Strategy and Baseline Adaptation: We have modified all baseline encoders to run in the lightweight deployment framework that was used for PANDORA since existing baselines usually do not provide implementations that are fit for edge devices. By doing this, we make it clear that all differences in performance are due to the architectural designs such as Mamba-MOE and Transformers and not to the differences in engineering or optimization.

The results are summarized in Table IX, presented in a *Baseline / **Ours*** format.

TABLE IX
ON-DEVICE PERFORMANCE: BASELINE / **PANDORA**. **BOLD** DENOTES MAMBA-MOE RESULTS.

Metric	CICIDS2017 (Baseline / **Ours**)	CICIoT2023 (Baseline / **Ours**)	TTDF-IoT-2025 (Baseline / **Ours**)
Throughput (Flows/sec)	0.52 / **2.03**	1.12 / **4.26**	0.98 / **3.82**
Latency (ms/flow)	215.10 / **58.20**	142.50 / **37.34**	135.20 / **34.76**
CPU Usage (%)	345.20 / **118.70**	355.80 / **125.60**	340.50 / **120.20**
Memory Usage (MB)	85.60 / **24.48**	82.40 / **24.16**	84.10 / **24.30**

Analysis: As demonstrated in Table IX, PANDORA (highlighted in **bold**) significantly outperforms the baseline adaptation across all metrics.

- **Throughput and Latency:** The baseline architecture struggles on the resource-constrained device, achieving throughputs approximately 3–4$\times$ lower than PANDORA. For instance, on CICIoT2023, PANDORA processes **4.26 flows/sec** compared to 1.12 flows/sec for the baseline. This is driven by the linear complexity $O(L)$ of our Mamba-MoE encoder, whereas baselines relying on attention or convolutions incur far higher costs.
- **Resource Utilization:** PANDORA maintains a lightweight footprint, using only ≈ 24 MB memory and $\approx 120\%$ CPU (1.2 cores). Baselines nearly saturate the device (CPU $> 340\%$, Memory > 80 MB), making them impractical for edge IoT gateways.

This confirms PANDORA's architectural improvements with real-world efficacy, thus enabling broad use in IoT security that requires real-time response. PANDORA exhibits equal and stable performance in real-time across all datasets. Detection latency at CICIDS2017 tightly fluctuated around 62 ms, whereas the more diverse IoT datasets (CICIoT2023 and TTDF-IoT-IDS2025) reveal slightly lower median values (58 ms and 61 ms respectively) with wider discrepancies across the data points. Even in difficult data conditions, detections are consistently performed within the real-time limit. The combination of the extremely high zero-day *Novelty Detection Rate* of (98–100%) associated with these results suggests that PANDORA is both durable and accessible in practice for edge-based intrusion detection.

VII. Discussion and Limitations

A. Robustness to Novel Attack Behaviors

PANDORA adopts an adaptive soft threshold (τ_{soft}) based on the 95^{th} percentile of the benign probabilistic manifold. This design improves detection of both high-volume and stealthy threats.

Volumetric Attacks: The compression of all interarrival times into extremely small values and a reduction of temporal variability σ_t^2 to virtually zero by high-rate flooding (UDP/DDoS) causes a variance profile that differs fundamentally from a benign flow of IoT data. In addition, the Wasserstein distance enables the identification of this variational collapse, even though the high-rate flood protocol may match that of a benign flow, thereby eliminating the need for manual threshold adjustment to detect high-rate floods.

Stealthy Attacks: These attacks (XSS, SQLi, backdoors) tend to imitate practical function methods; however, they alter the correlations between deeper features of the system, thus creating samples with low probability relative to the learned manifold. While Euclidean metrics resist change when comparing these abnormal samples to standard samples, PANDORA's uncertainty-aware embeddings greatly enhance the degree of deviation while retaining their high sensitivity.

B. Handling Severe Class Imbalance

Real IoT networks exhibit extreme imbalance with benign-to-attack ratios such as 100:1, 1000:1 and 10,000:1. PANDORA mitigates this by using episodic meta-learning, which constructs balanced N-way, K-shot episodes irrespective of raw distribution. As validated in TABLE X, prototype learning is driven by feature modality rather than frequency, allowing PANDORA to sustain stable performance under increasing imbalance.

TABLE X
AUC and accuracy for different shot adaptations under extreme imbalance for CICIDS2017

Shot	AUC (100:1)	AUC (1000:1)	AUC (10000:1)	Acc (100:1)	Acc (1000:1)	Acc (10000:1)
1-Shot	0.95	0.97	0.97	0.94	0.93	0.93
5-Shot	0.97	0.97	0.97	0.99	0.99	0.99
10-Shot	0.97	0.97	0.96	0.98	0.97	0.97

C. Scalability Analysis for High-Density Deployments

While physical validation on large-scale clusters is cost-prohibitive, we numerically estimate the scalability of the PANDORA system based on the empirical resource profile shown in Table IX of this paper. We have analyzed scalability in two different dimensions based on IoT testbed standards [81].

1) Horizontal Scalability (Distributed Inference): PANDORA's ultra-lightweight memory footprint (≈24 MB) allows for a fully distributed deployment strategy. PANDORA's approach allows for deployment directly onto Edge Nodes thus eliminating Processing Bottlenecks experienced with Centralized NIDS via Gateway. Processing Load for N devices in a network is $O(1)$ per node versus $O(N)$ in a centralized setup. As a result, if the deployment stays distributed, growing the network to 1,000+ nodes does not result in a decrease in individual detection latency (maintained at ≈37ms).

2) Vertical Efficiency (Algorithmic Complexity): In scenarios that require centralized aggregation such as an IoT Gateway handling traffic from multiple sensors, PANDORA's architectural advantage becomes critical.

- *Linear vs. Quadratic Growth:* Standard Transformer-based NIDS exhibit quadratic complexity $O(L^2)$ with respect to sequence length and traffic volume. An increase in traffic density by 10× leads to a 100× increase in required processing.
- *PANDORA's Advantage:* The Mamba-MoE backbone operates with linear complexity $O(L)$. As shown in the ablation study (Table VIII), this yields a throughput of 4.26 Flow/sec on a Raspberry Pi 4B, outperforming baselines.

The linear scaling of PANDORA allows it to maintain real-time response even under high-concurrency attack storms, whereas traditional architectures become saturated.

D. Mitigating False Positives and Threshold Independence

One of the major challenges involving zero-shot NIDS is differentiating between standard (benign) and abnormal (high-variance) network traffic and detecting zero-day attacks. To avoid improperly labeling normal traffic as attacks (false positives), we developed two mechanisms:

1) Decoupling Detection from Prototype Creation: We strictly separate the *detection* phase from the *adaptation* phase to prevent model corruption.

- *Detection:* Samples exceeding the threshold τ_{soft} are flagged using the Novelty Score (Eq. 13). These are treated as potential threats but do not instantly update the model.
- *Adaptation:* New class prototypes are only created after accumulating multiple confirmed samples in the support set. This avoids spurious classes caused by isolated false positives.

