Distributed Denial of Service Attack Vector Identification in IoT Networks Using Wavelet Transform and Hybrid Deep Learning

Tohid Behdadnia, *Member, IEEE,* Klaas Thoelen, *Member, IEEE,* Joaquin Garcia-Alfaro, *Senior Member, IEEE,* Geert Deconinck, *Senior Member, IEEE*

***Abstract*—Robust defense mechanisms against DDoS attacks often require not only precise detection of anomalous flows, but also real-time identification and quantification of the implicated attack vector. This has become highly challenging with the increasing scale and complexity of IoT networks, especially under widespread traffic encryption. Most prior works in this area are either limited to binary classification or rely on packet-level signatures that become inaccessible or highly altered by encryption protocols such as TLS, DTLS, or VPN.**

This paper presents a novel methodology and multi-class classification framework for DDoS attack-vector identification and quantification across different OSI layers, using an approach that is agnostic to payload content and packet-level signatures and instead focuses exclusively on traffic timing characteristics. The proposed framework utilizes packet inter-arrival times (PIATs) as its sole observable, enabling not only the detection of attack traffic but also the identification of attack vectors and quantification of attack intensity, even in the presence of cryptographic overheads and the resulting increase in temporal uncertainty. Discrete wavelet transforms are applied to PIAT sequences to capture both transient and persistent timing patterns across multiple scales. These patterns are then processed by a hybrid deep learning architecture that integrates convolutional layers, BiLSTM units, and an attention mechanism.

Comprehensive experiments are conducted on the IEEE P2668-MLIDD and Bot-IoT datasets, augmented by incorporating cryptographic processing overhead to imitate encrypted network conditions. The results demonstrate that the framework achieves high accuracy in both attack vector and intensity classification, and consistently sustains robust performance across different network and communication settings.

***Index Terms*—Anomaly detection, Deep learning, Distributed denial-of-service attack, Feature extraction, Internet of Things, Network security, Wavelet transform.**

I. Introduction

As technology, industry, and society advance toward greater levels of automation, the adoption of the internet of things (IoT) and its integration into digital communication networks have experienced rapid growth. Many IoT devices rely on critical communication infrastructures such as cellular networks, satellite communication systems, Wi-Fi networks, etc. to facilitate real-time data transmission, remote management, and enhanced connectivity between devices, enabling them to perform a wide range of functions from simple data collection to complex automated decision-making. However, the heterogeneity of IoT devices, coupled with the lack of common standards for their integration into digital networks, introduces significant security vulnerabilities, thus increasing their susceptibility to compromise and exploitation [1]. Compromised IoT devices can form a botnet (i.e., a network of massively infected devices controlled by an attacker [2]), which are capable of launching large-scale cyberattacks, such as distributed denial of service (DDoS), targeting the very communication networks, systems, and servers they depend on or interact with. These attacks have the potential to not only compromise the functionality of the affected systems but also induce cascading disruptions across other critical systems and services that depend on their availability [3], [4].

To counter DDoS attacks, a common approach involves detecting and filtering traffic flows originating from malicious sources, which are often intermingled with benign traffic. Although numerous mechanisms for detecting DDoS attacks in IoT networks have been well-documented and widely implemented [5], the growing complexity of these networks and the evolving sophistication of associated attack methods significantly limit the effectiveness and optimal deployment of existing solutions. This limitation is primarily due to exposure to broader attack vectors originating from various layers of the open systems interconnection (OSI) model. In particular, these attack vectors include, but are not limited to, ICMP floods at the network layer, TCP SYN floods and UDP floods at the transport layer, and MQTT publish floods, HTTP floods, and CoAP floods at the application layer, each having unique attack patterns designed to overwhelm specific network components and disrupt service availability [6]–[9]. Given these diversities and complexities, mere detection of an attack is insufficient for mounting an effective response. Instead, for a robust response (e.g., deploying adaptive traffic filtering and rate limiting), rapid and accurate identification of the attack vectors and quantification of their intensities are imperative [10].

Recent advancements in artificial intelligence (AI), particularly through machine learning (ML) methodologies, have substantially enhanced the capabilities of AI-driven anomaly

Tohid Behdadnia, Klaas Thoelen, and Geert Deconinck are with the Research Group ELECTA, Department of Electrical Engineering, KU Leuven, Leuven, 3000 Leuven, Belgium, and also with the EnergyVille, 3600 Genk, Belgium (e-mail: tohid.behdadnia@kuleuven.be; klaas.thoelen@kuleuven.be; geert.deconinck@kuleuven.be).

Joaquin Garcia-Alfaro is with the SAMOVAR laboratory, Télécom Sud-Paris, Institut Polytechnique de Paris, Palaiseau 91120, France (e-mail: joaquin.garcia_alfaro@telecom-sudparis.eu).

detection algorithms in IoT networks. These algorithms are now more effective at identifying and quantifying attack vectors within broad and heterogeneous IoT environments, building on decades of foundational research. The performance and reliability of some of these algorithms have been thoroughly evaluated in several key studies [10]–[13]. The ML models employed in these studies predominantly rely on a set of meticulously crafted features, which can be extracted from packet headers, packet payloads, protocol characteristics, and traffic flow metrics. However, the accessibility of these features and the reliability of the information provided by them in different network and communication settings are rarely discussed in previous works. To elaborate, the deployment of security measures such as transport layer security (TLS) [14], datagram transport layer security (DTLS) [15], and virtual private network (VPN) [16], while recommended for enhancing security and privacy, introduces significant complexities into the process of extracting features from communication traffic. This complication arises mainly from packet encryption, which obscures or alters certain packet-level and flow-level attributes, potentially impacting all or a subset of the initially extractable features from packet headers, payloads, the underlying protocol, traffic flow metrics, etc. [17]–[19].

To address these challenges, we propose a novel framework that efficiently integrates advanced feature extraction and deep learning (DL) techniques to enable robust identification of DDoS attack vectors (such as ICMP floods, TCP SYN floods, UDP floods, MQTT publish floods, HTTP floods, and CoAP floods) and quantification of attack intensity (low, medium, high) in near real-time. Our proposed method is designed to operate independent of packet-level or flow-level features that may be obscured (e.g., application-layer information), significantly modified (e.g., packet size), or inaccessible (e.g., payload content) due to the enforcement of security standards at various OSI layers. Instead, it leverages time-frequency features derived directly from packet inter-arrival times (PIATs), which exhibit less susceptibility to network and configuration changes, ensuring high performance even when applied across different network and communication settings, including those configured with TLS, DTLS, and VPN.

The framework proposed in this study consists of three core modules: 1) traffic capturing via overlapping sliding time windows, 2) discrete wavelet transform (DWT)-based time-frequency feature extraction, and 3) a spatio-temporal hybrid DL model integrating convolutional neural networks (CNN) and attention-enhanced bidirectional long short-term memory (BiLSTM-ATT). Initially, the overlapping sliding window technique is applied to capture PIATs, forming the raw data sequences. Subsequently, these data sequences undergo DWT decomposition, yielding sets of wavelet coefficients corresponding to different resolution levels. Each set captures localized time-frequency information, enabling multi-scale analysis of temporal patterns. The resulting coefficients are processed by a CNN to extract spatial representations across decomposition levels. These representations are then fed into a BiLSTM network augmented with an attention mechanism, which selectively emphasizes informative temporal dependencies in both directions, thereby improving the model's predictive accuracy.

Our proposed methodology is conceptualized as a generic solution for DDoS attack vector identification (AVI), applicable across diverse network environments and datasets. To establish this, we first employ the IEEE P2668-compliant multi-layer DDoS threat modeling framework and its associated IoT-specific dataset, IEEE P2668-MLIDD (publicly available in: [10], [20]), as the primary basis for framework development. For independent cross-dataset validation, we further utilize the widely recognized Bot-IoT dataset [21]–[25] to evaluate performance beyond a single data source.

The structural characteristics of the adopted IoT network, the datasets used, and the potential integration point of the proposed AVI framework within the network topology are described in the following sections. In our analysis, both the IEEE P2668-MLIDD and Bot-IoT datasets are used as baseline traffic to represent typical protocol behavior (dynamics) and associated DDoS attack vectors. These datasets are natively devoid of any cryptographic processing overhead, thereby providing a clear and unobstructed view of the underlying traffic patterns and enabling the analysis of timing information (i.e., PIATs over time) in their most basic (unaltered) form.

To further extend these datasets and enable a more comprehensive evaluation, we employ Monte Carlo-based simulations to statistically model the effects of encryption overhead on timing characteristics. This allows us to analyze the resulting changes in the temporal dynamics of DDoS attacks and to quantitatively assess the framework's ability to discriminate attack vectors under encryption-induced increases in timing uncertainty.

The contributions of this paper are summarized as follows:

- *General Contribution:* This study presents a novel methodology and multi-class classification framework for attack-vector identification and quantification in heterogeneous and encrypted IoT networks, using only packet-timing characteristics. The proposed pipeline runs in near real-time and maintains strong performance across diverse network configurations, including deployments that use TLS, DTLS, and VPN protocols.
- *Cryptographic Processing Impact Modeling:* The impact of cryptographic processing overhead on packet timing attributes (i.e., PIATs) is statistically modeled using Monte Carlo-based simulations, resulting in stochastic variations that introduce noise-like distortions into the timing features, which degrade the clarity of class-distinct patterns. The extent of this degradation is validated through both visual inspection, using dimensionality reduction techniques such as t-distributed Stochastic Neighbor Embedding (t-SNE), and quantitative evaluation, including clustering quality metrics that measure intra-class compactness and inter-class separability.
- *Feature Extraction:* An efficient feature extraction strategy is proposed, where an overlapping time window-based traffic capturing mechanism and DWT are integrated to derive discriminative features directly from PIATs. This method is meticulously crafted to facilitate robust time-frequency feature extraction and to mitigate extended periods of non-responsiveness (deadlocks) dur-

ing the traffic capturing phase, thereby supporting accurate and near real-time performance in the downstream analysis.
- *Hybrid DL-Based Classification Architecture:* A hybrid DL-based classification architecture, comprising CNN, BiLSTM, and an attention mechanism (i.e., CNN-BiLSTM-ATT), is designed and meticulously tuned to capture and encode the spatio-temporal characteristics embedded in wavelet-transformed coefficients.
- *Validation and Adaptability Assessment*: The effectiveness and adaptability of the proposed framework are confirmed through comprehensive evaluations, including both quantitative metrics and qualitative assessments. Comparative results against shallow and deep models show consistent performance gains across all tested network environments.

The remainder of this paper is structured as follows: Section II reviews the relevant literature on AI-driven DDoS attack detection strategies. Section III describes the structural characteristics of the IoT network adopted in this study, specifies the potential integration point of the AVI systems within the network topology, characterizes the datasets used in this study, and models the impact of cryptographic processing overhead on their timing attributes. Section IV provides a detailed overview of the proposed AVI framework and its constituent modules. Section V outlines the module-wise hyperparameter tuning strategy, where each component is optimized individually while considering inter-module dependencies and their influence on end-to-end system performance. Section VI describes the evaluation setup, analyzes the model's performance, benchmarks the proposed approach against existing methods, and assesses its effectiveness across multiple datasets. Finally, Section VIII concludes the paper and discusses potential directions for future research.

II. Related Work

In the field of network security, the deployment of AI-driven anomaly detection techniques has markedly progressed in tackling the increasing complexities and volumetric expansions in contemporary network environments. Initially, simpler ML models, such as decision trees, support vector machines, k-nearest neighbors, etc., were implemented [26], [27]. These models effectively handled the simple network architectures and lower traffic volumes typical of earlier digital infrastructure phases.

As network infrastructures have evolved in scale and intricacy, the inadequacies of shallow ML models in addressing the sophisticated and multi-dimensional nature of modern cyber threats have become evident [28]. This has led to a paradigm shift toward adopting more advanced DL techniques. This new phase in anomaly detection incorporates various architectures, such as CNNs, recurrent neural networks (RNNs), and autoencoders (AEs), which are notable for their capacity to perform deep feature extraction—essential for deciphering the complex patterns embedded within extensive network traffic datasets [29], [30].

Illustratively, in [31], Najar *et al.* proposed a CNN-based DDoS attack detection framework for software defined networking (SDN) that addresses the challenges of class imbalance and non-informative feature selection observed in prior work. Through the use of balanced random sampling and information gain (IG)–based feature selection, their approach achieves binary classification accuracy exceeding 99.99% and multi-class accuracy of 98.64%. In [32], El-Sayed *et al.* introduced a novel DDoS attack detection framework that mitigates the adverse effects of feature redundancy and high dimensionality in network flow data. The method employs IG and random forest algorithms to identify and select a minimal set of highly discriminative features. Detection is then performed using a two-stage LSTM autoencoder architecture. Experimental results demonstrate that this feature-optimized DL approach achieves high detection accuracy while substantially reducing computational complexity. In [33], Abid *et al.* present a novel hybrid deep neural network architecture for DDoS attack detection and multi-class attack classification in SDN-enabled IoT environments. The proposed framework employs parallel convolutional and LSTM branches, each trained on distinct sets of flow-based features, with their outputs fused at the feature level. This joint spatial–temporal modeling approach enables effective learning of complex traffic dynamics, resulting in significant improvements in detection accuracy and false positive rate over traditional ML and DL-based algorithms. The study demonstrates the framework's capability to generalize across evolving DDoS attack patterns and heterogeneous network conditions.

Similarly, in [34], Nandanwar *et al.* developed a transfer learning-based hybrid CNN–BiLSTM intrusion detection model for IoT environments, capable of distinguishing benign traffic from nine different classes of Mirai and BASHLITE botnet attacks across multiple IoT devices. This approach addresses the narrow attack coverage of earlier works by enabling multiclass detection across various botnet variants, and achieves a testing accuracy of 99.52%, outperforming traditional DL methods. In [35], Ali *et al.* proposed a Vision Transformer-based framework for detecting DDoS attacks in IoT-edge environments, specifically focused on securing firmware over-the-air updates. The method transforms flow-level network traffic features into standardized representations using quantile normalization, reshapes them into compact RGB images, and uses a lightweight Vision Transformer model optimized with TensorFlow Lite for classification. The framework supports multi-class detection of 23 distinct DoS/DDoS attack types, achieving 99.50% accuracy while maintaining low memory usage and latency, enabling real-time deployment on edge devices with limited resources.

Despite the above discussion highlighting the effectiveness of approaches introduced in [31]–[35], the applicability of these models in encrypted network environments is constrained by their reliance on feature sets that are obscured, significantly altered, or rendered inaccessible through encryption. To address these constraints, research on encrypted traffic analysis has emerged and steadily expanded, aiming not only to mitigate the limitations of the above studies in anomaly detection but also to enable application identification, traffic management, and other tasks within encrypted domains. For instance, in [36], Shapira and Shavitt have introduced a

novel method for classifying encrypted Internet traffic, aimed at identifying specific applications within secure channels such as VPNs and Tor. By transforming basic flow data (specifically, packet sizes and arrival times) into an image-based format termed *FlowPic*, and subsequently analyzing these images through CNNs, they achieved high accuracy in classifying various types of network traffic and identifying distinct applications. However, as this method heavily relies on packet size, it is not effective for attack detection, as attackers can bypass the detection by employing techniques such as randomizing packet sizes and fragmentation. Furthermore, in [37], Kunda *et al.* introduced a classification system named *TSCRNN*, which can be used to enhance network resource management and quality of service within encrypted industrial IoT environments. This system leverages a combination of CNNs and BiLSTM networks to extract features from traffic flows, focusing on their spatiotemporal characteristics. Tested on the ISCXTor2016 dataset, the *TSCRNN* method achieved superior accuracy compared to other conventional classification techniques. However, similar to *FlowPic*, this work primarily focuses on distinguishing traffic types in Tor and non-Tor settings, thus leaving its applicability in specific anomaly and attack detection open to further exploration.