2) Scenario-Independent Statistical Thresholding: Instead of a rigid hard threshold, PANDORA uses an adaptive soft threshold τ_{soft} (Eq. 14).

- Anchoring τ_{soft} to the 95th percentile of validation Wasserstein distances statistically controls the False Positive Rate (FPR), independent of deployment environment.
- High-variance benign flows are accepted, while structural inconsistencies from zero-day attacks remain detectable.

VIII. Conclusion

PANDORA is a pioneering framework for the detection of zero-day attacks in resource-constrained IoT edge environments. PANDORA goes a step further by replacing static embeddings and similarity metrics which are typically used in

the traditional approaches with a probabilistic and uncertainty-aware metric which is trained through the PMSD loss, thus allowing the detection of hidden and known threats with reliability. The Mamba-MoE encoder, which is lightweight, ensures that the ability to represent is very high while the computational cost is very low, thus allowing real-time inference on the device. PANDORA is not only able to demonstrate its strengths in standard benchmarks and on the freshly created TTDFIOTIDS2025 dataset but also in zero-shot generalization, few-shot adaptability, and domain shift resilience. The interpretability and runtime efficiency of the system are additional factors that make it an excellent choice for embedded cybersecurity. In summary, PANDORA provides a systematic, complete solution for the next generation of intrusion detection in dynamic, adversarial IoT environments.

ACKNOWLEDGEMENT

The project (TTDF/6G/517) is sponsored by the Telecom Technology Development Fund (TTDF), Department of Telecommunication (DoT), Government of India. We gratefully acknowledge their support and encouragement.

REFERENCES

[1] E. H. Houssein, M. A. Othman, W. M. Mohamed, and M. Younan, "Internet of things in smart cities: Comprehensive review, open issues, and challenges," *IEEE Internet of Things Journal*, vol. 11, no. 21, pp. 34 941–34 952, 2024.

[2] A. Alanhdi and L. Toka, "A survey on integrating edge computing with ai and blockchain in maritime domain, aerial systems, iot, and industry 4.0," *IEEE Access*, vol. 12, pp. 28 684–28 709, 2024.

[3] C. Bollineni, M. Sharma, A. Hazra, P. Kumari, S. Manipriya, and A. Tomar, "Iot for next-generation smart healthcare: A comprehensive survey," *IEEE Internet of Things Journal*, pp. 1–1, 2025.

[4] S. S. Mahadik, P. M. Pawar, and R. Muthalagu, "Heterogeneous iot (hetiot) security: techniques, challenges and open issues," *Multimedia Tools and Applications*, vol. 83, no. 12, pp. 35 371–35 412, 2024. [Online]. Available: https://doi.org/10.1007/s11042-023-16715-w

[5] E. Pagliari, L. Davoli, and G. Ferrari, "Harnessing communication heterogeneity: Architectural design, analytical modeling, and performance evaluation of an iot multi-interface gateway," *IEEE Internet of Things Journal*, vol. 11, no. 5, pp. 8030–8051, 2024.

[6] I. Gamiz, C. Regueiro, O. Lage, E. Jacob, and J. Astorga, "Challenges and future research directions in secure multi-party computation for resource-constrained devices and large-scale computations," *International Journal of Information Security*, vol. 24, no. 1, pp. 27–, 2024. [Online]. Available: https://doi.org/10.1007/s10207-024-00939-4

[7] S. Hudda and K. Haribabu, "A review on wsn based resource constrained smart iot systems," *Discover Internet of Things*, vol. 5, no. 1, pp. 56–, 2025. [Online]. Available: https://doi.org/10.1007/s43926-025-00152-2

[8] S. A. Ghazi Asgar and N. Reddy, "Analysis of misconfigured iot mqtt deployments and a lightweight exposure detection system," 03 2025.

[9] S. Halder, M. Rafiqul Islam, Q. Mamun, A. Mahboubi, P. Walsh, and M. Zahidul Islam, "A comprehensive survey on ai-enabled secure social industrial internet of things in the agri-food supply chain," *Smart Agricultural Technology*, vol. 11, p. 100902, 2025. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2772375525001352

[10] V. M. U, V. Babu Kumaravelu, V. K. C, R. A, S. Chinnadurai, R. Venkatesan, H. Hai, and P. Selvaprabhu, "Ai-powered iot: A survey on integrating artificial intelligence with iot for enhanced security, efficiency, and smart applications," *IEEE Access*, vol. 13, pp. 50 296–50 339, 2025.

[11] S. Trilles, S. S. Hammad, and D. Iskandaryan, "Anomaly detection based on artificial intelligence of things: A systematic literature mapping," *Internet of Things*, vol. 25, p. 101063, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2542660524000052

[12] T. Sowmya and E. Mary Anita, "A comprehensive review of ai based intrusion detection system," *Measurement: Sensors*, vol. 28, p. 100827, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2665917423001630

[13] A. S. Jacobs, R. Beltiukov, W. Willinger, R. A. Ferreira, A. Gupta, and L. Z. Granville, "Ai/ml for network security: The emperor has no clothes," in *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 1537–1551. [Online]. Available: https://doi.org/10.1145/3548606.3560609

[14] C. Liu, B. Chen, W. Shao, C. Zhang, K. K. L. Wong, and Y. Zhang, "Unraveling attacks to machine-learning-based iot systems: A survey and the open libraries behind them," *IEEE Internet of Things Journal*, vol. 11, no. 11, pp. 19 232–19 255, 2024.

[15] R. Vadisetty, "Adaptive machine learning-based intrusion detection systems for iot era," in *Proceedings of 5th International Ethical Hacking Conference*, M. Chakraborty, S. P. Chakrabarty, A. Penteado, and V. E. Balas, Eds. Singapore: Springer Nature Singapore, 2025, pp. 251–273.

[16] L. Yang, A. Shami, G. Stevens, and S. de Rusett, "Lccde: A decision-based ensemble framework for intrusion detection in the internet of vehicles," in *GLOBECOM 2022 - 2022 IEEE Global Communications Conference*, 2022, pp. 3545–3550.

[17] S. Racherla, P. Sripathi, N. Faruqui, M. Alamgir Kabir, M. Whaiduzzaman, and S. Aziz Shah, "Deep-ids: A real-time intrusion detector for iot nodes using deep learning," *IEEE Access*, vol. 12, pp. 63 584–63 597, 2024.

[18] S. Najafli, A. Toroghi Haghighat, and B. Karasfi, "Taxonomy of deep learning-based intrusion detection system approaches in fog computing: a systematic review," *Knowledge and Information Systems*, vol. 66, no. 11, pp. 6527–6560, Nov 2024. [Online]. Available: https://doi.org/10.1007/s10115-024-02162-y

[19] R. Jordaney, K. Sharad, S. K. Dash, Z. Wang, D. Papini, I. Nouretdinov, and L. Cavallaro, "Transcend: Detecting concept drift in malware classification models," in *26th USENIX Security Symposium (USENIX Security 17)*. Vancouver, BC: USENIX Association, Aug. 2017, pp. 625–642. [Online]. Available: https://www.usenix.org/conference/usenixsecurity17/technical-sessions/presentation/jordaney

[20] L. Yang, W. Guo, Q. Hao, A. Ciptadi, A. Ahmadzadeh, X. Xing, and G. Wang, "CADE: Detecting and explaining concept drift samples for security applications," in *30th USENIX Security Symposium (USENIX Security 21)*. USENIX Association, Aug. 2021, pp. 2327–2344. [Online]. Available: https://www.usenix.org/conference/usenixsecurity21/presentation/yang-limin

[21] M. Abdullahi, H. Alhussian, N. Aziz, S. Jadid Abdulkadir, Y. Baashar, A. Ahmed Alashhab, and A. Afrin, "A systematic literature review of concept drift mitigation in time-series applications," *IEEE Access*, vol. 13, pp. 119 380–119 410, 2025.

[22] F. Alotaibi and S. Maffeis, "Rasd: Semantic shift detection and adaptation for network intrusion detection," in *the 39th International Conference on ICT Systems Security and Privacy Protection (SEC 2024)*. Springer, 2024, pp. 16–30.

[23] S. Layeghy, M. Baktashmotlagh, and M. Portmann, "Di-nids: Domain invariant network intrusion detection system," *Knowledge-Based Systems*, vol. 273, p. 110626, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0950705123003763

[24] J. Yan, Y. Cheng, Q. Wang, L. Liu, W. Zhang, and B. Jin, "Transformer and graph convolution-based unsupervised detection of machine anomalous sound under domain shifts," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 8, no. 4, pp. 2827–2842, 2024.

[25] Y. Xian, B. Schiele, and Z. Akata, "Zero-shot learning - the good, the bad and the ugly," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[26] C. Lu, X. Wang, A. Yang, Y. Liu, and Z. Dong, "A few-shot-based model-agnostic meta-learning for intrusion detection in security of internet of things," *IEEE Internet of Things Journal*, vol. 10, no. 24, pp. 21 309–21 321, 2023.

[27] X. Gao, K. Chen, Y. Zhao, P. Zhang, L. Han, and D. Zhang, "A zero-shot learning-based detection model against zero-day attacks in iot," in *2024 9th International Conference on Electronic Technology and Information Science (ICETIS)*, 2024, pp. 309–314.

[28] K. Saurabh, U. Singh, A. Mishra, R. Vyas, and O. Vyas, "Zero-shot based hybrid neural network system for enhancing zero-day attack detection," in *2024 IEEE 21st India Council International Conference (INDICON)*, 2024, pp. 1–6.

[29] J. Cai, Y. Zhang, S. Wang, J. Fan, and W. Guo, "Wasserstein embedding learning for deep clustering: A generative approach," *IEEE Transactions on Multimedia*, vol. 26, pp. 7567–7580, 2024.

[30] X. Liu, Y. Bai, Y. Lu, A. Soltoggio, and S. Kolouri, "Wasserstein task embedding for measuring task similarities," *Neural Networks*, vol. 181, p. 106796, 2025. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0893608024007202

[31] S. K. R. Mallidi and R. R. Ramisetty, "Advancements in training and deployment strategies for ai-based intrusion detection systems in iot: a systematic literature review," *Discover Internet of Things*, vol. 5, no. 1, p. 8, Jan 2025. [Online]. Available: https://doi.org/10.1007/s43926-025-00099-4

[32] S. Ullah, J. Ahmad, M. A. Khan, M. S. Alshehri, W. Boulila, A. Koubaa, S. U. Jan, and M. M. Iqbal Ch, "Tnn-ids: Transformer neural network-based intrusion detection system for mqtt-enabled iot networks," *Computer Networks*, vol. 237, p. 110072, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1389128623005170

[33] L. D. Manocchio, S. Layeghy, W. W. Lo, G. K. Kulatilleke, M. Sarhan, and M. Portmann, "Flowtransformer: A transformer framework for flow-based network intrusion detection systems," *Expert Systems with Applications*, vol. 241, p. 122564, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S095741742303066X

[34] U. C. Akuthota and L. Bhargava, "Transformer-based intrusion detection for iot networks," *IEEE Internet of Things Journal*, vol. 12, no. 5, pp. 6062–6067, 2025.

[35] A. Singh and J. Jang-Jaccard, "Autoencoder-based unsupervised intrusion detection using multi-scale convolutional recurrent networks," 2022. [Online]. Available: https://arxiv.org/abs/2204.03779

[36] M. M. Rahman, S. A. Shakil, and M. R. Mustakim, "A survey on intrusion detection system in iot networks," *Cyber Security and Applications*, vol. 3, p. 100082, 2025. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2772918424000481

[37] C. Lu, X. Wang, A. Yang, Y. Liu, and Z. Dong, "A few-shot-based model-agnostic meta-learning for intrusion detection in security of internet of things," *IEEE Internet of Things Journal*, vol. 10, no. 24, pp. 21 309–21 321, 2023.

[38] Y. Ge, T. Li, and Q. Zhu, "Scenario-agnostic zero-trust defense with explainable threshold policy: A meta-learning approach," in *IEEE INFOCOM 2023 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, 2023, pp. 1–6.

[39] Z. Wang, Y. Lu, W. Wu, Y. Lu, and H. Wang, "A few-shot and anti-forgetting network intrusion detection system based on online meta learning," in *GLOBECOM 2024 - 2024 IEEE Global Communications Conference*, 2024, pp. 3877–3882.

[40] C. Rajathi and P. Rukmani, "Hybrid learning model for intrusion detection system: A combination of parametric and non-parametric classifiers," *Alexandria Engineering Journal*, vol. 112, pp. 384–396, 2025. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1110016824012651

[41] F. Martinez, M. Mapkar, A. Alfatemi, M. Rahouti, Y. Xin, K. Xiong, and N. Ghani, "Redefining ddos attack detection using a dual-space prototypical network-based approach," in *2024 33rd International Conference on Computer Communications and Networks (ICCCN)*, 2024, pp. 1–9.

[42] Z. Li, C. Xu, K. Deng, and C. Liu, "A subspace-based few-shot intrusion detection system for the internet of things," *Frontiers of Information Technology & Electronic Engineering*, vol. 26, no. 6, pp. 862–876, Jun 2025. [Online]. Available: https://doi.org/10.1631/FITEE.2400556

[43] V. Ravi, R. Chaganti, and M. Alazab, "Deep learning feature fusion approach for an intrusion detection system in sdn-based iot networks," *IEEE Internet of Things Magazine*, vol. 5, no. 2, pp. 24–29, 2022.

[44] N. F. Syed, M. Ge, and Z. Baig, "Fog-cloud based intrusion detection system using recurrent neural networks and feature selection for iot networks," *Computer Networks*, vol. 225, p. 109662, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S138912862300107X

[45] M. Sarhan, S. Layeghy, N. Moustafa, M. Gallagher, and M. Portmann, "Feature extraction for machine learning-based intrusion detection in iot networks," *Digital Communications and Networks*, vol. 10, no. 1, pp. 205–216, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2352864822001754

[46] A. Gu and T. Dao, "Mamba: Linear-time sequence modeling with selective state spaces," *arXiv preprint arXiv:2312.00752*, 2023.