Beyond traffic profiling and application classification, several studies have targeted anomaly detection in encrypted settings by formulating the task as encrypted traffic multiclassification or encrypted anomaly inference [38]–[43]. Specifically, in [38], Zhu *et al.* address abnormal and encrypted IoT traffic classification by enforcing a fixed flow representation via packet truncation, padding, and byte-level normalization, then learning temporal dependencies using a BiLSTM module followed by a one-dimensional CNN. They further incorporate a cost penalty matrix through a dedicated cost penalty layer and an improved cross-entropy loss to alleviate class imbalance. In [39], Long *et al.* target encrypted traffic anomaly detection without decryption by combining a pretreatment module (flow purification, flow parsing, normalization, and data blocking) with parallel stacked AE-based feature extraction, followed by L1-regularization-based feature selection prior to classification, and report reduced feature redundancy and lower extraction overhead while maintaining detection effectiveness. In [41], Canavese *et al.* propose encryption-agnostic classifiers based on flow statistics derived from IP and TCP headers, enabling traffic-originator classification for both clear-text and encrypted connections without deep packet inspection. In [42], Bakhshi *et al.* evaluate multiple DL designs for encrypted Internet traffic anomaly detection and motivate a CNN+GRU hybrid, arguing that convolutional feature learning combined with recurrent time-domain modeling improves discrimination between normal and encrypted attack traffic while reducing manual feature engineering.

While these approaches achieve improved detection accuracy under encryption-induced constraints, they generally lack the ability to identify and quantify distinct attack vectors within a unified analytical framework. This shortcoming motivates the present study, which aims not only to reduce reliance on extensive feature sets but also to develop a model capable of maintaining high detection performance by effectively learning from minimal temporal features derived directly from packet timing behavior.

III. Overview of the IoT Network and Data Generation Setup

To support the design, implementation, and evaluation of the proposed AVI framework, this section provides a detailed overview of the adopted IoT network architecture, identifies the potential operational placement of the AVI system within the network topology, characterizes the raw datasets used in this study, and analyzes the impact of cryptographic overheads on their timing attributes.

A. IoT Network Architecture and Threat Modeling

The IoT network architecture employed in this study is adapted from the IEEE P2668-compliant multi-layer DDoS threat modeling framework originally introduced in [10], with modifications to incorporate the proposed AVI framework. The adapted architecture is illustrated in Fig. 1 and includes the following primary components: benign IoT devices, botnet, victim IoT server (VIS), application network, and AVI system.

- **Benign IoT Devices:** These include typical infrastructure components such as LoRa gateways and data-exchange routers, each assigned a unique IP address. These devices communicate with the VIS using TCP/IP-based multi-layer protocols and generate normal traffic to simulate legitimate IoT operations.
- **Botnet:** This consists of compromised nodes—including general-purpose computers and IoT routers—infected with malware such as Mirai [44]. These devices are remotely controlled and generate coordinated DDoS traffic using various transport- and application-layer protocols.
- **Victim IoT Server (VIS):** The VIS acts as a publicly accessible endpoint that handles both legitimate and malicious traffic. It is configured to support multiple protocol layers and has limited computational and bandwidth capacity, making it susceptible to resource exhaustion under DDoS conditions.
- **Application Network:** This component models the set of legitimate end-user services and applications—such as cloud-based interfaces—that interact with the VIS over the Internet. It generates benign service requests using protocols like MQTT, CoAP, and HTTP, reflecting typical real-world usage scenarios. The application network enables realistic simulation of client-server communication, facilitating accurate evaluation of system behavior under mixed traffic conditions.
- **AVI System:** The AVI system captures and monitors network traffic from multiple IP addresses, extracting relevant features to enable the identification and quantification of DDoS attack vectors.

B. Deployment of AVI within the Network Topology

To proactively mitigate service disruptions targeting the VIS, the proposed AVI framework is conceptually deployed upstream in the traffic path to enable early detection and response. It interfaces with two critical components:

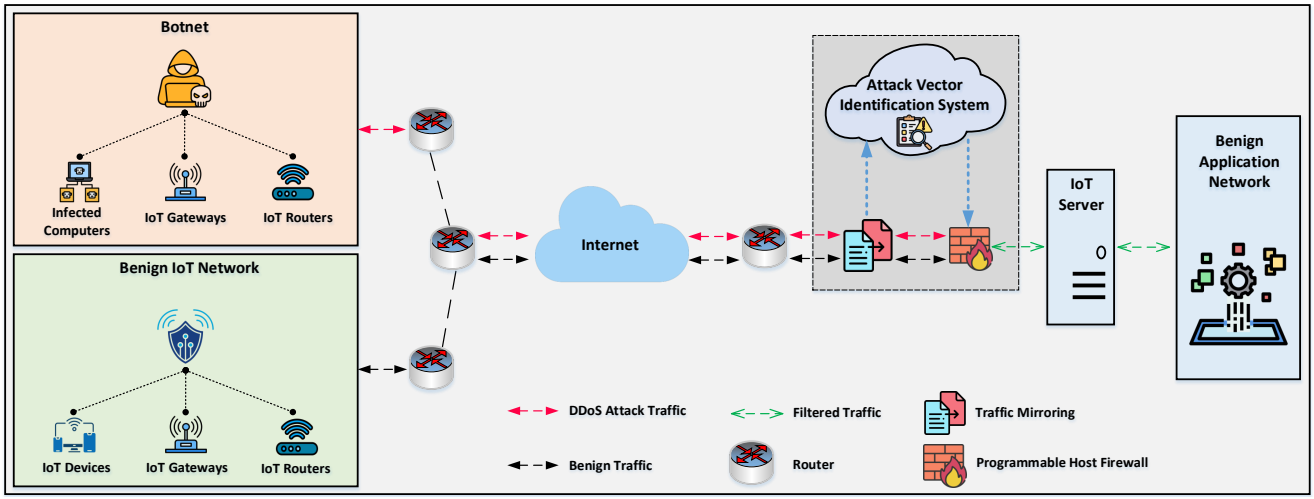

Fig. 1. IoT network topology comprising benign IoT devices, botnet, VIS, and the proposed AVI system.

1) **Traffic Mirroring System:** This component passively duplicates ingress traffic destined for the VIS, enabling real-time inspection without introducing additional latency or altering the delivery of the original packets. Traffic mirroring can be implemented using switch-based port mirroring (e.g., SPAN or RSPAN), physical Network TAPs, or software-based mechanisms such as Netfilter (e.g., iptables or nftables TEE targets) or extended Berkeley Packet Filters (eBPF) in kernel space. The duplicated traffic is forwarded to a dedicated interface on the AVI framework, where timing-based features are extracted for analysis.
2) **Programmable Host-Based Firewall:** This component enforces mitigation policies—such as source IP filtering, connection throttling, or traffic shaping—based on the detection outputs produced by the AVI framework. Integration is achieved through a well-defined control channel, which may involve local inter-process communication (e.g., UNIX domain sockets) or secure remote procedure calls (e.g., RESTful API over HTTPS). The firewall can be configured dynamically in response to detection signals, enabling automated defense actions. In virtualized or SDN environments, rule updates may also be propagated via a centralized SDN controller interfacing with the AVI module.

C. Dataset Description

1) IEEE P2668-MLIDD: The main dataset utilized in this study originates from the IEEE P2668-compliant multi-layer DDoS threat modeling framework, detailed in Section III-A, and is publicly accessible via the IEEE DataPort under the name IEEE P2668-MLIDD, provided in native `.pcap` format [20]. The dataset is pre-partitioned into training and testing subsets and comprises approximately 22 GB of network traffic, encompassing over 58 million data samples. These samples were generated using well-established tools, including hping3, Node-Red, and IoT-Flock. Both the training and testing partitions cover six widely used IoT network protocols along with their associated DDoS attack vectors: TCP SYN flood, UDP flood, ICMP flood, MQTT Publish flood, HTTP flood, and CoAP flood. Each malicious IP address corresponds to a specific attack vector and is executed at one of three predefined intensity levels—low, medium, or high—resulting in a total of 18 distinct attack classes (6 attack vectors × 3 intensities). Traffic generation sessions are time-aligned across all protocols and attack vectors, with each session lasting approximately 3600 seconds (1 hour) in the training set and 1800 seconds (1/2 hour) in the test set. This alignment facilitates fixed-size segmentation and yields a nearly balanced distribution of time series samples across different protocols and corresponding attack vectors—an essential property for ensuring unbiased model training and consistent performance evaluation of DL-based AVI systems. Further details regarding the dataset are available in [10], [20].

2) Bot-IoT: The second dataset used in this study is Bot-IoT [25]. It was collected in the UNSW Canberra Cyber Range Lab and released in multiple formats, including raw `.pcap` traces (reported at 69.3 GB, with more than 72 million records) and derived flow exports (Argus and CSV). Although Bot-IoT contains several attack categories (e.g., scanning and information theft), our study focuses specifically on the DDoS portion of the dataset. The DDoS traces include TCP-based flooding generated with hping3, UDP flooding generated with hping3, and HTTP flooding generated with the GoldenEye tool. Relative to IEEE P2668-MLIDD, Bot-IoT provides a smaller set of DDoS variants (TCP, UDP, HTTP) and does not define explicit low, medium, and high intensity labels. Instead, labeling is provided through an attack indicator together with attack category and subcategory fields. In this dataset, the DDoS attack subset exhibits class imbalance across attack vectors, which requires class-wise undersampling before it can be used to form balanced training and testing partitions, consistent with our IEEE P2668-MLIDD setup where each class is represented equally. In this work, Bot-IoT is used primarily for cross-dataset and cross-network evaluation, to quantify how well the proposed model transfers to a distinct data source collected under different network and testbed

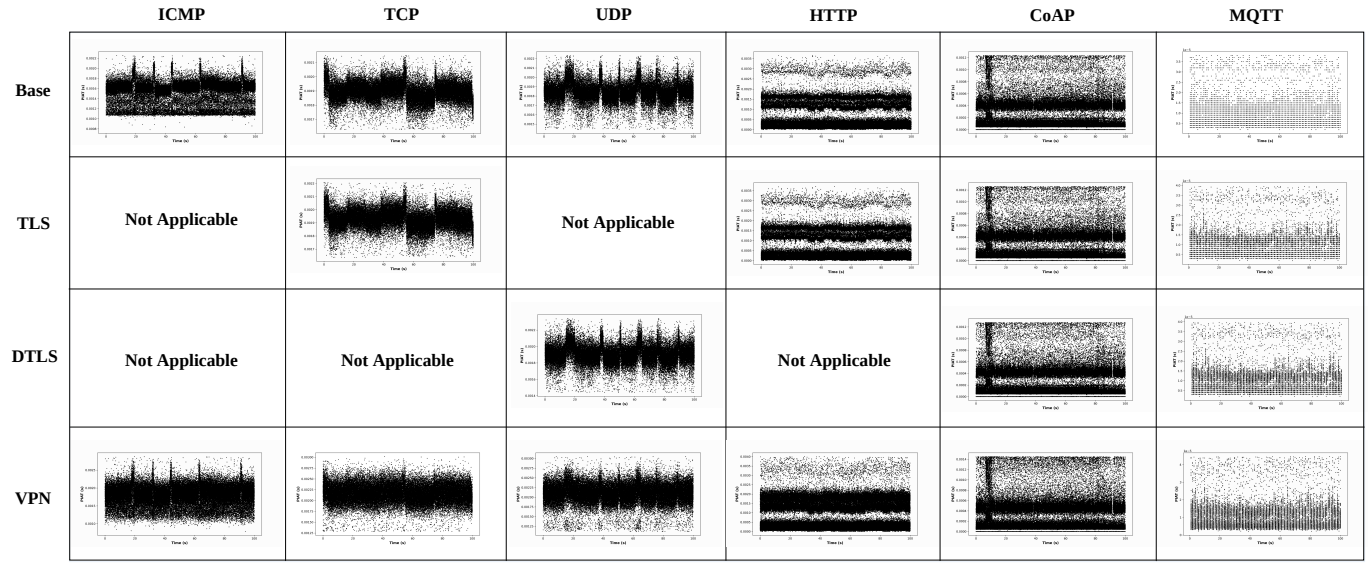

Fig. 2. Dynamics of PIATs in IoT networks under different security configurations. (**Note**: This figure distinctly illustrates the variations in PIAT patterns across different traffic types and network configurations over an extended observational interval (100 seconds). It is noteworthy, however, that within shorter observational intervals (e.g., 0.2 seconds), these patterns may not remain visually distinguishable with comparable clarity.)

conditions, and to assess robustness to site- and dataset-specific effects. Further details regarding the Bot-IoT dataset are available in [21]–[25].

D. Encryption-Induced Overheads and Their Impact on PIATs

The IoT network environments underlying the IEEE P2668-MLIDD and Bot-IoT dataset do not incorporate any security mechanisms (e.g., TLS, DTLS, VPN). In this baseline non-encrypted configuration, packet arrival latency and jitter[1] are predominantly dictated by intrinsic transmission and protocol-processing overheads associated with fundamental communication processes, including the handling of protocol-specific headers and control information, as well as timing variations introduced across the communication stack by queueing, retransmissions, buffering, host-side scheduling, etc. Under these conditions, the PIAT is defined as $T_i = t_{i+1} - t_i$, where t_i and t_{i+1} represent the arrival times of consecutive packets. However, the incorporation of security measures introduces additional latency overhead and also increase packet reordering, thereby potentially influencing flow-level and packet-level timing attributes. In particular, these effects give rise to cryptographic processing latency overheads, denoted by O^{enc}, which can be quantitatively expressed as follows:

$$O^{\text{enc}} = \tau^{\text{enc}}(t) \times S \tag{1}$$

[1]Notice that, here, *jitter* is not an intrinsic byproduct of the encryption process itself, but rather a timing variation that arises independently of encryption due to ordinary network behavior. Beyond the IEEE P2668-MLIDD and Bot-IoT datasets considered in this study, baseline jitter may also be sampled from other publicly available datasets, including CIC-DDoS2019 [45] and MedBIoT [46], and subsequently augmented by combining fixed per-packet delays to emulate realistic network noise. However, intentionally introduced timing randomization (e.g., beacon jitter) for evasion in post-exploitation and command-and-control (C2) communications is not considered in this work and is outside the scope of the present study.

Here $\tau^{\text{enc}}(t)$ is the time required to encrypt one byte of data at time t, which may vary with the instantaneous processing load and execution conditions of the host system, and S denotes the packet size. Including Eq. (1), the adjusted PIAT is calculated as:

$$\begin{aligned} T'_i = t'_{i+1} - t'_i &= (t_{i+1} + O^{\text{enc}}_{i+1}) - (t_i + O^{\text{enc}}_i) \\ &= T_i + (O^{\text{enc}}_{i+1} - O^{\text{enc}}_i) = T_i + \Delta O^{\text{enc}} \end{aligned} \tag{2}$$

where O^{enc}_i and O^{enc}_{i+1} is the cryptographic processing time for the i-th and $(i+1)$-th packets, respectively. Given that both τ^{enc} and S may vary across different networks, packet types, and security protocols, the values of O^{enc} and, consequently, ΔO^{enc} are inherently variable. To statistically model the impact of ΔO^{enc} variations on PIAT patterns, we conduct $100\,000$ Monte Carlo simulations, considering the potential additional overheads arising from the deployment of encryption protocols (e.g., AES-GCM, AES-CBC, ect.). Fig. 2 illustrates PIATs versus time (over a period of 100 seconds) for various DDoS attack vectors, both in the absence and presence of TLS, DTLS, and VPN (here, the IEEE P2668-MLIDD dataset is used as the baseline traffic flow to depict reference attack vectors under non-encrypted network conditions). Observation of Fig. 2 reveals that PIATs in their baseline profiles display distinct patterns, contributing to their identifiability. Upon the integration of TLS or DTLS, these patterns are largely retained. This preservation can be attributed to the uniform latency overhead imposed on successive packets within the traffic flow ($O^{\text{enc}}_i \approx O^{\text{enc}}_{i+1}$). According to Eq. (2) this results in an almost unchanged PIAT ($T'_i \approx T_i$). Conversely, employing VPN leads to decreased uniformity in overhead distribution among successive packets, which in turn elevates the entropy within PIAT patterns. This heightened entropy has the potential to adversely affect the distinctiveness between different protocol flows, such as ICMP, TCP, and UDP, thereby complicating their accurate identification.

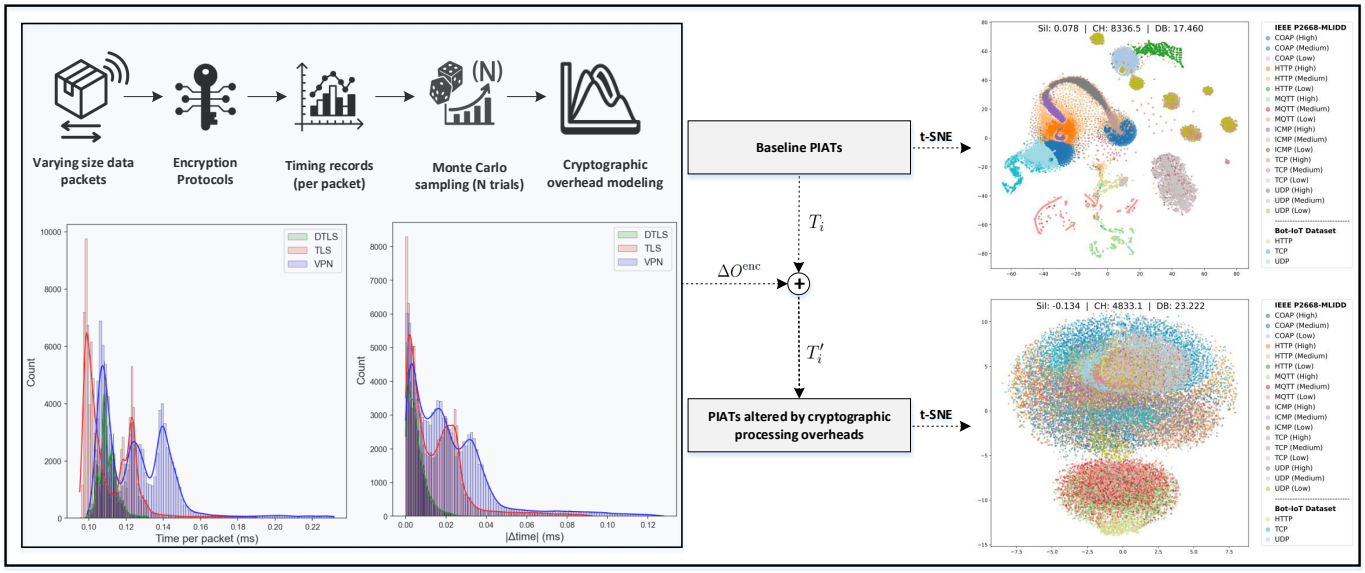

Fig. 3. t-SNE visualization of PIAT timing features with and without encryption overheads, showing that baseline traffic forms tighter, more separated class clusters, whereas added encryption-induced latency increases cluster overlap and dispersion in the embedding space.

To further investigate the effect of encryption-induced overhead on the separability of attack vectors when relying solely on timing-based features, we employ t-SNE on the PIAT feature set. t-SNE is a non-linear manifold learning approach that reduces high-dimensional data to a low-dimensional representation while preserving local neighborhood structure, enabling clear visualization of class-level clustering shaped by temporal characteristics. Fig. 3 shows the two-dimensional embeddings for both baseline traffic and traffic including encryption overheads. In the baseline setting, clusters corresponding to some traffic types are not fully compact but maintain some distinguishable boundaries. This suggests that PIAT-based features retain a degree of discriminative structure in the absence of encryption-related disturbances. By contrast, when latency overheads introduced by security protocols are incorporated, this structure deteriorates. The degradation is most pronounced in VPN traffic, where non-uniform distributions of overhead across successive packets increase entropy in packet timing attributes, thereby disrupting temporal regularities. This leads to greater overlap and dispersion in the t-SNE space, reflecting a marked reduction in class-specific separability.

To quantify this degradation in separability, we compute three clustering evaluation metrics on the t-SNE-transformed features:

- **Davies–Bouldin (DB) score:** Lower values indicate better-defined and more separated clusters.
- **Silhouette (Sil) score:** Ranges from -1 to 1; higher values reflect clearer separation and tighter cohesion.
- **Calinski–Harabasz (CH) score:** Higher values suggest well-separated and compact clusters.

Table I summarizes these metrics for the baseline network setting as well as for network conditions incorporating encryption-induced overheads introduced by TLS, DTLS, and VPN configurations. The baseline yields better overall scores, confirming moderate class separability. When encryption overhead is introduced—especially under VPN—there is a significant increase in the DB score and a noticeable decrease in the Sil and CH scores, all indicating reduced inter-class distinction and more dispersed clustering.

TABLE I
CLUSTERING METRICS ON T-SNE EMBEDDINGS OF PIAT FEATURES ACROSS BASELINE AND ENCRYPTED TRAFFIC

Score \ Traffic Type	VPN	TLS	DTLS	Base
DB	23.22	21.02	18.73	17.46
Sil	- 0.134	- 0.092	0.066	0.078
CH	4833.1	5509.3	7001.59	8336.5

IV. PROPOSED FRAMEWORK

In this section, we detail each phase of the proposed AVI framework, providing comprehensive descriptions of the modules and the fundamental techniques utilized within them. Specifically, we explore the intricacies of the traffic capturing, feature extraction, and learning phase, illustrating how these components integrate to form a unified framework capable of identifying and quantifying DDoS attack vectors.

A. Traffic Capturing Phase

Considering the exclusive reliance on PIATs as the sole input for our AVI model, it is imperative that this information is captured effectively for any arbitrary IP address (IP_X) in real-time. Given that the traffic capturing phase operates on a long-term, continuous basis, the subsequent feature extraction phase can only commence after the completion of the traffic capture. This sequential dependency results in a delay in initiating the feature extraction and learning processes, a phenomenon referred to as *long-term deadlock* of the traffic capturing phase [10]. To mitigate this issue, we employ a series of overlapping time windows, W_j, where $j \in \{1, 2, 3, \ldots\}$

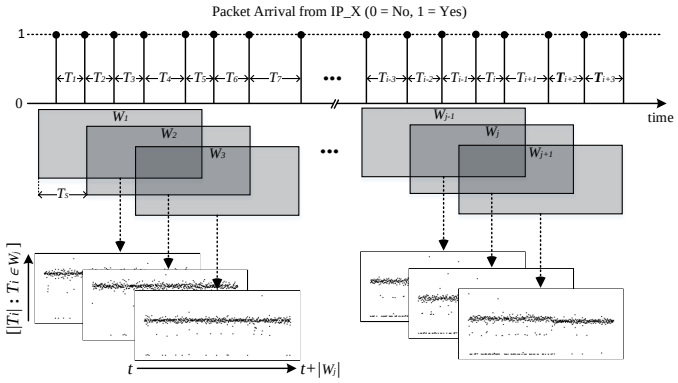

Fig. 4. Schematic of the sliding overlapping time window mechanism for real-time data sequence generation from PIATs.

represents the sequential counter of each time window, with a shifting time T_s that is less than the duration of W_j ($T_s < W_j$ to ensure the overlap). Upon the completion of each time window W_j, the traffic capturing module processes and transmits the PIATs, segmented by individual IP addresses, to the feature extraction phase. Specifically, the vector of PIAT measurements corresponding to each IP address during W_j is defined as:

$$\vec{\mathcal{S}}^{W_j} \leftarrow [|T_i| : T_i \in W_j], \quad i \in \{1, 2, 3, \ldots\}$$

where each $|T_i|$ quantifies the PIAT measurements in seconds. Fig. 4 provides a detailed illustration of this procedure. In this procedure, the size of the window W_j and the shifting time T_s are key parameters that determine both the frequency of data transmission and the temporal resolution of the analysis. Properly adjusting these parameters is essential for maintaining a balance between real-time responsiveness, computational efficiency, and accuracy, which is critical for the model to effectively adapt to varying network conditions and traffic volumes. In Section V-A, we conduct a study that assesses the model's performance across specific ranges of W_j and T_s and introduces the detected optimal settings for our application and dataset in the explored range.

B. Feature Extraction Phase

In the feature extraction phase, we adopt the DWT to perform time-frequency analysis on traffic flows. DWT is a widely validated signal processing technique that decomposes time-series data into multiple frequency subbands while preserving temporal locality [47]. Compared to the short-time Fourier transform (STFT), which relies on fixed-size windows and imposes a fundamental trade-off between time and frequency resolution, the DWT offers adaptive time-frequency resolution. This enables DWT to capture both short-term transients and long-range dependencies more effectively, making it well-suited for analyzing non-stationary and bursty traffic flows typically observed in IoT environments [48]. Additionally, unlike the continuous wavelet transform, which generates a highly redundant representation and incurs substantial computational overhead, DWT produces a compact, non-redundant output that is computationally efficient. These advantages, along with the benefits highlighted in [47] when compared to empirical mode decomposition and the Hilbert-Huang transform, make DWT a practical and scalable choice for real-time and large-scale applications, justifying its selection as the core time-frequency analysis method in our framework.

The DWT process is performed through the successive application of low-pass and high-pass filters using wavelets—zero-mean, localized functions derived from a predefined mother wavelet. Among various DWT implementations, we utilize the algorithm introduced by Mallat [49], known for its stability, efficient computation, and hierarchical structure. Mallat's algorithm performs multiresolution decomposition up to a specified level N, allowing for structured and interpretable feature extraction. The decomposition at level N is mathematically formulated as:

$$\vec{\mathcal{S}}^{W_j} = \vec{\mathcal{A}}_N^{W_j} + \sum_{n=1}^{N} \vec{\mathcal{D}}_n^{W_j}$$

where $\vec{\mathcal{A}}_N^{W_j}$ denotes the approximation coefficient vector at level N, providing the coarse-scale approximation of the signal, and $\vec{\mathcal{D}}_n^{W_i}$ represents the detail coefficient vector at each level n from 1 to N. These detail coefficients capture the high-frequency components, revealing finer details of the signal across progressively finer resolutions.

For each $\vec{\mathcal{S}}^{W_j}$ the highest level of wavelet decomposition, N, is determined by the characteristics of both the signal and the wavelet filter. The calculation of N can be expressed as:

$$N = \left\lfloor \log_2 \left(\frac{|\vec{\mathcal{S}}^{W_j}|}{|L_{\text{filter}}| - 1} \right) \right\rfloor \tag{3}$$

Here, $|\vec{\mathcal{S}}^{W_j}|$ denotes the length of the input vector (time-series signal), $|L_{\text{filter}}|$ represents the number of coefficients in the employed wavelet filter (i.e., the filter length), and $\lfloor \cdot \rfloor$ is the floor function used to round down to the nearest integer, determining the maximum number of decomposition levels applicable to the given signal. Furthermore, the length of each vector of detail coefficients at decomposition level n is determined by:

$$|\vec{\mathcal{D}}_n^{W_j}| = \left\lceil \frac{|\vec{\mathcal{S}}^{W_j}|}{2^n} \right\rceil \tag{4}$$

where $\lceil \cdot \rceil$ denotes the ceiling function, accounting for rounding up in cases where the division does not result in an integer.

According to Eq. (3) and Eq. (4), N and $|\vec{\mathcal{D}}_n^{W_j}|$ are subject to variation based on the length of $\vec{\mathcal{S}}^{W_j}$, which in turn is variable and influenced by the fluctuating number of packets received during each time window W_j. This variability in the lengths of the coefficient vectors and the achievable levels of decomposition leads to differing sizes of output during the feature extraction phase. Given that this output will subsequently be utilized in the learning phase, it is imperative to standardize these outputs to a uniform size. To address this requirement, a methodical approach is employed wherein the coefficients are mapped and subsequently padded to fit within a fixed-size, predefined feature matrix $M \in \mathbb{R}^{\mathcal{F} \times \mathcal{N}}$.

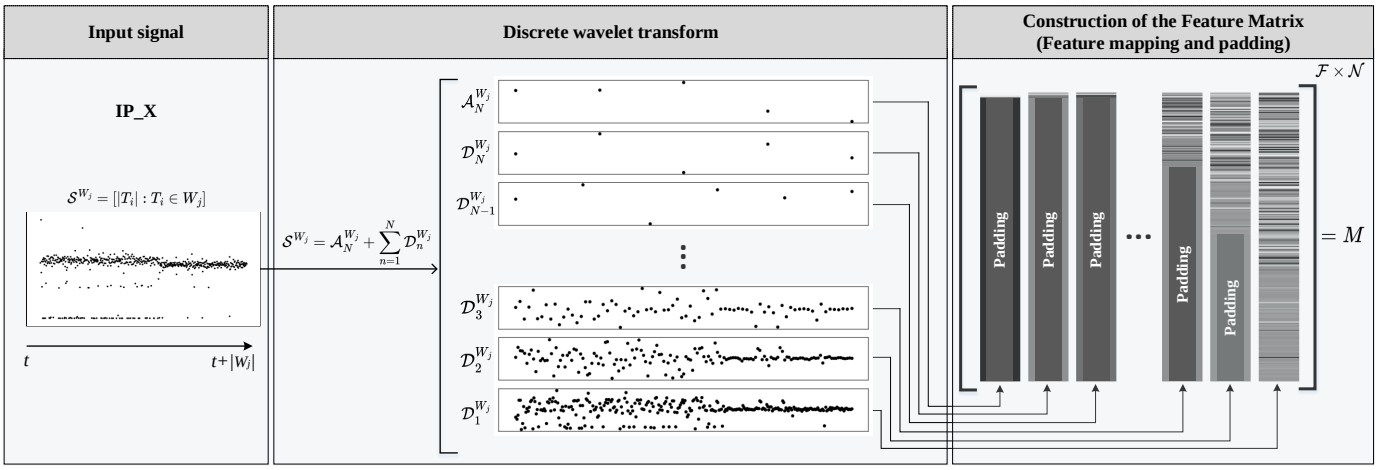

Fig. 5. Signal decomposition using DWT and subsequent feature matrix assembly (IP_X is an arbitrary IP address).