[47] T. Dao and A. Gu, "Transformers are SSMs: Generalized models and efficient algorithms through structured state space duality," in *International Conference on Machine Learning (ICML)*, 2024.

[48] S. Mu and S. Lin, "A comprehensive survey of mixture-of-experts: Algorithms, theory, and applications," 2025. [Online]. Available: https://arxiv.org/abs/2503.07137

[49] F. Wei, H. Li, Z. Zhao, and H. Hu, "xNIDS: Explaining deep learning-based network intrusion detection systems for active intrusion responses," in *32nd USENIX Security Symposium (USENIX Security 23)*. Anaheim, CA: USENIX Association, Aug. 2023, pp. 4337–4354. [Online]. Available: https://www.usenix.org/conference/usenixsecurity23/presentation/wei-feng

[50] D. K. Nkashama, A. Soltani, J.-C. Verdier, M. Frappier, P.-M. Tardif, and F. Kabanza, "Robustness evaluation of deep unsupervised learning algorithms for intrusion detection systems," 2023. [Online]. Available: https://arxiv.org/abs/2207.03576

[51] L. Yang and A. Shami, "Towards autonomous cybersecurity: An intelligent automl framework for autonomous intrusion detection," in *Proceedings of the Workshop on Autonomous Cybersecurity (AutonomousCyber '24), ACM Conference on Computer and Communications Security (CCS) 2024*, Salt Lake City, UT, USA, 2024, pp. 1–11.

[52] I. Mohammed Sayem, M. Islam Sayed, S. Saha, and A. Haque, "Enids: A deep learning-based ensemble framework for network intrusion detection systems," *IEEE Transactions on Network and Service Management*, vol. 21, no. 5, pp. 5809–5825, 2024.

[53] F. Alotaibi and S. Maffeis, "Mateen: Adaptive ensemble learning for network anomaly detection," in *Proceedings of the 27th International Symposium on Research in Attacks, Intrusions and Defenses*, ser. RAID '24. New York, NY, USA: Association for Computing Machinery, 2024, p. 215–234. [Online]. Available: https://doi.org/10.1145/3678890.3678901

[54] H. Tsang, M. A. Salahuddin, N. Limam, and R. Boutaba, "Meta-atmos+: A meta-reinforcement learning framework for threat mitigation in software-defined networks," in *2023 IEEE 48th Conference on Local Computer Networks (LCN)*, 2023, pp. 1–9.

[55] Z. Wang, Y. Lu, W. Wu, Y. Lu, and H. Wang, "A few-shot and anti-forgetting network intrusion detection system based on online meta learning," in *GLOBECOM 2024 - 2024 IEEE Global Communications Conference*, 2024, pp. 3877–3882.

[56] U. Zukaib, X. Cui, C. Zheng, M. Hassan, and Z. Shen, "Meta-ids: Meta-learning-based smart intrusion detection system for internet of medical things (iomt) network," *IEEE Internet of Things Journal*, vol. 11, no. 13, pp. 23 080–23 095, 2024.

[57] Y. Wu, G. Lin, L. Liu, Z. Hong, Y. Wang, X. Yang, Z. L. Jiang, S. Ji, and Z. Wen, "Masinet: Network intrusion detection for iot security based on meta-learning framework," *IEEE Internet of Things Journal*, vol. 11, no. 14, pp. 25 136–25 146, 2024.

[58] N. Niknami, V. Mahzoon, and J. Wu, "Ptn-ids: Prototypical network solution for the few-shot detection in intrusion detection systems," in *2024 IEEE 49th Conference on Local Computer Networks (LCN)*, 2024, pp. 1–9.

[59] Z. Shi, D. Zhao, Y. Zhu, G. Xie, Q. Li, and Y. Jiang, "Helios: Learning and adaptation of matching rules for continual in-network malicious traffic detection," in *Proceedings of the ACM on Web Conference 2025*, ser. WWW '25. New York, NY, USA: Association for Computing Machinery, 2025, p. 2319–2329. [Online]. Available: https://doi.org/10.1145/3696410.3714742

[60] S. Wali, Y. A. Farrukh, I. Khan, and N. D. Bastian, "Meta: Toward a unified, multimodal dataset for network intrusion detection systems," *IEEE Data Descriptions*, vol. 1, pp. 50–57, 2024.

[61] J. Liu, M. Simsek, M. Nogueira, and B. Kantarci, "Multidomain transformer-based deep learning for early detection of network intrusion," in *GLOBECOM 2023 - 2023 IEEE Global Communications Conference*. IEEE, Dec. 2023, p. 6056–6061. [Online]. Available: http://dx.doi.org/10.1109/GLOBECOM54140.2023.10436976

[62] M. S. Alshehri, O. Saidani, F. S. Alrayes, S. F. Abbasi, and J. Ahmad, "A self-attention-based deep convolutional neural networks for iiot networks intrusion detection," *IEEE Access*, vol. 12, pp. 45 762–45 772, 2024.

[63] Y. Li and Y. Su, "A network traffic anomaly classification model based on self-attention mechanism and convolutional gated recurrent unit," *IEEE Access*, vol. 13, pp. 134 804–134 821, 2025.

[64] Y. Wang, M. A. Azad, M. Zafar, and A. Gul, "Enhancing ai transparency in iot intrusion detection using explainable ai

techniques," *Internet of Things*, p. 101714, 2025. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2542660525002288

[65] L. A. C. Ahakonye, C. I. Nwakanma, J. M. Lee, and D.-S. Kim, "Machine learning explainability for intrusion detection in the industrial internet of things," *IEEE Internet of Things Magazine*, vol. 7, no. 3, pp. 68–74, 2024.

[66] B. Sharma, L. Sharma, C. Lal, and S. Roy, "Explainable artificial intelligence for intrusion detection in iot networks: A deep learning based approach," *Expert Systems with Applications*, vol. 238, p. 121751, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417423022534

[67] O. Barut, Y. Luo, P. Li, and T. Zhang, "R1dit: Privacy-preserving malware traffic classification with attention-based neural networks," *IEEE Transactions on Network and Service Management*, vol. 20, no. 2, pp. 2071–2085, 2023.

[68] G. G. Chrysos, Y. Wu, R. Pascanu, P. H. Torr, and V. Cevher, "Hadamard product in deep learning: Introduction, advances and challenges," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 47, no. 8, pp. 6531–6549, 2025.

[69] Z. Chen, V. Badrinarayanan, C.-Y. Lee, and A. Rabinovich, "Gradnorm: Gradient normalization for adaptive loss balancing in deep multitask networks," 2018. [Online]. Available: https://arxiv.org/abs/1711.02257

[70] S. Liu, E. Johns, and A. J. Davison, "End-to-end multi-task learning with attention," 2019. [Online]. Available: https://arxiv.org/abs/1803.10704

[71] T. Krauß, J. König, A. Dmitrienko, and C. Kanzow, "Automatic adversarial adaption for stealthy poisoning attacks in federated learning."

[72] A. Kendall, Y. Gal, and R. Cipolla, "Multi-task learning using uncertainty to weigh losses for scene geometry and semantics," 2018. [Online]. Available: https://arxiv.org/abs/1705.07115

[73] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," in *International Conference on Information Systems Security and Privacy*, 2018. [Online]. Available: https://api.semanticscholar.org/CorpusID:4707749

[74] N. Koroniotis, N. Moustafa, E. Sitnikova, and B. Turnbull, "Towards the development of realistic botnet dataset in the internet of things for network forensic analytics: Bot-iot dataset," 2018. [Online]. Available: https://arxiv.org/abs/1811.00701

[75] E. C. P. Neto, S. Dadkhah, R. Ferreira, A. Zohourian, R. Lu, and A. A. Ghorbani, "Ciciot2023: A real-time dataset and benchmark for large-scale attacks in iot environment," *Sensors*, vol. 23, no. 13, 2023. [Online]. Available: https://www.mdpi.com/1424-8220/23/13/5941

[76] Z. K. Maseer, R. Yusof, N. Bahaman, S. A. Mostafa, and C. F. M. Foozy, "Benchmarking of machine learning for anomaly based intrusion detection systems in the cicids2017 dataset," *IEEE Access*, vol. 9, pp. 22 351–22 370, 2021.

[77] S. S. Panwar, Y. P. Raiwani, and L. S. Panwar, "An intrusion detection model for cicids-2017 dataset using machine learning algorithms," in *2022 International Conference on Advances in Computing, Communication and Materials (ICACCM)*, 2022, pp. 1–10.

[78] M. Houichi, F. Jaidi, and A. Bouhoula, "A novel framework for attack detection and localization in smart cities," in *2024 17th International Conference on Security of Information and Networks (SIN)*, 2024, pp. 1–9.

[79] S. Hizal, U. Cavusoglu, and D. Akgun, "A novel deep learning-based intrusion detection system for iot ddos security," *Internet of Things*, vol. 28, p. 101336, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2542660524002774

[80] L. Yang, A. Moubayed, and A. Shami, "Mth-ids: A multitiered hybrid intrusion detection system for internet of vehicles," *IEEE Internet of Things Journal*, vol. 9, no. 1, pp. 616–632, 2022.

[81] N. Chaabouni, M. Mosbah, A. Zemmari, C. Sauvignac, and P. Faruki, "Network intrusion detection for iot security based on learning techniques," *IEEE Communications Surveys Tutorials*, vol. 21, no. 3, pp. 2671–2701, 2019.

TABLE XI
5-SHOT PERFORMANCE ON TTDFIOTIDS2025

Class	Prec.	Recall	F1	Support
Amplification	1.000	0.596	0.747	10000
Benign	0.955	0.740	0.834	10000
Bruteforce_Ftppatator	0.984	0.948	0.965	10000
Bruteforce_Scripted	0.000	0.000	0.000	10000
Ddos_Ackfrag	1.000	0.998	0.999	10000
Ddos_Attachdetach	0.654	0.978	0.784	10000
Ddos_Authtau	0.993	0.998	0.995	10000
Ddos_Connect	0.848	1.000	0.918	10000
Ddos_Goldeneye	0.981	0.754	0.853	10000
Ddos_Httpflood	0.564	0.107	0.180	10000
Ddos_Hulk	0.509	0.826	0.630	10000

TABLE XII
5-SHOT PERFORMANCE ON TTDFIOTIDS2025.

Class	Prec.	Recall	F1	Support
Ddos_Udpfrag (New)	**0.934**	**0.996**	**0.964**	**49990**
Ddos_Udplag	0.976	1.000	0.988	10000
Ddos_Udpplain	0.166	0.028	0.048	10000
Dos_Slowlorris	0.984	0.999	0.992	10000
Dos_Tcpconnectflood	0.991	0.484	0.650	10000
Malware_Backdoor	0.585	0.548	0.566	10000
Recon_Dnsenumeration	0.845	0.596	0.699	10000
Recon_Portscan	0.484	1.000	0.652	10000
Recon_Vulnerabilityscan	0.998	0.996	0.997	10000
Reflection_Cldap	0.631	0.999	0.774	10000
Reflection_Ssdp	0.745	0.999	0.854	10000
Accuracy			**0.800**	**369990**
Macro Avg	**0.776**	**0.777**	**0.756**	**369990**
Weighted Avg	**0.793**	**0.800**	**0.778**	**369990**

APPENDIX A
ADDITIONAL RESULTS AND SIMULATION

A. TTDFIoTIDS2025 Results

B. Algorithms for PANDORA

This section presents the core procedures used in PANDORA: (i) episodic probabilistic metric-space training and (ii)

Algorithm 1 PANDORA Episodic Training Step

Input: Support Set $S = \{(\boldsymbol{x}_i^s, y_i^s)\}$, Query Set $Q = \{(\boldsymbol{x}_j^q, y_j^q)\}$, Model f_ϕ, Margin m, Weight λ
Output: Total Loss $\mathcal{L}_{\text{PMSD}}$
$(\boldsymbol{\mu}^s, \log \boldsymbol{\sigma}^{2,s}) \leftarrow f_\phi(S); \quad (\boldsymbol{\mu}^q, \log \boldsymbol{\sigma}^{2,q}) \leftarrow f_\phi(Q)$
for $k = 1$ to N **do**
  $S_k \leftarrow \{i \mid y_i^s = k\}$
  $\boldsymbol{\mu}_{c_k} \leftarrow \frac{1}{|S_k|} \sum_{i \in S_k} \boldsymbol{\mu}_i^s$
  $\boldsymbol{\Sigma}_{c_k} \leftarrow \text{diag}\left(\frac{1}{|S_k|} \sum_{i \in S_k} \exp(\log \boldsymbol{\sigma}_i^{2,s})\right)$
end for
$\mathcal{L}_{\text{Wass}} \leftarrow 0$
for each $(\boldsymbol{x}_j^q, y_j^q) \in Q$ **do**
  $\boldsymbol{\Sigma}_j^q \leftarrow \text{diag}(\exp(\log \boldsymbol{\sigma}_j^{2,q}))$
  Compute $d_k = W_2^2\big(\mathcal{N}(\boldsymbol{\mu}_j^q, \boldsymbol{\Sigma}_j^q), \mathcal{N}(\boldsymbol{\mu}_{c_k}, \boldsymbol{\Sigma}_{c_k})\big)$
  $\mathcal{L}_{\text{Wass}} \leftarrow \mathcal{L}_{\text{Wass}} - \log\left(\frac{\exp(-d_{y_j^q})}{\sum_k \exp(-d_k)}\right)$
end for
$\mathcal{L}_{\text{PMSD}} \leftarrow \mathcal{L}_{\text{Wass}}$
return $\mathcal{L}_{\text{PMSD}}$

few-shot adaptation for novel classes.

a) Algorithm 1: Episodic Training Step.: This algorithm computes class prototypes in the probabilistic metric space and evaluates Wasserstein distances between query distributions and class distributions. The resulting metric-based softmax forms the episodic loss used to train PANDORA.

b) Algorithm 2: Few-Shot Adaptation.: This procedure fine-tunes the trained model using joint sampling of old and new classes, enabling robust incorporation of novel classes with minimal data. It performs gradient updates using the same PMSD loss used during base training.