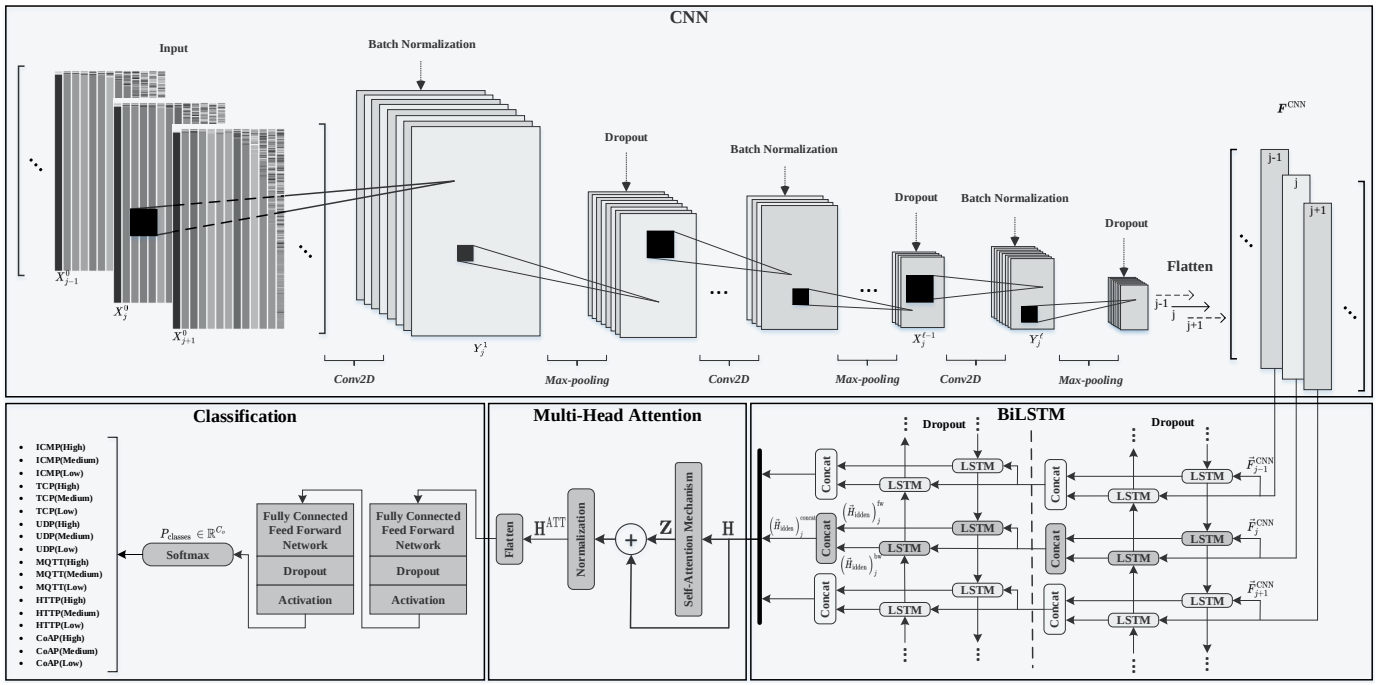

Fig. 6. Architectural integration of CNN and BiLSTM-ATT networks within the proposed hybrid DL model.

Here, $\mathcal{F}$ is designated as the maximum possible size of a coefficient vector, calculated based on the largest observed $\vec{\mathcal{S}}^{W_j}$, denoted as $\vec{\mathcal{S}}^{W_j}_{max}$, and $\mathcal{N}$ represents the maximum depth of decomposition that is achievable with $\vec{\mathcal{S}}^{W_j}_{max}$. Fig. 5 offers a visual illustration of the feature extraction procedure.

C. Deep Learning Phase

The proposed hybrid DL model integrates CNNs and BiLSTM-ATT networks to effectively extract and learn both spatial and temporal dependencies from wavelet transformed coefficients. In this subsection, we provide a detailed exposition of this architectural integration, illustrated in Fig. 6, specifically delineating how these disparate components are cohesively structured and interconnected to facilitate a thorough analysis of the input data.

1) Input Representation: Our model's input is structured as a four-dimensional tensor, designated by $\mathbf{X} \in \mathbb{R}^{\mathcal{K}\times\mathcal{F}\times\mathcal{N}\times 1}$, where the dimensions are specified as follows: $\mathcal{K}$ represents the number of discrete time steps in the sequential input; $\mathcal{F}$ indicates the number of rows in M; $\mathcal{N}$ denotes the number of columns in M; and the last dimension specifies a monochromatic channel representation, standard in scenarios where the input data consists of single-channel information, such as grayscale images or single-parameter measurements, which in our case corresponds to the assembled vectors of wavelet coefficients.

2) Time-Distributed Convolutional Feature Extraction: At the initial stage of the proposed architecture, time-distributed two-dimensional convolutional layers are applied independently across each discrete time step j. Each convolutional

block comprises a 2D convolution, followed sequentially by batch normalization, max pooling, and dropout regularization. These operations are designed to extract hierarchical spatial representations from the wavelet-transformed coefficients at each time slice. The convolutional transformation at layer ℓ is defined as:

$$Y_j^{\ell} = \sigma\left(\mathbf{W}_{\text{Conv}}^{\ell} * X_j^{\ell-1} + \mathbf{b}_{\text{Conv}}^{\ell}\right)$$

where $\mathbf{W}_{\text{Conv}}^{\ell}$ and $\mathbf{b}_{\text{Conv}}^{\ell}$ denote the convolutional kernel weights and biases, respectively, and $\sigma(\cdot)$ represents the non-linear activation function applied element-wise.

The output of the final convolutional layer in the CNN module is flattened at each time step to form a structured sequence of feature vectors:

$$\boldsymbol{F}^{\text{CNN}} = \left\{\vec{F}_j^{\text{CNN}}\right\}_{j=1}^{\mathcal{K}} \in \mathbb{R}^{\mathcal{K}\times d_{\text{CNN}}}$$

where $d_{\text{CNN}} = |\vec{F}_j^{\text{CNN}}|$ defines the dimensionality of the flattened feature vector for each time step j, and $\mathcal{K}$ indicates the total number of temporal segments. This sequence $\boldsymbol{F}^{\text{CNN}}$ serves as the input to the subsequent BiLSTM module.

3) Temporal Feature Learning with BiLSTM: At the second stage of the proposed architecture, the model transitions to temporal feature learning through interconnected BiLSTM layers. The operational dynamics of each LSTM layer are governed by the following equations:

$$(F_{\text{orget}})_j = \sigma\left(\mathbf{W}_{F_{\text{orget}}}\vec{F}_j^{\text{CNN}} + \mathbf{U}_{F_{\text{orget}}}(H_{\text{idden}})_{j-1} + \mathbf{b}_{F_{\text{orget}}}\right) \quad (5)$$

$$(I_{\text{nput}})_j = \sigma\left(\mathbf{W}_{I_{\text{nput}}}\vec{F}_j^{\text{CNN}} + \mathbf{U}_{I_{\text{nput}}}(H_{\text{idden}})_{j-1} + \mathbf{b}_{I_{\text{nput}}}\right) \quad (6)$$

$$(O_{\text{utput}})_j = \sigma\left(\mathbf{W}_{O_{\text{utput}}}\vec{F}_j^{\text{CNN}} + \mathbf{U}_{O_{\text{utput}}}(H_{\text{idden}})_{j-1} + \mathbf{b}_{O_{\text{utput}}}\right) \quad (7)$$

$$\begin{aligned}(C_{\text{ell}})_j = (C_{\text{ell}})_{j-1} \otimes (F_{\text{orget}})_j + (I_{\text{nput}})_j \\ \otimes \left(\tanh\left(\mathbf{W}_{C_{\text{ell}}}\vec{F}_j^{\text{CNN}} + \mathbf{U}_{C_{\text{ell}}}(H_{\text{idden}})_{j-1} + \mathbf{b}_{C_{\text{ell}}}\right)\right)\end{aligned} \quad (8)$$

$$(H_{\text{idden}})_j = O_{\text{utput}} \otimes \tanh\left((C_{\text{ell}})_{j-1}\right) \quad (9)$$

Eqs. (5)-(9) articulate the operation of five key components within the LSTM architecture: the forget gate $(F_{\text{orget}})_j$, input gate $(I_{\text{nput}})_j$, output gate $(O_{\text{utput}})_j$, transferable cell state $(C_{\text{ell}})_j$, and hidden state $(H_{\text{idden}})_j$. These components are crucial for modulating the flow of information through the model and are implemented using a series of weight matrices: $\{(\mathbf{W}_{F_{\text{orget}}}, \mathbf{U}_{F_{\text{orget}}}), (\mathbf{W}_{I_{\text{nput}}}, \mathbf{U}_{I_{\text{nput}}}), (\mathbf{W}_{O_{\text{utput}}}, \mathbf{U}_{O_{\text{utput}}}), (\mathbf{W}_{C_{\text{ell}}}, \mathbf{U}_{C_{\text{ell}}})\}$ and bias vectors: $\{\mathbf{b}_{F_{\text{orget}}}, \mathbf{b}_{I_{\text{nput}}}, \mathbf{b}_{O_{\text{utput}}}, \mathbf{b}_{C_{\text{ell}}}\}$. These parameters facilitate linear transformations and control the gating mechanisms, enabling the model to dynamically learn from and respond to the input data throughout the training process.

To capture bidirectional temporal dependencies, the BiLSTM architecture processes the input sequence $\boldsymbol{F}^{\text{CNN}}$ along two parallel temporal paths. Specifically, at each time step j, the forward LSTM computes a hidden state $(\vec{H}_{\text{idden}})_j^{\text{fw}}$ by traversing the sequence in the forward direction, while the backward LSTM computes $(\vec{H}_{\text{idden}})_j^{\text{bw}}$ by traversing it in reverse. These two hidden states are concatenated to form a comprehensive representation at each time step:

$$(\vec{H}_{\text{idden}})_j^{\text{concat}} = \text{Concat}\left[(\vec{H}_{\text{idden}})_j^{\text{fw}}, (\vec{H}_{\text{idden}})_j^{\text{bw}}\right]$$

The concatenated outputs across all time steps define a structured sequence of hidden states:

$$\mathbf{H} = \left\{(\vec{H}_{\text{idden}})_j^{\text{concat}}\right\}_{j=1}^{\mathcal{K}} \in \mathbb{R}^{\mathcal{K}\times d_{\text{BiLSTM}}}$$

where $\mathcal{K}$ denotes the number of discrete time steps in the sequence, and $d_{\text{BiLSTM}} = 2\mathcal{H}$, with $\mathcal{H}$ representing the number of hidden units in each LSTM direction. The sequence $\mathbf{H}$ is then used as input to the subsequent attention layer.

4) Temporal Context Refinement via Multi-Head Self-Attention: In the third stage of the architecture, the sequence of concatenated hidden states is enhanced using a multi-head self-attention mechanism to capture global contextual dependencies across time steps. To perform this, each attention head $m \in \{1, \ldots, \mathcal{M}\}$, independently projects the input sequence $\mathbf{H}$ into three learned representations: queries, keys, and values. These are computed as:

$$\mathbf{Q}^{(m)} = \mathbf{H}\mathbf{W}_{(m)}^{Q}, \quad \mathbf{K}^{(m)} = \mathbf{H}\mathbf{W}_{(m)}^{K}, \quad \mathbf{V}^{(m)} = \mathbf{H}\mathbf{W}_{(m)}^{V}$$

where $\mathbf{W}_{(m)}^{Q}, \mathbf{W}_{(m)}^{K}, \mathbf{W}_{(m)}^{V} \in \mathbb{R}^{d_{\text{BiLSTM}}\times d_k}$ are trainable weight matrices, and d_k denotes the dimensionality of the subspace used by each head.

Each attention head then calculates its output using the scaled dot-product attention:

$$\mathbf{A}^{(m)} = \text{Softmax}\left(\frac{\mathbf{Q}^{(m)}(\mathbf{K}^{(m)})^{\top}}{\sqrt{d_k}}\right) \cdot \mathbf{V}^{(m)}$$

which produces $\mathbf{A}^{(m)} \in \mathbb{R}^{\mathcal{K}\times d_k}$. The outputs of all $\mathcal{M}$ heads are then concatenated and passed through a final linear projection:

$$\mathbf{Z} = \text{Concat}\left[\mathbf{A}^{(1)}, \ldots, \mathbf{A}^{(\mathcal{M})}\right]\mathbf{W}^{O}, \quad \mathbf{W}^{O} \in \mathbb{R}^{(\mathcal{M}\times d_k)\times d_{\text{BiLSTM}}}$$

bringing the combined output back to the original dimensionality d_{BiLSTM}. To retain the original sequence information and ensure stable optimization, a residual connection is applied followed by layer normalization:

$$\mathbf{H}^{\text{ATT}} = \text{LayerNorm}(\mathbf{H} + \mathbf{Z})$$

This yields the final attention-refined sequence $\mathbf{H}^{\text{ATT}} \in \mathbb{R}^{\mathcal{K}\times d_{\text{BiLSTM}}}$, which is used in the subsequent classification stage.

5) Classification with Fully Connected Feed Forward Network: In the final stage of the model, the output of the attention layer $\mathbf{H}^{\text{ATT}}$ is flattened into a single feature vector. This vector is then fed to a series of fully connected (Dense) layers. These layers are designed to refine and enhance the feature set further, preparing the data for final classification decisions. To mitigate the risk of overfitting and enhance the model's generalization across unseen data, dropout regularization is incorporated alongside each Dense layer. The processing chain reaches its culmination in a softmax output layer, which translates the activations from the last Dense layer into a probability distribution over potential output classes, represented as $P_{\text{classes}} \in \mathbb{R}^{C_o}$, where, C_o denotes the number of classes.

TABLE II
SEARCH SPACE AND SELECTED VALUES FOR W_j AND T_s

Parameter	Search Space	Selected Value
W_j	{0.5, 1, 5, 10}	1
T_s	{0.1, 0.2, 0.3, 0.5}	0.2

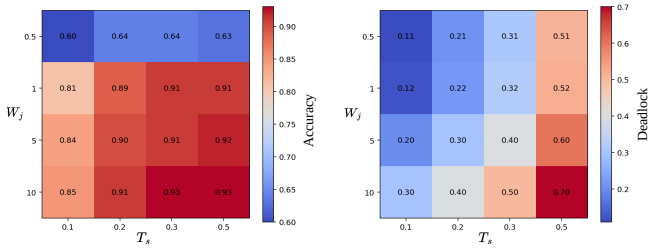

Fig. 7. Trends in model accuracy (Scale: 0-1) and deadlock duration (seconds) with varying W_j and T_s.

TABLE III
SEARCH SPACE AND SELECTED VALUES FOR DWT KEY PARAMETERS

Parameter	Search Space	Selected Value
Wavelet Family	{db, sym, coif}	db
Wavelet Order	{1, 2, 3, 4, 5}	1

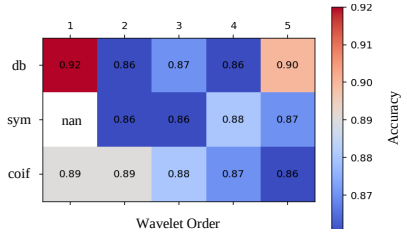

Fig. 8. Trends in model accuracy (scale: 0-1) with varying wavelet families and orders.

V. MODULE-WISE HYPERPARAMETER TUNING AND DETAILS OF THE SELECTED CONFIGURATION

As discussed in Section IV, the proposed framework comprises three interconnected modules. Optimal performance of the overall framework critically depends on the effective hyperparameter tuning of each module. Nevertheless, simultaneous hyperparameter optimization across all modules poses substantial technical challenges. These challenges include an exponential expansion in the dimensionality and complexity of the hyperparameter search space, difficulty in isolating and attributing variations in performance to individual module parameters, considerable computational overhead, and dimensional inconsistencies arising from data transformations between consecutive modules. To mitigate these challenges, we adopt a sequential, module-wise hyperparameter optimization strategy. The detailed methodology for each stage of this sequential optimization process is presented in the subsequent subsections.

A. Tuning the Traffic Capturing Module

In the architecture of the proposed framework, the initial parameters that require calibration pertain to the traffic capturing module, specifically W_j and T_s. As discussed in Section IV-A, these parameters play a crucial role in maintaining the balance between real-time responsiveness and model accuracy. To effectively calibrate these parameters, their impact on the system's workflow performance is thoroughly assessed through empirical evaluation. For this assessment, the configurations of subsequent modules (feature extraction module and DL module) are set to their basic forms. Specifically, feature extraction is performed using Haar wavelets (the simplest and most commonly used wavelet), and the DL architecture is simplified to include a single convolutional layer followed by a unidirectional LSTM layer. Rather than maximizing classification accuracy, our primary objective here is to elucidate the effects of varying W_j and T_s on overall system performance within a controlled and simplified experimental setup. This preliminary assessment ensures that these parameters are appropriately configured before proceeding with the selection of wavelet configuration and extensive hyperparameter tuning of the full-scale DL model.