Algorithm 2 PANDORA Few-Shot Adaptation

Input: Trained Model f_ϕ, Training Data D_{train}, Adaptation Set D_{adapt}, Episodes E_{adapt}, LR η'
Output: Fine-tuned Model $f_{\phi'}$
$\phi' \leftarrow \phi$
$D_{\text{combined}} \leftarrow D_{\text{train}} \cup D_{\text{adapt}}$
for $e = 1$ to E_{adapt} **do**
    Sample support set S and query set Q from D_{combined}
    Compute $\mathcal{L}_{\text{PMSD}}$ using Algorithm 1
    $\phi' \leftarrow \phi' - \eta' \nabla_{\phi'} \mathcal{L}_{\text{PMSD}}$
end for
return $f_{\phi'}$

C. Programmatic, Adaptive, and Concurrent Attack Generation Framework

This appendix provides an extended overview of our end-to-end attack generation, execution, monitoring, and detection pipeline. The figures illustrate the workflow across reconnaissance, adaptive reinforcement-learning (RL)–driven attacks, multi-vector volumetric exploitation, and the monitoring components used to capture ground-truth PCAP and CSV traces. Together, this framework ensures that the dataset reflects realistic, diverse, and dynamically evolving adversarial behavior.

Algorithm 3 Generic Attack Module Structure

Input: Target T, objective O, rate r, mode m
Initialize module parameters
while attack is active **do**
    if AI payload enabled **then**
        $P \leftarrow$ GeneratePayload_AI()
    else
        $P \leftarrow$ TemplateLibrary.Get(O)
    end if
    $pkt \leftarrow$ BuildPacket(T, P)
    if header randomization enabled **then**
        $pkt \leftarrow$ MutateHeaders(pkt)
    end if
    SendPacket(pkt)
    WaitInterval(mode m, rate r)
end while
Log results and cleanup

a) Explanation:: The algorithm formalizes the behavior of all attack modules in the framework. Packet generation, evasion, timing, and concurrency are embedded into a unified execution structure.

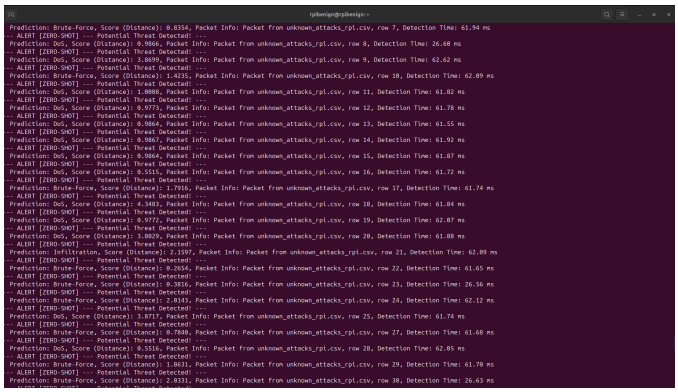

Fig. 4. A real-time intrusion detection system (IDS) deployed on a single device, analyzing packets from a CSV file. It successfully identifies and alerts on various incoming threats, including `Brute-Force`, `DoS`, and `Infiltration` attacks, providing prediction scores and detection times for each event.

Real-Time Intrusion Detection: Figure 4 illustrates the real-time IDS used to validate generated attack traffic. The IDS processes CSV windows and triggers alerts with confidence scores and timestamps, confirming the semantic correctness of traffic samples.

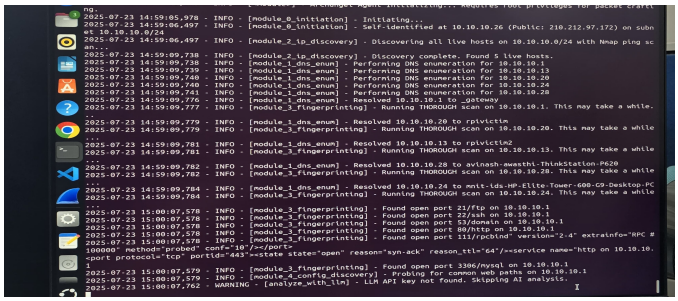

Fig. 5. Automated reconnaissance phase discovering live hosts and performing DNS enumeration and service fingerprinting.

Automated Reconnaissance and Target Profiling: Before generating attacks, our framework performs automated host discovery, DNS enumeration, banner grabbing, and service fingerprinting (Figure 5). These results enable protocol-aware attack selection.

RL-Driven Adaptive Attacks: Figures 6 and 7 show RL-driven attackers that observe system latency and degradation signals to adjust flooding intensity. This evolves non-static, non-repetitive traffic patterns difficult for signature-based IDS.

Multi-Vector Attack Execution: The RL agent schedules sequential multi-vector attacks, observing system transitions (*Normal*→*Degraded*→*Critical*) and refining its attack policy.

Monitoring and Ground-Truth Capture: During attacks, victim hosts collect traffic via `tcpdump` (Figure 8), producing synchronized PCAPs aligned with IDS alerts and system metrics.

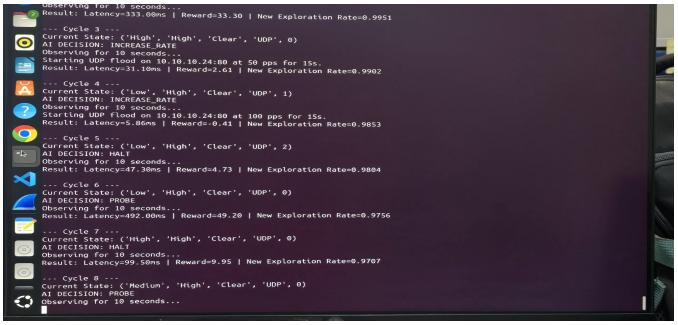

Fig. 6. An RL agent executing an adaptive UDP flood attack, choosing actions based on network latency and reward feedback.

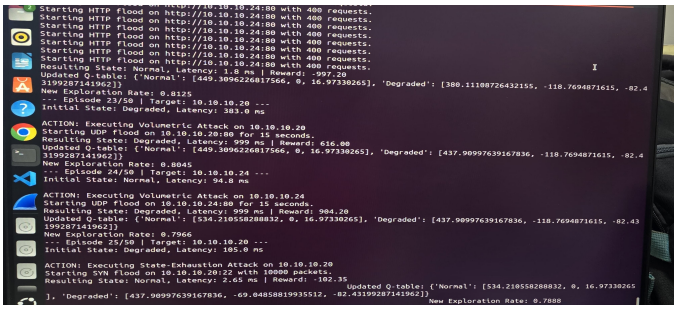

Fig. 7. RL coordination of HTTP, UDP, and SYN floods with Q-table updates after each state transition.

Modular Attack Framework Architecture: Our Modular Attack Framework supports 63+ attacks using a unified template encompassing target selection, payload generation, packet construction, and concurrent execution backends.

D. Model Interpretability based on Feature Attention

In our analysis, we resorted to PANDORA's feature-level attention weights to uncover the attributes that had the greatest impact on the inference. As depicted in Table XIV, we present the ten superior features from the datasets CICIDS2017, CICIoT2023, and TTDF25, thus underlining the consistency of PANDORA's learned representations.

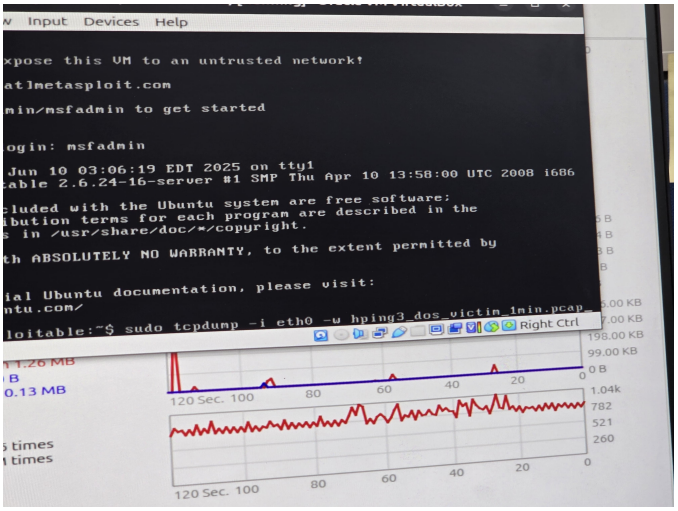

Fig. 8. Packet capture during DoS simulation using `tcpdump`, with traffic spike visualization.

TABLE XIII
ATTACK TAXONOMY ORGANIZED BY DESIGN GOALS

Goal	Attacks
Amplification & Reflection	`Amplification Chargen, Cldap, Dns, Mssql, Netbios, Ntp, Portmap, Snmp, Ssdp, Tftp, Reflection, Reflection Cldap, Reflection Ssdp`
Credential Attacks	`Bruteforce, Dictionary, Ftp Patator, Scripted, Ssh Patator, Web Login`
DDoS Attacks	`Ddos, Ack Frag, Attach Detach, Auth Tau, Golden Eye, Http Flood, Hulk, Icmp Flood, Loic Http, Loic Udp, Pshack, Publish Flood, Rstfin, Sctp, Syn Ack, Syn Flood, Synonymous Ip, Tcpflood, Udp Flood, Udp Frag, Udp Lag, Udp Plain`
Classic DoS	`Dos, Slowlorris, Tcp Connect Flood, Udp Flood`
Malware	`Backdoor, Dropbox, RAT, Reverse Shell, Self Propagating Payloads`
Reconnaissance	`Dns Enumeration, Host Discovery, Os Scan, Ping Sweep, Port Scan, Vulnerability Scan`
Spoofing	`Spoofing, Arp, Dns`
Web Attacks	`Browser Hijacking, CMI, SQLi, Uploading, XSS`

TABLE XIV
TOP 10 MOST IMPORTANT FEATURES LEARNED BY ATTENTION (ALL DATASETS)

Rank	CIC17	CICIoT23	TTDF25
		Temporal Features	
1	Bwd Iat Min	Rate	Bwd Blk Rate Avg
2	Fwd Iat Max	Srate	Fwd Iat Max
3	Idle Max	Duration	Flow Duration
4	Flow Duration	Drate	Bwd Iat Mean
5	Fwd Iat Std	Flow Duration	Fwd Blk Rate Avg
6	Bwd Iat Std	Iat	Bwd Iat Std
7	Fwd Iat Min	–	Fwd Iat Tot
8	Flow Iat Min	–	Bwd Iat Min
9	Idle Min	–	Flow Iat Min
10	Active Std	–	Fwd Iat Min
		Volumetric Features	
1	Bwd Pkt Len Min	Avg	Down Up Ratio
2	Tot Fwd Pkts	Psh Flag Number	Ack Flag Cnt
3	Bwd Pkt Len Std	Min	Pkt Len Min
4	Max Pkt Len	Icmp	Fwd Act Data Pkts
5	Flow Pkts Per Sec	Max	Bwd Byts B Avg
6	Cwe Flag Cnt	Https	Bwd Pkts B Avg
7	Syn Flag Cnt	Tot Size	Bwd Pkt Len Std
8	Bwd Pkt Len Mean	Rst Count	Protocol
9	Fwd Psh Flags	Ack Count	Fwd Psh Flags
10	Pkt Len Var	Syn Flag Number	Bwd Pkt Len Max

APPENDIX B
ARTIFACT APPENDIX

The artifact appendix provides a self-contained roadmap for setting up and evaluating the artifact for the paper: PANDORA: Lightweight Adversarial Defense for Edge IoT using Uncertainty-Aware Metric Learning.

PANDORA-AE is the official artifact package, which contains the source code, required datasets, a pre-configured Docker image, and reproduction scripts to re-run the experiments described in the PANDORA paper. The artifact reproduces central claims (SOTA comparison, ablations, few-shot/zero-shot results, hyperparameter sensitivity) and produces the tables/figures reported in the manuscript.

A. Description & Requirements

This section provides the necessary information to recreate the experimental setup used to run the PANDORA artifact.

1) How to access: The artifact is available as a pre-built Docker image and a source code repository: https://github.com/avinash-developer/Pandora_NDSS_2026.git

- **Docker Image:** `docker pull` ghcr.io/anonyomousartifactsresearch/pandora-ae:1.0
- **Source Repository:** https://github.com/avinash-developer/Pandora_NDSS_2026.git (requires `git lfs pull` to fetch datasets)
- **Complete Dataset and Code Versions:** The complete artifacts and raw dataset are available at: https://doi.org/10.5281/zenodo.17881774.

2) Hardware dependencies: The hardware requirements are split into three levels. Level 1 is the minimum required for the Artifact Evaluation Committee (AEC) to validate the primary claims.

- **Level 1: Primary Artifact Evaluation (CPU-Only, via Docker) (Highly Recommeded)**
  - **CPU:** x86-64 multi-core processor (e.g., Intel Core i5/i7, AMD Ryzen 5/7, or equivalent).
  - **RAM:** 16 GB
  - **Disk Space:** 30 GB (for Docker image and datasets)
- **Level 2: On-Device Performance Evaluation (Optional)**
  - **Device:** Raspberry Pi 4B (8GB RAM model).
  - **OS:** Ubuntu Server installed on the Raspberry Pi.
  - (Note: We also provide a version that can be executed on standard Ubuntu/Windows systems to demonstrate functionality without the specific hardware, though performance metrics will differ.)
- **Level 3: Full Model Training from Scratch (Optional)**
  - **GPU:** NVIDIA GPU with 8GB VRAM (e.g., NVIDIA RTX 3070).
  - **RAM:** 64 GB.
  - **OS:** Ubuntu 24.04.