The range of values explored for W_j and T_s is summarized in Table II, with the final selection determined based on the trade-off between observed deadlock duration and accuracy outcomes. As illustrated in Fig. 7, both model accuracy and deadlock duration exhibit an increasing trend as W_j and T_s increase. However, beyond $W_j = 1$ second and $T_s = 0.2$ seconds, the rate of accuracy improvement diminishes, while the increase in deadlock duration becomes more substantial. Based on this trade-off analysis, $W_j = 1$ and $T_s = 0.2$ are selected as the optimal parameter values within the explored range.

B. Tuning the Feature Extraction Module

IoT networks commonly face DDoS attacks that differ widely in intensity, leading to packet floods that can escalate from a few packets per second in less severe cases to several thousand packets per second during intense events. Given the fluctuating nature of network traffic, particularly under different attack scenarios, effective feature extraction must be performed to ensure that meaningful patterns are captured from PIAT without unnecessary loss of critical information. In the DWT-based feature extraction, the selection of appropriate wavelet families and filter sizes is crucial, as different configurations yield distinct sets of coefficients. Given the extensive array of wavelet types available, conducting a comprehensive evaluation of every possibility remains impractical. Therefore, this study narrows its focus to an assessment of select wavelet families—namely Daubechies (db), Symlets (sym), and Coiflets (coif)—with wavelet orders ranging from 1 to 5. Fig. 8 delineates the impact of these particular wavelet configurations on the accuracy of a basic CNN-LSTM model. Concurrently, Table III encapsulates the parameter space explored and the selected value based on the achieved model accuracy. Mirroring the calibration strategy employed for W_j and T_s, the primary aim here is not to maximize

TABLE IV
HYPERPARAMETER SEARCH SPACE AND SELECTED VALUES FOR THE CNN-BILSTM-ATT MODEL

Hyperparameter	Search Space	Selected Value
Number of CNN Layers	{1, 2, 3}	3
Number of LSTM Layers	{1, 2, 3}	2
Number of Dense Layers	{1, 2, 3}	2
CNN Filters (1)	{32, 64, 128, 256}	128
CNN Filters (2)	{32, 64, 128, 256}	64
CNN Filters (3)	{32, 64, 128, 256}	128
CNN Kernel Size (1)	$\{3\times3, 5\times5, 7\times7\}$	5×5
CNN Kernel Size (2)	$\{3\times3, 5\times5, 7\times7\}$	3×3
CNN Kernel Size (3)	$\{3\times3, 5\times5, 7\times7\}$	3×3
CNN Activation	{ReLU, Tanh, Sigmoid}	ReLU
CNN Regularization	{0.0001, 0.001, 0.01}	0.001
CNN Spatial Dropout Rate	{0.1, 0.2, 0.5}	0.1
LSTM Units	{50, 100, 150, 200, 250}	200
LSTM Dropout	{0.1, 0.2, 0.5}	0.1
LSTM Recurrent Dropout	{0.0, 0.1, 0.2, 0.5}	0.1
Attention Heads	{2, 4, 8}	4
Attention Key Dim	{32, 64, 128}	64
Attention Dropout	{0.1, 0.2, 0.5}	0.1
Dense Layer Units (1)	{64, 128, 256}	256
Dense Layer Units (2)	{64, 128, 256}	128
Dense Activation	{ReLU, Tanh, Sigmoid}	ReLU
Dense Dropout Rate	{0.1, 0.2, 0.5}	0.1
Optimizer	{Adam, SGD, RMSprop}	Adam
Learning Rate	log-uniform $[10^{-6}, 10^{-2}]$	1.01×10^{-5}

classification accuracy. Instead, the objective is to thoroughly examine the impact of various wavelet types on overall system performance within a controlled and streamlined experimental setup. This ensures that wavelet parameters are configured in a more appropriate manner before the transition to advanced hyperparameter tuning, which is essential for optimizing the full-scale DL model.

According to Table III, Daubechies 1 ($|L_{\text{filter}}| = 2$) is identified as the most advantageous selection. Building on this selection, as explicated in Section IV-B, the dimensions of the feature matrix can be calculated as: $\mathcal{F} = \left\lceil \frac{|\vec{\mathcal{S}}^{W_j}_{max}|}{2} \right\rceil$ and $\mathcal{N} = \left\lfloor \log_2 \left(|\vec{\mathcal{S}}^{W_j}_{max}| \right) \right\rfloor$. This implies that at each time step $j \in \{1, 2, 3, \ldots\}$, the assembled feature matrix is embedded within a multidimensional space, represented as $M \in \mathbb{R}^{\mathcal{F}\times\mathcal{N}}$, whose dimensionality can be reduced using statistical transformations for visualization purposes or to facilitate faster training and inference.

C. Tuning and Training the DL Module

The final module of the proposed framework requiring hyperparameter tuning is the hybrid DL-based multi-class classifier. The objective is to determine the optimal configuration that maximizes classification accuracy, given previously selected parameters ($W_j = 1$ and $T_s = 0.2$ for the traffic capturing phase and the db1 wavelet for the time-frequency feature extraction phase). To this end, the training partition of the IEEE P2668-MLIDD dataset is utilized, where cryptographic processing overheads are applied to the traffic traces, as described in Section III-D. Subsequently, the traffic

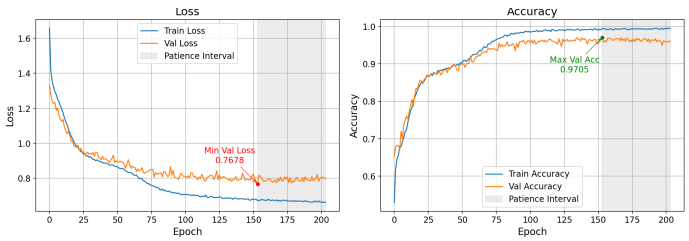

Fig. 9. Performance curves of the tuned model.

capturing module records PIATs over time, segmenting them into fixed one-second intervals to generate raw data sequences. Approximately 3600 samples per attack class (one sample per second from sessions lasting roughly one hour) are obtained from these segmented records. Given 18 distinct attack classes (six DDoS attack vectors, each conducted at three intensity levels), this process yields approximately 64800 samples in total.

To reduce computational complexity while preserving class balance, a stratified sampling approach is used to construct the tuning set: 1000 samples per class are selected for training and 200 per class for validation, yielding 18000 training samples and 3600 validation samples.

Hyperparameter tuning is conducted using Bayesian optimization through the Optuna framework, employing the Tree-Structured Parzen Estimator (TPE) algorithm to efficiently explore the hyperparameter search space. In addition to general architectural parameters, such as learning rate, CNN kernel size, LSTM units, etc., dropout rates in intermediate and output layers are tuned to reduce overfitting, along with L2 regularization coefficients in convolutional layers. Early stopping with a patience of 50 epochs is also applied as an additional measure against overfitting. Each optimization trial runs for a maximum of 500 epochs with a fixed batch size of 32 to maintain stable training dynamics and manageable memory consumption. Ultimately, the configuration that achieves the highest validation accuracy is selected as optimal.

The final model is trained using the Adam optimizer and categorical cross-entropy loss. A complete overview of the search space, selected hyperparameters, and final values is provided in Table IV. The training and validation curves for the chosen configuration are shown in Fig. 9, demonstrating stable convergence and consistent generalization performance.

The final tuned model comprises approximately 3.62 million trainable parameters, with an estimated computational cost of 3.1×10^9 floating-point operations (FLOPs) per forward pass at a batch size of 32.

VI. MODEL EVALUATION AND COMPARATIVE ANALYSIS

A. Evaluation Sets and Performance Metrics

To provide a rigorous evaluation of the AVI system under diverse DDoS attack scenarios, protocol-specific cryptographic overheads are incorporated into the timing attributes of the IEEE P2668-MLIDD and Bot-IoT dataset. The procedure

for integrating these cryptographic overheads is detailed in Section III-D. On this basis, the resulting test subsets are defined as follows:

$$\mathbf{TEST}_1 \in \{\mathfrak{T}_{\text{Base}_1}, \mathfrak{T}_{\text{TLS}_1}, \mathfrak{T}_{\text{DTLS}_1}, \mathfrak{T}_{\text{VPN}_1}\} \tag{10}$$

$$\mathbf{TEST}_2 \in \{\mathfrak{T}_{\text{Base}_2}, \mathfrak{T}_{\text{TLS}_2}, \mathfrak{T}_{\text{DTLS}_2}, \mathfrak{T}_{\text{VPN}_2}\} \tag{11}$$

where $\mathbf{TEST}_1$ is derived from IEEE P2668-MLIDD and $\mathbf{TEST}_2$ is derived from the DDoS subset of Bot-IoT. To ensure class-balanced evaluation, each test subset is formed via undersampling to approximately 1800 instances per class. Consequently, each IEEE P2668-MLIDD test configuration contains $18 \times 1800 = 32400$ samples, where 18 is the number of available DDoS attack classes in IEEE P2668-MLIDD. Likewise, each Bot-IoT test configuration contains $3 \times 1800 = 5400$ samples, where 3 is the number of classes in the DDoS portion of Bot-IoT.

To assess the classification performance across the aforementioned test subsets, we employ four standard evaluation metrics: overall accuracy, macro-averaged precision, macro-averaged recall, and macro-averaged F1-score. Macro-averaging is selected as the aggregation strategy due to the balanced class distribution in each test subset, ensuring that all classes contribute equally to the final evaluation metrics. The corresponding mathematical definitions are as follows:

$$\textbf{Overall Accuracy } (ACC) = \frac{1}{I}\sum_{i=1}^{I} \mathbb{1}(y_i = \hat{y}_i) \tag{12}$$

$$\textbf{Macro Precision } (P_{\textbf{macro}}) = \frac{1}{C_o}\sum_{c=1}^{C_o} \frac{TP_c}{TP_c + FP_c} \tag{13}$$

$$\textbf{Macro Recall } (R_{\textbf{macro}})\textbf{:} = \frac{1}{C_o}\sum_{c=1}^{C_o} \frac{TP_c}{TP_c + FN_c} \tag{14}$$

$$\textbf{Macro F1-score } (F1_{\textbf{macro}})\textbf{:} = \frac{1}{C_o}\sum_{c=1}^{C_o} \frac{2 \cdot P_{\text{macro}} \cdot R_{\text{macro}}}{P_{\text{macro}} + R_{\text{macro}}} \tag{15}$$

where in Eq. (12), I is defined as the total number of samples, y_i as the true label, $\hat{y}_i$ as the predicted label, and $\mathbb{1}(\cdot)$ as the indicator function, and in Eqs. (13)–(15) TP_c, FP_c, and FN_c are defined as the true positives, false positives, and false negatives for class c, with C_o denoting the total number of classes.

B. Model Evaluation on IEEE P2668-MLIDD

The proposed AVI model is first evaluated on the test partition of the IEEE P2668-MLIDD dataset defined in Eq. (10). The classification results are provided in Fig. 10, encompassing confusion matrices for detailed error analysis, aggregated performance metrics for overall assessment, and per-class accuracy distributions to evaluate class-specific effectiveness.

1) Overall Performance: A comparative evaluation across heterogeneous network configurations reveals that classification accuracy follows a descending trend in the order Base (97.98%), DTLS (97.89%), TLS (97.31%), and VPN (94.12%), indicating a progressive reduction in the discriminability of traffic classes with increasing protocol-induced timing variability and cryptographic processing complexity. Misclassification rates remain consistently low for $\mathfrak{T}_{\text{Base}_1}$, $\mathfrak{T}_{\text{TLS}_1}$, and $\mathfrak{T}_{\text{DTLS}_1}$, yet exhibit a noticeable increase in $\mathfrak{T}_{\text{VPN}_1}$. The deterioration observed under VPN conditions can be ascribed to entropy amplification in PIAT-based features, arising from diminished uniformity in latency–overhead distributions across consecutive packets. Such distortions obscure class-specific temporal signatures, disrupt flow-level statistical dependencies, and reduce inter-class separability within the feature space, thereby impairing the model's capacity to achieve accurate classification.

2) Class-wise Performance: A closer examination of the confusion matrices reveals persistent misclassification specifically among ICMP, TCP, and UDP flood attacks, independent of the network configuration. These classes exhibit the highest levels of mutual misidentification, particularly at lower intensities for ICMP and UDP, where reduced packet emission rates diminish the separability of their timing-based feature distributions. Likewise, MQTT floods exhibit recurring misclassification between high, medium, and low intensities. The underlying cause of these misclassification patterns is the strong similarity of PIAT distributions across these traffic types, as PIAT-based features capture intrinsic timing characteristics that are closely aligned for ICMP, TCP, and UDP floods (see Fig. 2) and undergo only minor shifts across different intensities of MQTT floods (see Fig. 11).

C. Cross-Dataset Evaluation

To assess the generalization capability of the proposed AVI model, a second evaluation phase is conducted on the Bot-IoT dataset. Fig. 12 illustrates PIATs over time for representative DDoS attack vectors in Bot-IoT. The DDoS subset of Bot-IoT contains three attack-vector classes (TCP, UDP, and HTTP). Compared with IEEE P2668-MLIDD, which contains 18 classes, this reduced label space yields lower multi-class decision complexity. Consistent with this observation, when the model is trained on the Bot-IoT training partitions and tested on $\mathfrak{T}_{\text{VPN}_2}$, it achieves an accuracy of approximately 97.44% (almost 3% higher than the 18-class classification task using IEEE P2668-MLIDD). In contrast, under cross-dataset validation, where the model is trained exclusively on IEEE P2668-MLIDD and evaluated on Bot-IoT, the accuracy drops substantially to near-chance levels. This degradation is primarily attributed to distribution shift between the datasets (see the differences between Fig. 12 and Fig. 2), including differences in network topology, background traffic composition, and workload characteristics.

From a deployment perspective, both extreme train-evaluation settings, one yielding very high accuracy (97.44%) and the other yielding near-chance accuracy, are unlikely. In

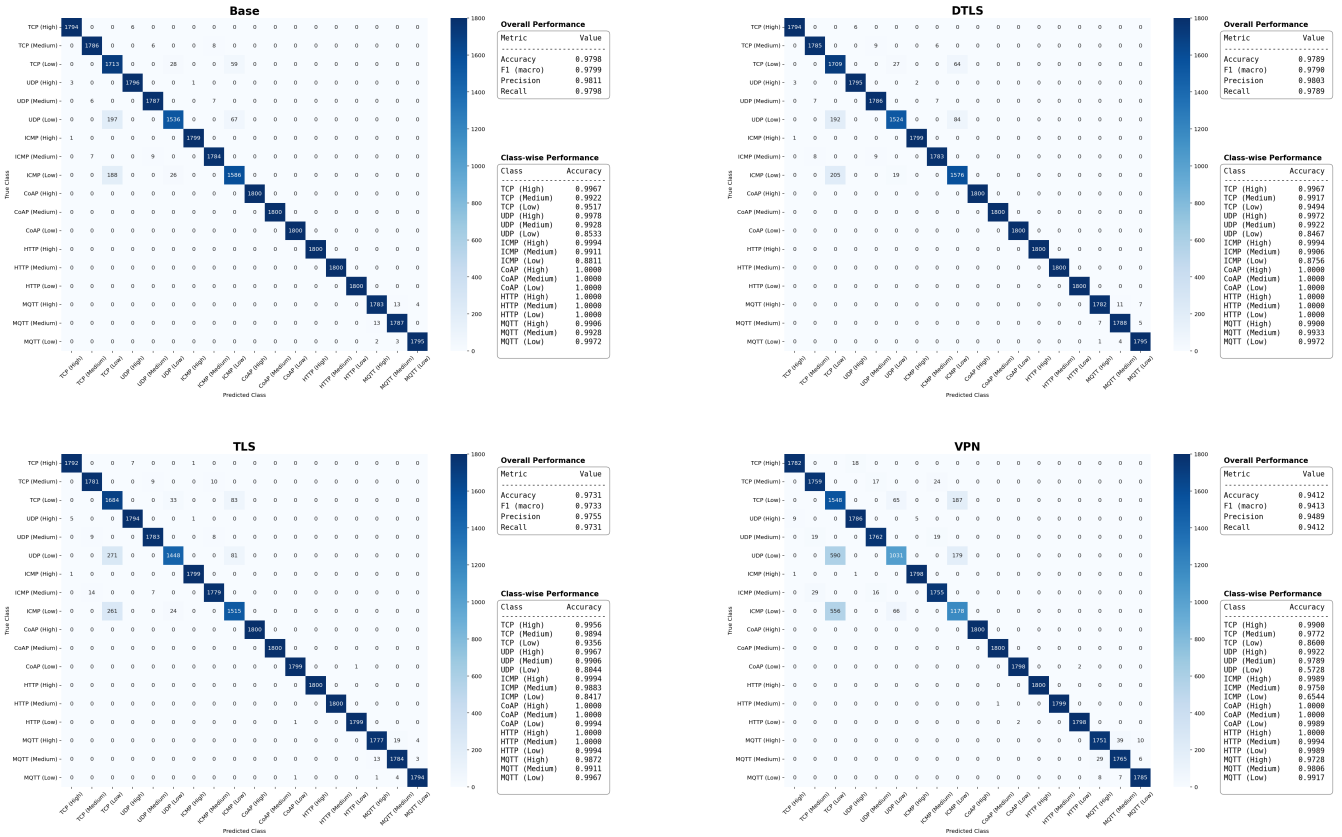

Fig. 10. Confusion matrices showing overall and class-wise performance metrics under different network configurations.

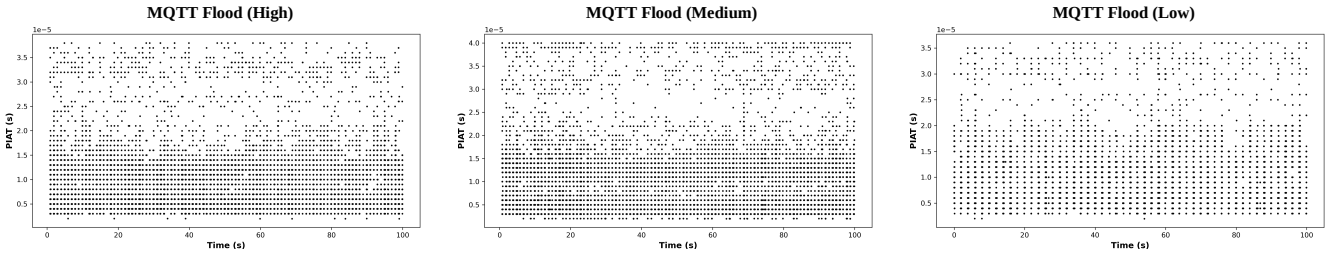

Fig. 11. PIAT patterns across MQTT flood intensities.

the first extreme (training and testing strictly within Bot-IoT), the operating network is implicitly assumed to remain stable, which is rarely the case in real environments. Network topologies evolve over time due to the addition or removal of subnets and changes in the device population (e.g., new IoT nodes, replacements, etc.). In parallel, background workloads shift with service updates and time-varying load patterns. These factors alter timing behavior and PIAT distributions, meaning that evaluation on a fixed closed-world split can overestimate performance under real deployment drift.

In the second extreme (training only on IEEE P2668-MLIDD and testing on a fully different dataset, such as Bot-IoT), the inference-time traffic is assumed to follow an almost completely unseen distribution. This is also uncommon in practice, because deployment typically includes at least a limited amount of site-specific data for calibration, adaptation, or periodic updates. Moreover, botnets and DDoS campaigns usually leverage already deployed, but compromised, hosts and services in the target environment (or closely connected environments), so the generated attack traffic is superimposed on existing application workloads. Therefore, realistic conditions tend to fall between these extremes, where one domain is under-represented relative to the other, rather than being entirely absent.

To quantify the impact of such imbalance, we construct mixed training sets by combining Bot-IoT and IEEE P2668-MLIDD at different ratios and evaluate the resulting performance. Table V presents the results of these experiments. Across all configurations in Table V, the total number of training samples is kept constant at 3600 per class, while only the proportion of Bot-IoT versus IEEE P2668-MLIDD samples is varied. Each trained model is then evaluated on the same test set ($\mathfrak{T}_{\text{VPN}_2}$) to isolate the effect of injecting limited in-domain (Bot-IoT) supervision into an otherwise out-

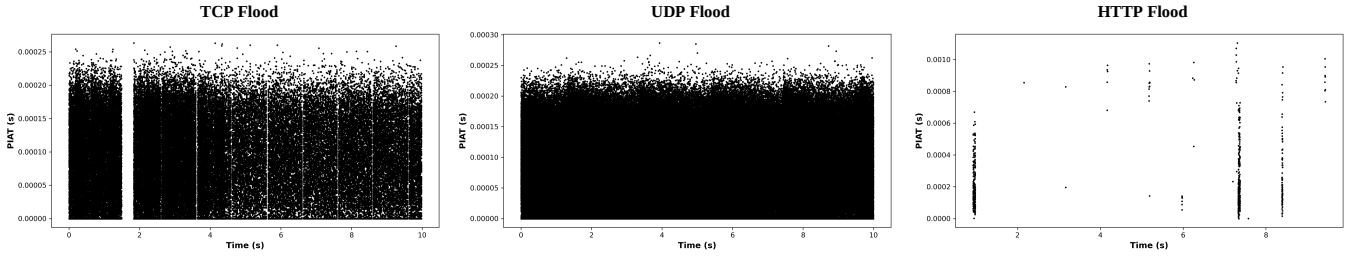

Fig. 12. Temporal PIAT patterns of Bot-IoT DDoS attack vectors.

TABLE V
CROSS-DATASET EVALUATION RESULTS

Test data / Train data	$\mathfrak{T}_{\text{VPN}_2}$															
	Overall Performance (%)				Class-wise Performance (%)											
					TCP				UDP				HTTP			
	ACC	P_{macro}	R_{macro}	$F1_{\text{macro}}$	ACC	Precision	Recall	F1-score	ACC	Precision	Recall	F1-score	ACC	Precision	Recall	F1-score
Bot-IoT 100% ($\frac{I}{C_0} = 3600$)	97.44	97.45	97.44	97.45	96.89	95.56	96.89	96.22	95.61	96.79	95.61	96.20	99.83	100	99.83	99.92
Bot-IoT 80% ($\frac{I}{C_0} = 2880$) **IEEE P2668-MLIDD 20%** ($\frac{I}{C_0} = 720$)	96.80	96.83	96.80	96.79	93.56	96.73	93.56	95.11	96.83	93.81	96.83	95.30	100	99.94	100	99.97
Bot-IoT 50% ($\frac{I}{C_0} = 1800$) **IEEE P2668-MLIDD 50%** ($\frac{I}{C_0} = 1800$)	95.04	95.16	95.04	95.03	95.03	89.94	95.83	92.79	89.33	95.66	89.33	92.39	99.94	99.89	99.94	99.92
Bot-IoT 10% ($\frac{I}{C_0} = 360$) **IEEE P2668-MLIDD 90%** ($\frac{I}{C_0} = 3240$)	93.74	94.06	93.74	93.74	95.94	86.70	95.94	91.09	85.78	95.54	85.78	90.40	99.50	99.94	99.50	99.72
Bot-IoT 1% ($\frac{I}{C_0} = 36$) **IEEE P2668-MLIDD 99%** ($\frac{I}{C_0} = 3564$)	90.57	90.91	90.57	90.66	90.22	83.07	90.22	86.50	86.06	90.01	86.06	87.99	95.44	99.65	95.44	97.50
IEEE P2668-MLIDD 100% ($\frac{I}{C_0} = 3600$)	Near-chance performance															

of-domain training process.

As shown in the last row, training solely on IEEE P2668-MLIDD yields near-chance performance on Bot-IoT, confirming the severity of the cross-dataset shift. However, this behavior changes sharply once even a very small Bot-IoT fraction is introduced into the training set. In particular, adding only 1% Bot-IoT data, corresponding to just 36 samples per class, while keeping the remaining 99% from IEEE P2668-MLIDD, increases the overall accuracy to 90.57%. This gain is also reflected in the class-wise scores, where TCP, UDP, and HTTP reach accuracy of 90.22%, 86.06%, and 95.44%, respectively. As the Bot-IoT share increases further, performance steadily approaches the in-domain upper bound (97.44%), indicating that modest site-specific labeling can largely mitigate the distribution mismatch without requiring full retraining on a large in-domain corpus. This is consistent with practical deployment, where the model can be periodically updated online using a small number of newly labeled, site-specific samples, avoiding full retraining while correcting drift.

D. Visualization of Intermediate Layers With t-SNE

In this section, t-SNE is applied to intermediate model activations to quantify the contribution of each architectural block, DWT, CNN, BiLSTM, and the attention module, to the learned representation and to class-wise separability. For a fair comparison, the same held-out test subset is embedded at every stage, so that changes in cluster compactness and inter-class spacing can be attributed to the corresponding transformation. The resulting embeddings are reported in Fig. 13, together with three clustering indices, DB, Sil, and CH, defined and interpreted in Section III-D.

At the raw input stage, the embedding exhibits substantial overlap and poorly defined class boundaries. This is consistent with the clustering indices, which indicate weak separability (DB = 40.215, Sil = -0.139, CH = 1571.8). Local neighborhoods are weakly aligned with class labels, suggesting that the original PIAT-based feature space provides limited discriminative structure for reliable classification.

Applying the DWT, which is linear and unsupervised, improves alignment by reducing redundancy and suppressing noise-like variation induced by cryptographic timing overheads within the PIAT features. As a result, intra-class dispersion decreases, and the embedding becomes more compact, reflected by DB = 18.124, Sil = 0.145, and CH = 5643.0. However, notable inter-class overlap persists, for example among ICMP, TCP, and UDP flood traffic, and across different intensity levels of MQTT floods. This behavior is expected, since the DWT does not incorporate class-specific (supervised) learning.

At the CNN stage, separation becomes clearer for multiple traffic types, including improved differentiation between HTTP and CoAP, and increased separability among high-intensity ICMP, UDP, and TCP floods (with ICMP high becoming more isolated). By extracting spatially localized patterns from the DWT-transformed inputs, the CNN increases intra-class compactness and begins to introduce inter-class margins. The indices capture this overall gain, with Sil increasing to 0.202 and CH rising to 7298.1. In contrast, DB increases to 30.493 relative to the DWT stage, indicating non-

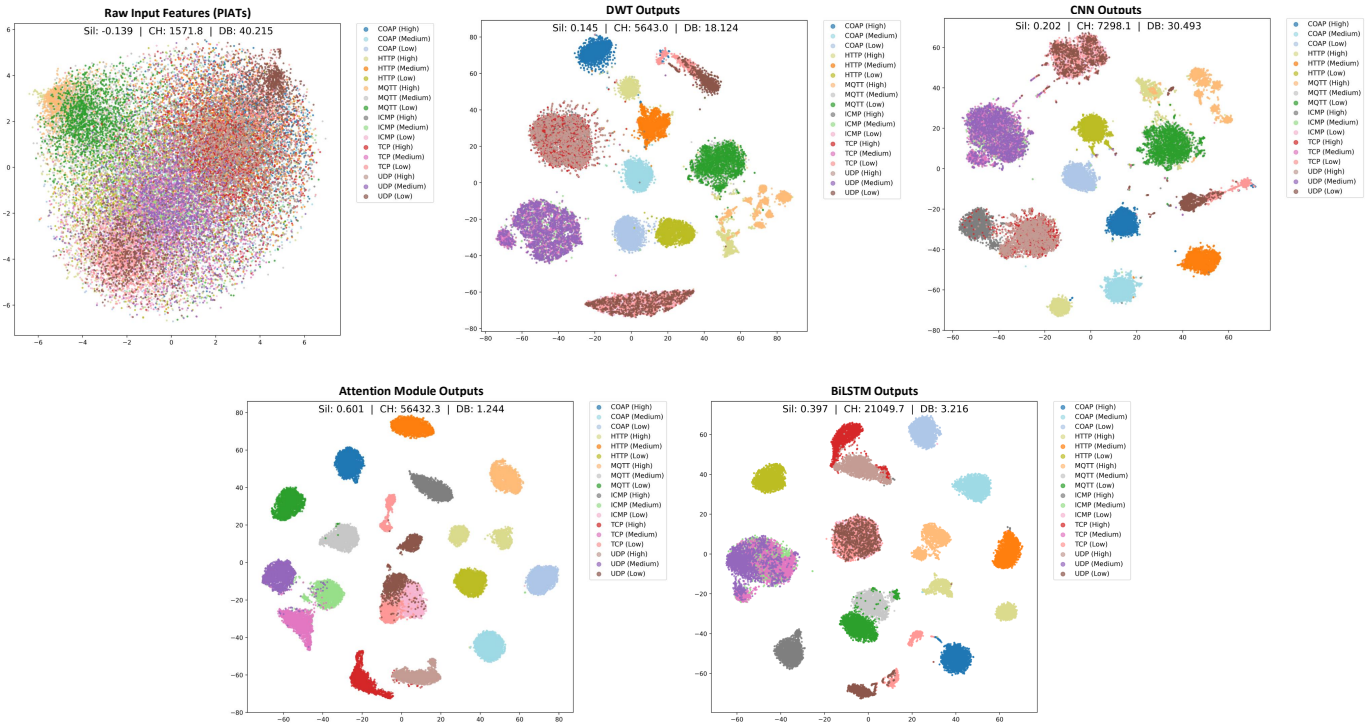

Fig. 13. t-SNE embeddings of intermediate activations illustrate how successive architectural blocks increase class separability by reducing overlap and sharpening cluster boundaries.

uniform cluster compactness and persistent overlap in parts of the representation space. This is consistent with residual mixing among ICMP, TCP, and UDP floods at low and medium intensities, overlap between high-intensity UDP and TCP floods, and the continued partial mixing of MQTT floods across intensity levels.

Next, incorporating the BiLSTM substantially improves cluster quality by exploiting temporal dependencies that help disambiguate samples with similar spatial characteristics. Intra-class spread contracts and inter-class margins expand, yielding DB = 3.216, Sil = 0.397, and CH = 21049.7. Overlap among high-intensity ICMP, TCP, and UDP floods is largely resolved, while some overlap remains at low and medium intensities. For MQTT floods, mixing between medium and low intensities is strongly reduced, with only minor residual overlap.

Finally, the attention mechanism further improves separability by emphasizing context-relevant temporal features, thereby refining the representation and resolving ambiguities that remain after the BiLSTM. This stage achieves the strongest clustering performance overall, with DB reduced to 1.244 and Sil and CH increased to 0.601 and 56432.3, respectively. Overall, the attention-stage embedding shows markedly improved separation among TCP, UDP, and ICMP floods at low and medium intensities, with additional gains in discrimination across the remaining attack types and their respective intensity levels.