3) Software dependencies: All primary dependencies are managed within the provided Docker container.

- **Host OS:** Any modern Linux, Windows, or macOS that can run Docker.
- **Primary Environment:** Docker.
- **Windows Host:** Windows Subsystem for Linux (WSL) is required to run Docker.
- **Local (Manual) Setup:** Python 3.12, `pip`, `venv`, and `git-lfs`. See `requirements.txt` in the repository for all Python packages.

4) Benchmarks: The artifact uses pre-processed, CSV-ready variants of the following public datasets, which are included in the repository via Git LFS (in the `data/` directory):

- `CICIDS2017_Ready.csv`
- `CICIot2023_Ready.csv`
- `TTDF_IoT_IDS_2025_Ready_Again.csv`
- `CICIDS2018_Domain_Shift-Ready_-Again.csv`

B. Artifact Installation & Configuration

The recommended method for installation is to use the pre-built Docker image, which contains all necessary dependencies.

1) Pull the pre-built Docker image:
   `docker pull` ghcr.io/anonyomousartifactsresearch/pandora-ae:1.0
2) Create a local directory to store results:
   `mkdir -p $(pwd)/results`
3) Run the container, mounting the results directory. For CPU-only evaluation (Level 1):
   `docker run -it --name pandora-ae -v $(pwd)/results:/app/results` ghcr.io/anonyomousartifactsresearch/pandora-ae:1.0
4) (Optional) If you have an NVIDIA GPU (Level 3):
   `docker run -it --name pandora-ae -v $(pwd)/results:/app/results --gpus all` ghcr.io/anonyomousartifactsresearch/pandora-ae:1.0

Once inside the container, you will be at the `/app` directory, ready to run the experiments. While running the code in a container, try rerunning the code. Either close or remove the previous container, or rename the container.

C. Experiment Workflow

The experimental workflow is automated via shell scripts located in the `reproduce_results/` directory. Each script corresponds to a significant claim or figure in the paper. The scripts handle training, evaluation, and generation of result CSVs and plots, saving all outputs to the `/app/results` directory (which is mounted to your host machine). A master script, `run_all.sh`, is provided to execute all experiments sequentially.

D. Major Claims

This artifact supports the following major claims made in the paper:

- **(C1):** PANDORA outperforms the state-of-the-art (PTN-IDS) on CICIDS2017 under few-shot settings. This is proven by experiment **(E1)** whose results are reported in **Table III, IV**.

- **(C2):** The components of the proposed PMSD loss (Wasserstein, Triplet, Euclidean) are all critical to performance. This is proven by the ablation experiment **(E2)** whose results are reported in **Table VII**.
- **(C3):** The Mamba-MoE encoder architecture provides a superior trade-off of performance and efficiency compared to a standard Transformer encoder for this task. This is proven by the ablation experiment **(E3)** whose results are reported in **Table VIII**.
- **(C4):** PANDORA achieves high performance in known, zero-shot, and few-shot scenarios on modern large-scale IoT datasets (CICIoT2023, TTDFIOTIDS2025) given in **Table VI**. This is proven by experiment **(E4)**.
- **(C5):** The model's performance is stable across a range of key hyperparameters. This is proven by experiment **(E5)**, which generates the hyperparameter sensitivity plots.
- **(C6):** PANDORA can adapt to domain shifts (CICIDS2017) and perform zero-shot learning on unseen attacks (CICIDS2018) given in **Table V**. This is proven by experiment **(E6)**.

E. Evaluation

All commands below assume you are inside the Docker container at the `/app` directory. All scripts will write their outputs (logs, CSVs, plots) to sub-folders within `/app/results/`. The expected outputs are CSV files containing metrics that match (within a negligible margin) the tables referenced below. All commands are detailed in the `readme.md` file.

a) (E0) Run All.: This script executes all experiments (E1–E6) sequentially. *Execution:* `bashrun_all.sh` *Results:* Check `/app/results/` for all output folders (E1–E6).

b) (E1) SOTA Comparison (C1).: Reproduces **Table III and Table IV**. *Execution:* Run the following scripts: `reproduce_pandora_vs_ptnids_cicids2017_s1.sh`, `reproduce_pandora_vs_ptnids_cicids2017_s2.sh`, `reproduce_pandora_vs_ptnids_cicids2017_s3.sh`. *Results:* See `results/pandora_s*_cicids2017/` for CSVs and plots.

c) (E2) Loss Ablation (C2).: Reproduces **Table VII**. *Execution:* `bashreproduce_results/reproduce_loss_ablation_cicids2017.sh` *Results:* See `results/loss_ablation_*/ablation_results.csv`.

d) (E3) Architecture Ablation (C3).: Reproduces **Table VIII**. *Execution:* `bashreproduce_results/reproduce_architecture_ablation_ciciot2023.sh` *Results:* See `results/arch_ablation_*/architecture_ablation_results.csv`.

e) (E4) Main Results (C4).: Reproduces main paper results on modern datasets for **Table VI**. *Execution:* `bashreproduce_results/reproduce_pandora_ciciot2023_s_random.sh` and `bashreproduce_results/reproduce_pandora_ttdfiot_s_random.sh`. *Results:* Check respective `results/` folders for logs and CSVs.

f) (E5) Hyperparameter Sensitivity (C5).: Reproduces sensitivity plots. *Execution:* `bashreproduce_results/reproduce_hyperparams_ablation_cicids2017.sh` *Results:* See `results/` folder for generated `.png` plots.

g) (E6) Domain Shift (C6).: Reproduces domain shift and ZSL given in **Table V**. *Execution:* `bashreproduce_results/reproduce_cicids2017_domain_shift.sh` and `bashreproduce_results/reproduce_cicids2018_zsl.sh`. *Results:* Check respective `results/` folders for logs and CSVs.

F. Customization

The core training and evaluation scripts in `scripts/` can be modified to test custom data or model variations. However, for artifact evaluation, we recommend using the provided `reproduce_results/` scripts, which hardcode the paper's settings.

G. Notes

- **Result Variation:** The tables compare metrics like **F1 Score, Macro Avg, and Overall Accuracy**. While results on a **CPU** will always be consistent (deterministic), the results reported in the paper are an **average of multiple runs computed over both GPU and CPU**. Therefore, exact comparisons during reproduction **might vary by 1-2%**.
- **Mamba vs. Transformer (C3):** Some claims made in the paper regarding Mamba performing better than Transformer (Table VIII) are validated in this artifact using a CPU-based evaluation. Reviewer comments regarding this comparison will be addressed in the next iteration of the paper.
- **Table IX** presents the results generated in (C1, C4, and C6). The middle graph is saved in the results folder, and the feature importance is based on the 10 most recurring features in the model's decision.