E. Comparative Analysis

1) Qualitative Comparison: When it comes to comparing prior work on DDoS attack detection with our study, the first constraint arises from the fact that a large proportion of existing research—particularly those relying on conventional computer network datasets [9], [50]–[52]—targets primarily traditional protocol families such as TCP, UDP, and HTTP. This narrow focus results in insufficient consideration of IoT-specific attack scenarios leveraging protocols like MQTT and CoAP, which limits the ability of these frameworks to generalize to IoT environments. Although IoT-focused datasets, testbeds, and models have been proposed [10], [35], [53], [54], they mostly frame the task as binary detection rather than multi-class attack vector identification [10], or they often exclude the overhead and complexity introduced by security mechanisms from their assumptions [35]. As a result, evaluation under encrypted settings is sparse, so it remains unclear whether the input features used in those studies stay available and informative once encryption is present. Only a few works report results under encrypted conditions [38], and even these do not incorporate attack intensity-aware classification. Table VI summarizes the scope and assumptions of these studies for a qualitative comparison. Several items in Table VI share parts of our setup, yet ours is the only study that satisfies all criteria jointly.

2) Quantitative Comparison: To enable a meaningful and unbiased quantitative comparison with existing approaches, it is essential to evaluate all models under consistent experimental conditions. This includes using identical datasets and

TABLE VI
SCOPE AND ASSUMPTIONS OF PRIOR STUDIES COMPARED TO OUR WORK

Ref	Dataset	IoT-Specific Protocols Included?	Multi-Class Classification Performed?	Evaluated Under Encrypted Conditions?	Intensity Quantification as Part of Classification?
Doriguzzi-Corin *et al.* [9]	ISCX2012, CIC2017, CSECIC2018	× No	× No	× No	× No
Liu *et al.* [10]	IEEE P2668-MLIDD	✓ Yes	× No	× No	× No
Ali *et al.* [35]	CICIoT2023, CICIoMT2024	✓ Yes	✓ Yes	× No	× No
Zhu *et al.* [38]	ToN-IoT, BoT-IoT, ISCX VPN-NonVPN	✓ Yes	✓ Yes	✓ Yes	× No
This study	IEEE P2668-MLIDD, BoT-IoT	✓ Yes	✓ Yes	✓ Yes	✓ Yes

TABLE VII
COMPARATIVE PERFORMANCE OF AVI MODELS

Architectural Backbone	DWT-Based Preprocessing Applied	$\mathfrak{T}_{\text{Base}_1}$				$\mathfrak{T}_{\text{TLS}_1}$				$\mathfrak{T}_{\text{DTLS}_1}$				$\mathfrak{T}_{\text{VPN}_1}$			
		ACC	P_{macro}	R_{macro}	$F1_{\text{macro}}$	ACC	P_{macro}	R_{macro}	$F1_{\text{macro}}$	ACC	P_{macro}	R_{macro}	$F1_{\text{macro}}$	ACC	P_{macro}	R_{macro}	$F1_{\text{macro}}$
Shallow ML models																	
Random Forest	No	0.4715	0.6982	0.7184	0.7042	0.4429	0.6826	0.7029	0.6884	0.4541	0.6890	0.7091	0.6948	0.4278	0.6739	0.6942	0.6796
	Yes	0.6517	0.7810	0.7976	0.7869	0.6292	0.7703	0.7882	0.7767	0.6222	0.7674	0.7855	0.7738	0.6088	0.7619	0.7804	0.7684
k-Nearest Neighbors	No	0.4340	0.4801	0.5559	0.4750	0.3943	0.4535	0.5338	0.4521	0.3971	0.4559	0.5357	0.4543	0.2848	0.4061	0.5004	0.4024
	Yes	0.4989	0.7124	0.7324	0.7184	0.4744	0.6996	0.7199	0.7057	0.4804	0.7029	0.7231	0.7089	0.4570	0.6904	0.7107	0.6963
Support Vector Machine	No	0.4699	0.4939	0.5660	0.4946	0.4491	0.4811	0.5535	0.4806	0.4594	0.4878	0.5596	0.4875	0.4230	0.4640	0.5371	0.4622
	Yes	0.5524	0.7376	0.7572	0.7441	0.5182	0.7219	0.7417	0.7281	0.5350	0.7297	0.7494	0.7360	0.5021	0.7138	0.7339	0.7200
DL models																	
CNN	No	0.6518	0.7813	0.7977	0.7871	0.6274	0.7694	0.7875	0.7759	0.6419	0.7752	0.7928	0.7816	0.6009	0.7587	0.7773	0.7652
	Yes	0.8584	0.8851	0.8795	0.8747	0.8542	0.8814	0.8760	0.8709	0.8561	0.8827	0.8776	0.8726	0.7909	0.8552	0.8565	0.8493
LSTM	No	0.8540	0.8811	0.8758	0.8706	0.8432	0.8711	0.8667	0.8606	0.8470	0.8747	0.8699	0.8642	0.7882	0.8518	0.8546	0.8471
	Yes	0.2848	0.4061	0.5004	0.4024	0.2426	0.3788	0.4701	0.3691	0.2470	0.3821	0.4736	0.3730	0.1950	0.3194	0.4240	0.3176
BiLSTM	No	0.8745	0.8758	0.8745	0.8666	0.8676	0.8934	0.8874	0.8831	0.8678	0.8620	0.8678	0.8580	0.8165	0.8832	0.8740	0.8687
	Yes	0.3565	0.4420	0.5397	0.4468	0.3225	0.4260	0.5226	0.4271	0.3261	0.4278	0.5245	0.4293	0.3016	0.3608	0.4667	0.3617
Hybrid DL models																	
CNN-LSTM	No	0.8937	0.9041	0.8937	0.8899	0.8869	0.8945	0.8869	0.8818	0.8908	0.9002	0.8908	0.8865	0.8642	0.8529	0.8642	0.8533
	Yes	0.9600	0.9643	0.9600	0.9603	0.9555	0.9606	0.9555	0.9558	0.9574	0.9620	0.9574	0.9577	0.9301	0.9395	0.9301	0.9299
CNN-BiLSTM	No	0.9120	0.9234	0.9120	0.9106	0.9077	0.9193	0.9077	0.9057	0.9091	0.9194	0.9091	0.9072	0.8799	0.8848	0.8799	0.8733
	Yes	0.9690	0.9722	0.9690	0.9693	0.9626	0.9665	0.9626	0.9629	0.9635	0.9672	0.9635	0.9638	0.9339	0.9431	0.9339	0.9338
CNN-BiLSTM-ATT	No	0.9360	0.9445	0.9360	0.9360	0.9289	0.9381	0.9289	0.9285	0.9310	0.9397	0.9310	0.9307	0.9051	0.9176	0.9051	0.9030
	Yes	**0.9798**	**0.9811**	**0.9798**	**0.9799**	**0.9731**	**0.9755**	**0.9731**	**0.9733**	**0.9789**	**0.9803**	**0.9789**	**0.9790**	**0.9412**	**0.9489**	**0.9412**	**0.9413**

maintaining uniform assumptions regarding feature availability and informativeness, particularly under heterogeneous network configurations. Accordingly, we benchmark the proposed framework against a set of shallow and deep models using the IEEE P2668-MLIDD dataset, augmented to incorporate cryptographic processing overheads within its timing attributes. All trained models are evaluated on the test partitions defined in Eq. (10), with timing attributes used exclusively as the sole input features. The resulting comparative performance metrics are presented in Table VII.

The results demonstrate that shallow ML models trained directly on raw PIATs, without DWT-based preprocessing, exhibit limited discriminative capability for accurate AVI. DL models such as CNN, and particularly LSTM, and BiLSTM demonstrate better performance, yet their accuracy remains limited, especially in scenarios involving VPN-induced overheads. Hybrid architectures, including CNN-LSTM and CNN-BiLSTM, partially mitigate these limitations by combining spatial feature extraction with temporal modeling. Introducing an attention mechanism further improves these hybrids by enabling the network to focus on the most discriminative spatio-temporal features, yielding higher robustness. Nonetheless, even these enhanced models fail to maintain consistently high accuracy across all traffic categories due to their exclusive reliance on supervised feature learning, which restricts their ability to capture a sufficiently diverse and representative feature space.

The incorporation of DWT-based preprocessing enhances the performance of most evaluated models by reducing redundancy and mitigating noise, thereby yielding more discriminative feature representations prior to classification. Shallow classifiers benefit notably under this configuration, whereas CNN-based architectures exhibit pronounced improvements, as the wavelet-transformed inputs provide spatially structured patterns well-suited to convolutional feature extraction. In contrast, LSTM and BiLSTM models demonstrate reduced accuracy when applied directly to DWT-transformed PIAT features. This degradation arises from the redistribution of temporal dependencies across multiple frequency sub-bands induced by DWT, which diminishes the explicit sequential correlations required by recurrent models. Hybrid architectures (CNN-LSTM, CNN-BiLSTM) mitigate this limitation by leveraging convolutional layers to extract localized spatial patterns from the DWT-processed inputs, thereby supplying the recurrent layers with structured representations that retain residual temporal dynamics. Nonetheless, classification accuracy remains limited for traffic classes with substantial cryptographic overheads, such as VPN-protected flows. Incorporating an attention mechanism partially addresses this limitation by adaptively weighting the most informative features across temporal dimensions, enabling the proposed CNN-BiLSTM-ATT model to achieve an accuracy of 94.12%, as summarized in Table VII.

VII. Impact of Dimensionality Reduction on Computational Efficiency

The accurate identification and quantification of attack vectors in encrypted settings, based solely on timing-related features, remains inherently challenging when using conventional learning models. Nonetheless, advanced architectures, such as hybrid DL models, have demonstrated the ability to partially address these limitations. In particular, the proposed framework, which incorporates a multi-layered processing pipeline, yields notable improvements in classification performance. These accuracy gains, however, come with increased computational overhead, primarily due to the high dimensionality of the input feature matrix, $M \in \mathbb{R}^{1948\times11}$, and the depth of the model. While this level of computational demand aligns with the original deployment assumption of server-side integration (see Fig. 1), where sufficient resources are available and the model operates within expected limits, extending its applicability to alternative integration points across the network topology raises concerns about computational feasibility and underscores the need to explore more lightweight alternatives.

To explore this and provide a substantiated recommendation for reducing the computational overhead of the proposed model, we investigate the impact of input dimensionality reduction on computational efficiency, while systematically tracking potential trade-offs in classification accuracy. Unlike previous sections, all experiments here are performed on a CPU-only setup (Intel Core i7-1185G7), with GPU acceleration deliberately excluded to reflect constrained deployment scenarios.

The aim here is to reduce computational overhead on CPU-based systems by substituting direct processing of full wavelet coefficient sequences with a statistical feature transformation of the wavelet-decomposed data. This strategy, supported by prior studies showing that statistical descriptors reliably capture key temporal and spectral characteristics of the original signals [48], [55], reduces the input dimensionality without significant information loss. For each decomposition level, the coefficient set is compressed into a vector of statistical attributes, as summarized in Table VIII, resulting in a dimensionality reduction from $M \in \mathbb{R}^{1948\times11}$ to $M \in \mathbb{R}^{22\times11}$.

A quantitative evaluation of the model's performance before and after dimensionality reduction reveals a favorable accuracy–efficiency trade-off. As shown in Table IX, the application of statistical feature transformation yields only a marginal decrease in classification accuracy while reducing convergence time by approximately a factor of 11.6 and accelerating average inference by approximately a factor of ~ 6.8. These findings indicate that such transformation effectively mitigates computational overhead through dimensionality reduction while preserving competitive accuracy, confirming its suitability for deployment in resource-constrained environments requiring real-time operation.

TABLE VIII
Statistical Descriptors Applied to Wavelet Coefficients

ID	Feature Name
1	Mean
2	Standard Deviation
3	Maximum Value
4	Minimum Value
5	Median
6	Range (max-min)
7	Interquartile Range (middle 50% spread)
8	Energy (sum of squares)
9	Normalized Energy (energy per sample)
10	Absolute Energy (sum of absolute values)
11	Root Mean Square (RMS)
12	Skewness (distribution asymmetry)
13	Kurtosis (distribution peakedness)
14	Entropy (signal randomness)
15	Zero-Crossing Rate (rate of sign changes)
16	Max-to-Mean Ratio
17	Std-to-Mean Ratio
18	Coefficient of Variation (normalized variability)
19	Signal-to-Noise Ratio Estimate (SNR)
20	Hjorth Activity (signal variance)
21	Hjorth Mobility (mean frequency measure)
22	Hjorth Complexity (frequency variation)

VIII. Conclusion

This study proposes a unified framework for the identification and quantification of DDoS attack vectors in IoT networks, with a key focus on remaining usable and keeping strong detection performance under encrypted communications and limited site-specific samples during training. The framework comprises three main modules: (i) a traffic capture mechanism based on overlapping sliding time windows, (ii) a feature extraction pipeline that uses DWT to generate time-frequency representations from PIAT sequences, and (iii) a hybrid deep classifier that integrates CNN, BiLSTM, and an attention mechanism. By relying only on timing-based features, the framework remains applicable when payload content and packet-level signatures are hidden or altered and when temporal uncertainty increases due to encryption-induced processing overheads introduced by protocols such as TLS, DTLS, and VPN. The DWT stage strengthens the representation of temporal patterns while reducing the impact of noise-like timing distortions introduced by cryptographic overhead. The DL classifier then learns spatial and sequential dependencies across wavelet-derived features, where each module contributes to improved class separation and overall performance.

The overall architecture was configured through a modular tuning process, where the impact of each component was evaluated in the context of the full pipeline. The effect of individual modules on classification performance and feature-space structure was further examined through intermediate representation visualizations, including t-SNE. Extensive evaluation on the IEEE P2668-MLIDD and Bot-IoT datasets shows that the proposed method achieves high accuracy

TABLE IX
IMPACT OF INPUT DIMENSIONALITY ON COMPUTATIONAL OVERHEAD AND PERFORMANCE

Input Dimension	Avg. Inference Time (Second/Sample)	Convergence Time (Hours)	Max Accuracy Achieved (%)			
			$\mathfrak{T}_{\text{Base}_1}$	$\mathfrak{T}_{\text{TLS}_1}$	$\mathfrak{T}_{\text{DTLS}_1}$	$\mathfrak{T}_{\text{VPN}_1}$
1948×11	~ 0.133	~ 389	97.98	97.31	97.89	94.12
22×11	~ 0.0195	~ 33.5	95.84	95.10	95.11	93.00

(around 97%) even under VPN configuration and when only limited site-specific samples are available in the training set. Additional experiments also indicate that the model maintains strong performance under input dimensionality reduction, exhibiting only a 1-2% drop in accuracy. These results support deployment in resource-constrained settings with minimal loss in detection effectiveness.

Future work will extend the model beyond flooding-based DDoS vectors to cover additional threats, such as scanning activity, information theft, malware injection, and advanced persistent threats, by incorporating more diverse training data from heterogeneous network environments and testbeds. In addition, domain-adaptation techniques and unsupervised learning methods will be explored, alongside adaptive and online learning mechanisms, to improve cross-dataset generalization and preserve detection quality as traffic conditions, network characteristics, and attack behaviors change over time [56]. Moreover, acquiring and evaluating the model on real TLS, DTLS, and VPN traffic traces would significantly strengthen the claim of encryption-agnostic detection and may reveal timing-related characteristics that are not fully captured in simulated settings. Reinforcement learning-based mitigation strategies that adjust response policies based on the detected vector, intensity, and recurrence are another direction of interest [57]. Finally, adaptive wavelet modules that tune resolution parameters based on observed traffic characteristics [58], along with evaluation on anonymized traffic (e.g., Tor), will be considered to further improve cross-environment performance and practical relevance.

ACKNOWLEDGMENT

The work presented in this paper was conducted within the framework of two projects:

1) CYPRESS project (https://cypress-project.be), supported by the FPS Economy of Belgium through the Energy Transition Fund.
2) FUNCTIONARY project, a Catalisti initiative supported by VLAIO, (HBC.2024.0786), the industry federation Essenscia, and other industrial and academic partners in Belgium.

The authors thank the anonymous reviewers for their valuable comments and helpful suggestions.

REFERENCES

[1] A. Djenna, S. Harous, and D. E. Saidouni, "Internet of things meet internet of threats: New concern cyber security issues of critical cyber infrastructure," *Applied Sciences*, vol. 11, no. 10, 2021. [Online]. Available: https://www.mdpi.com/2076-3417/11/10/4580

[2] S. Dange and M. Chatterjee, "IoT botnet: The largest threat to the IoT network," in *Data Communication and Networks*, L. C. Jain, G. A. Tsihrintzis, V. E. Balas, and D. K. Sharma, Eds. Singapore: Springer Singapore, 2020, pp. 137–157.

[3] I. Stellios, P. Kotzanikolaou, M. Psarakis, C. Alcaraz, and J. Lopez, "A survey of IoT-enabled cyberattacks: Assessing attack paths to critical infrastructures and services," *IEEE Communications Surveys Tutorials*, vol. 20, no. 4, pp. 3453–3495, 2018.

[4] I. Makhdoom, M. Abolhasan, J. Lipman, R. P. Liu, and W. Ni, "Anatomy of threats to the Internet of Things," *IEEE Communications Surveys Tutorials*, vol. 21, no. 2, pp. 1636–1675, 2019.

[5] R. Vishwakarma and A. K. Jain, "A survey of DDoS attacking techniques and defence mechanisms in the IoT network," *Telecommunication systems*, vol. 73, no. 1, pp. 3–25, 2020.

[6] S. Chaudhary and P. K. Mishra, "DDoS attacks in industrial IoT: A survey," *Computer Networks*, vol. 236, p. 110015, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1389128623004607

[7] S. M. Almeghlef, A. A.-M. AL-Ghamdi, M. S. Ramzan, and M. Ragab, "Application layer-based denial-of-service attacks detection against IoT-CoAP," *Electronics*, vol. 12, no. 12, 2023. [Online]. Available: https://www.mdpi.com/2079-9292/12/12/2563

[8] S. Lakshminarayana, A. Praseed, and P. S. Thilagam, "Securing the IoT application layer from an MQTT protocol perspective: Challenges and research prospects," *IEEE Communications Surveys Tutorials*, vol. 26, no. 4, pp. 2510–2546, 2024.

[9] R. Doriguzzi-Corin, S. Millar, S. Scott-Hayward, J. Martínez-del Rincón, and D. Siracusa, "Lucid: A practical, lightweight deep learning solution for DDoS attack detection," *IEEE Transactions on Network and Service Management*, vol. 17, no. 2, pp. 876–889, 2020.

[10] Y. Liu, K.-F. Tsang, C. K. Wu, Y. Wei, H. Wang, and H. Zhu, "IEEE P2668-compliant multi-layer IoT-DDoS defense system using deep reinforcement learning," *IEEE Transactions on Consumer Electronics*, vol. 69, no. 1, pp. 49–64, 2023.

[11] R. Doshi, N. Apthorpe, and N. Feamster, "Machine learning DDoS detection for consumer internet of things devices," in *2018 IEEE Security and Privacy Workshops (SPW)*, 2018, pp. 29–35.

[12] Kamaldeep, M. Malik, and M. Dutta, "Feature engineering and machine learning framework for DDoS attack detection in the standardized internet of things," *IEEE Internet of Things Journal*, vol. 10, no. 10, pp. 8658–8669, 2023.

[13] S. Ali and Y. Li, "Learning multilevel auto-encoders for DDoS attack detection in smart grid network," *IEEE Access*, vol. 7, pp. 108 647–108 659, 2019.

[14] M. T. Paracha, D. J. Dubois, N. Vallina-Rodriguez, and D. Choffnes, "IoTLS: understanding TLS usage in consumer IoT devices," in *Proceedings of the 21st ACM Internet Measurement Conference*, ser. IMC '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 165–178. [Online]. Available: https://doi.org/10.1145/3487552.3487830

[15] G. Lessa dos Santos, V. T. Guimarães, G. da Cunha Rodrigues, L. Z. Granville, and L. M. R. Tarouco, "A DTLS-based security architecture for the Internet of Things," in *2015 IEEE Symposium on Computers and Communication (ISCC)*, 2015, pp. 809–815.

[16] M. Juma, A. A. Monem, and K. Shaalan, "Hybrid end-to-end VPN security approach for smart IoT objects," *Journal of Network and Computer Applications*, vol. 158, p. 102598, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1084804520300722

[17] L. Zhu, X. Tang, M. Shen, X. Du, and M. Guizani, "Privacy-preserving DDoS attack detection using cross-domain traffic in software defined networks," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 3, pp. 628–643, 2018.

[18] Z. Wang, K. W. Fok, and V. L. Thing, "Machine learning for encrypted malicious traffic detection: Approaches, datasets and comparative study,"

Computers Security, vol. 113, p. 102542, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167404821003667

[19] A. J. Pinheiro, J. de M. Bezerra, C. A. Burgardt, and D. R. Campelo, "Identifying IoT devices and events based on packet length from encrypted traffic," *Computer Communications*, vol. 144, pp. 8–17, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0140366419300052

[20] Y. Liu, K.-F. Tsang, C. K. Wu, Y. Wei, H. Wang, and H. Zhu, "IEEE P2668-compliant multi-layer IoT-DDoS dataset (IEEE P2668-MLIDD)," 2022. [Online]. Available: https://dx.doi.org/10.21227/j0f2-8h67

[21] 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," *Future Generation Computer Systems*, vol. 100, pp. 779–796, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167739X18327687

[22] N. Koroniotis, N. Moustafa, E. Sitnikova, and J. Slay, "Towards developing network forensic mechanism for botnet activities in the IoT based on machine learning techniques," in *Mobile Networks and Management*, J. Hu, I. Khalil, Z. Tari, and S. Wen, Eds. Cham: Springer International Publishing, 2018, pp. 30–44.

[23] N. Koroniotis, N. Moustafa, and E. Sitnikova, "A new network forensic framework based on deep learning for Internet of Things networks: A particle deep framework," *Future Generation Computer Systems*, vol. 110, pp. 91–106, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167739X19325105

[24] N. Koroniotis, N. Moustafa, F. Schiliro, P. Gauravaram, and H. Janicke, "A holistic review of cybersecurity and reliability perspectives in smart airports," *IEEE Access*, vol. 8, pp. 209 802–209 834, 2020.

[25] N. Koroniotis, N. Moustafa, E. Sitnikova, and B. Turnbull, "The BoT-IoT dataset," UNSW Canberra Cyber, University of New South Wales, 2018, [Online]. Available: https://research.unsw.edu.au/projects/bot-iot-dataset.

[26] R. Doshi, N. Apthorpe, and N. Feamster, "Machine learning DDoS detection for consumer internet of things devices," in *2018 IEEE Security and Privacy Workshops (SPW)*, 2018, pp. 29–35.

[27] S. H. Rafique, A. Abdallah, N. S. Musa, and T. Murugan, "Machine learning and deep learning techniques for internet of things network anomaly detection—current research trends," *Sensors*, vol. 24, no. 6, 2024. [Online]. Available: https://www.mdpi.com/1424-8220/24/6/1968

[28] A. M. Abdallah, A. Saif Rashed Obaid Alkaabi, G. Bark Nasser Douman Alameri, S. H. Rafique, N. S. Musa, and T. Murugan, "Cloud network anomaly detection using machine and deep learning techniques— recent research advancements," *IEEE Access*, vol. 12, pp. 56 749–56 773, 2024.

[29] M. F. Saiyedand and I. Al-Anbagi, "Deep ensemble learning with pruning for DDoS attack detection in IoT networks," *IEEE Transactions on Machine Learning in Communications and Networking*, vol. 2, pp. 596–616, 2024.

[30] Zabeehullah, F. Arif, Q. Mazhar Ul Haq, N. Ali Khan, I. Ud Din, A. Almogren, M. Ali Khan, O. Alsaleh, and M. Guizani, "Hybrid CNN-LSTM model for DDoS attack detection in Internet of Things-based healthcare industry 5.0," *IEEE Internet of Things Journal*, vol. 12, no. 22, pp. 46 075–46 082, 2025.

[31] A. A. Najar and S. Manohar Naik, "Cyber-secure SDN: A CNN-based approach for efficient detection and mitigation of DDoS attacks," *Computers Security*, vol. 139, p. 103716, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167404824000178

[32] M. S. E. Sayed, N.-A. Le-Khac, M. A. Azer, and A. D. Jurcut, "A flow-based anomaly detection approach with feature selection method against DDoS attacks in SDNs," *IEEE Transactions on Cognitive Communications and Networking*, vol. 8, no. 4, pp. 1862–1880, 2022.

[33] Y. A. Abid, J. Wu, G. Xu, S. Fu, and M. Waqas, "Multilevel deep neural network approach for enhanced distributed denial-of-service attack detection and classification in software-defined Internet of Things networks," *IEEE Internet of Things Journal*, vol. 11, no. 14, pp. 24 715–24 725, 2024.

[34] H. Nandanwar and R. Katarya, "TL-BILSTM IoT: transfer learning model for prediction of intrusion detection system in IoT environment," *International Journal of Information Security*, vol. 23, no. 2, pp. 1251–1277, 2024.

[35] M. Ali, Y. Saleem, S. Hina, and G. A. Shah, "DDoSViT: IoT DDoS attack detection for fortifying firmware Over-The-Air (OTA) updates using vision transformer," *Internet of Things*, vol. 30, p. 101527, 2025. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S254266052500040X

[36] T. Shapira and Y. Shavitt, "FlowPic: A generic representation for encrypted traffic classification and applications identification," *IEEE Transactions on Network and Service Management*, vol. 18, no. 2, pp. 1218–1232, 2021.

[37] K. Lin, X. Xu, and H. Gao, "TSCRNN: A novel classification scheme of encrypted traffic based on flow spatiotemporal features for efficient management of IIoT," *Computer Networks*, vol. 190, p. 107974, 2021.

[38] S. Zhu, X. Xu, H. Gao, and F. Xiao, "CMTSNN: A deep learning model for multiclassification of abnormal and encrypted traffic of internet of things," *IEEE Internet of Things Journal*, vol. 10, no. 13, pp. 11 773–11 791, 2023.

[39] G. Long and Z. Zhang, "Deep encrypted traffic detection: An anomaly detection framework for encryption traffic based on parallel automatic feature extraction," *Computational Intelligence and Neuroscience*, vol. 2023, no. 1, p. 3316642, 2023. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1155/2023/3316642

[40] I. H. Ji, J. H. Lee, M. J. Kang, W. J. Park, S. H. Jeon, and J. T. Seo, "Artificial intelligence-based anomaly detection technology over encrypted traffic: A systematic literature review," *Sensors*, vol. 24, no. 3, 2024. [Online]. Available: https://www.mdpi.com/1424-8220/24/3/898

[41] D. Canavese, L. Regano, C. Basile, G. Ciravegna, and A. Lioy, "Encryption-agnostic classifiers of traffic originators and their application to anomaly detection," *Computers Electrical Engineering*, vol. 97, p. 107621, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0045790621005528

[42] T. Bakhshi and B. Ghita, "Anomaly detection in encrypted internet traffic using hybrid deep learning," *Security and Communication Networks*, vol. 2021, no. 1, p. 5363750, 2021. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1155/2021/5363750

[43] T. Behdadnia, G. Deconinck, C. Ozkan, and D. Singelee, "Encrypted traffic classification for early-stage anomaly detection in power grid communication network," in *2023 IEEE PES Innovative Smart Grid Technologies Europe (ISGT EUROPE)*, 2023, pp. 1–6.

[44] C. Kolias, G. Kambourakis, A. Stavrou, and J. Voas, "DDoS in the IoT: Mirai and other botnets," *Computer*, vol. 50, no. 7, pp. 80–84, 2017.

[45] I. Sharafaldin, A. H. Lashkari, S. Hakak, and A. A. Ghorbani, "Developing realistic distributed denial of service (ddos) attack dataset and taxonomy," in *2019 International Carnahan Conference on Security Technology (ICCST)*, 2019, pp. 1–8.

[46] A. Guerra-Manzanares, J. Medina-Galindo, H. Bahsi, and S. Nõmm, "Medbiot: Generation of an iot botnet dataset in a medium-sized iot network," in *Proceedings of the 6th International Conference on Information Systems Security and Privacy - ICISSP*, INSTICC. SciTePress, 2020, pp. 207–218.

[47] T. Guo, T. Zhang, E. Lim, M. López-Benítez, F. Ma, and L. Yu, "A review of wavelet analysis and its applications: Challenges and opportunities," *IEEE Access*, vol. 10, pp. 58 869–58 903, 2022.

[48] T. Behdadnia, K. Thoelen, and G. Deconinck, "Early-stage anomaly detection in IT-OT power grid communication networks using wavelet transform and hybrid deep learning," *IEEE Transactions on Smart Grid*, vol. 16, no. 4, pp. 3305–3322, 2025.

[49] S. Mallat, "A theory for multiresolution signal decomposition: the wavelet representation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 11, no. 7, pp. 674–693, 1989.

[50] J. Ye, X. Cheng, J. Zhu, L. Feng, and L. Song, "A DDoS attack detection method based on SVM in software defined network," *Security and Communication Networks*, vol. 2018, no. 1, p. 9804061, 2018. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1155/2018/9804061

[51] R. Doriguzzi-Corin, S. Millar, S. Scott-Hayward, J. Martínez-del Rincón, and D. Siracusa, "Lucid: A practical, lightweight deep learning solution for DDoS attack detection," *IEEE Transactions on Network and Service Management*, vol. 17, no. 2, pp. 876–889, 2020.

[52] J. D. Gadze, A. A. Bamfo-Asante, J. O. Agyemang, H. Nunoo-Mensah, and K. A.-B. Opare, "An investigation into the application of deep learning in the detection and mitigation of DDOS attack on SDN controllers," *Technologies*, vol. 9, no. 1, 2021. [Online]. Available: https://www.mdpi.com/2227-7080/9/1/14

[53] F. Hussain, S. G. Abbas, G. A. Shah, I. M. Pires, U. U. Fayyaz, F. Shahzad, N. M. Garcia, and E. Zdravevski, "A framework for malicious traffic detection in IoT healthcare environment," *Sensors*, vol. 21, no. 9, 2021. [Online]. Available: https://www.mdpi.com/1424-8220/21/9/3025

[54] J. Granjal, J. M. Silva, and N. Lourenço, "Intrusion detection and prevention in CoAP wireless sensor networks using anomaly detection," *Sensors*, vol. 18, no. 8, 2018. [Online]. Available: https://www.mdpi.com/1424-8220/18/8/2445

[55] J. J. Q. Yu, Y. Hou, and V. O. K. Li, "Online false data injection attack detection with wavelet transform and deep neural networks," *IEEE*

Transactions on Industrial Informatics, vol. 14, no. 7, pp. 3271–3280, 2018.

[56] V. Tong, C. Dao, H.-A. Tran, D. Tran, H. Thi Thanh Binh, T. Hoang-Nam, and T. X. Tran, "Encrypted traffic classification through deep domain adaptation network with smooth characteristic function," *IEEE Transactions on Network and Service Management*, vol. 22, no. 1, pp. 331–343, 2025.

[57] I. Ortega-Fernandez and F. Liberati, "A review of denial of service attack and mitigation in the smart grid using reinforcement learning," *Energies*, vol. 16, no. 2, 2023. [Online]. Available: https://www.mdpi.com/1996-1073/16/2/635

[58] M. Wang and X. Sun, "Wavemamba: Intrusion detection in iot systems using a mamba module with embedded wavelet transform," *IEEE Internet of Things Journal*, vol. 13, no. 2, pp. 3418–3432, 2026.