SoK: Decoding the Enigma of Encrypted Network Traffic Classifiers

Nimesha Wickramasinghe*, Arash Shaghaghi*, Gene Tsudik†, Sanjay Jha*
*School of Computer Science and Engineering, The University of New South Wales, Sydney, Australia.
†School of Information & Computer Sciences, University of California Irvine, USA

***Abstract*—The adoption of modern encryption protocols such as TLS 1.3 has significantly challenged traditional network traffic classification (NTC) methods. As a consequence, researchers are increasingly turning to machine learning (ML) approaches to overcome these obstacles. This paper analyses ML-based NTC studies by developing a taxonomy of their design choices, benchmarking suites, and prevalent assumptions impacting classifier performance. Through this systematization, we demonstrate widespread reliance on outdated datasets, oversights in design choices, and the consequences of unsubstantiated assumptions. Our evaluation reveals that the majority of proposed encrypted traffic classifiers have mistakenly utilized unencrypted traffic due to the use of legacy datasets. Furthermore, by conducting 348 feature occlusion experiments on state-of-the-art classifiers, we show how oversights in NTC design choices lead to overfitting and validate or refute prevailing assumptions with empirical evidence. By highlighting lessons learned, we offer strategic insights, identify emerging research directions, and recommend best practices to support the development of real-world applicable NTC methodologies.**

1. Introduction

Network Traffic Classification (NTC) is a fundamental process that identifies and categorizes traffic into predefined classes. NTC is crucial for various applications, including network management and security, Quality of Service (QoS) provisioning, and lawful interception. However, the increasing adoption of encryption protocols, particularly Transport Layer Security (TLS) 1.3, has introduced significant challenges to traditional NTC methods. Encrypted traffic obscures payload content, rendering many conventional classification techniques ineffective and necessitating new approaches that do not rely on signatures in plain-text payloads (e.g., deep packet inspection [21]). As a result, researchers are increasingly turning to machine learning (ML) to address these challenges [50], [92].

Despite the pressing need for effective NTC in the context of modern encrypted traffic, current research efforts face several challenges. One of them is the reliance on outdated datasets collected before 2018 [19], [27], [65], [87]. We show that these legacy datasets do not accurately reflect the characteristics of contemporary network traffic, particularly with the adoption of TLS 1.3. Furthermore, such datasets often contain unencrypted traffic or utilize deprecated cipher suites (e.g., 3DES and RC4), leading to deceptive results when training and evaluating ML models.

Another significant challenge stems from design choices when developing NTC models. Many studies overlook the potential for overfitting due to session-specific artifacts, which are often uninformative (i.e., initialized using pseudo-random values during session establishment). We show that this results in classifiers that perform well on test data but fail to generalize to real-world scenarios, undermining their robustness and reliability.

Furthermore, the field is fraught with unsubstantiated and often conflicting assumptions. Many studies assume that imperfect randomness of ciphers causes discernible patterns in encrypted payloads, which can be exploited for NTC [10], [28], [33], [50], [55], [74], [92]. By making this assumption increasingly questionable, TLS 1.3 guarantees that the only learnable characteristic of a ciphertext is its length [66]. Additionally, there are disputes over the impact of practices such as truncating or padding payload data on classification performance [78]. These conflicting views create confusion and hinder the development of effective NTC methodologies.

To the best of our knowledge, these methodological pitfalls across the entire classification pipeline have not received sufficient scrutiny. Therefore, we address outstanding challenges in NTC through this study by making the following contributions:

1) **Systematization of Knowledge**: We comprehensively analyse NTC studies, identifying their design choices and benchmarking suites.
2) **Discuss pitfalls in NTC**: By critically evaluating widely used network traffic datasets, design choices, and common assumptions, we identify and demonstrate common pitfalls in NTC.
3) **CipherSpectrum**: We introduce CipherSpectrum, a contemporary network traffic dataset, to address the limitations of existing datasets. For cipher-agnostic NTC, the dataset uniformly presents traffic sessions encrypted with all three mandated/recommended cipher suites of TLS 1.3.
4) **Strategic insights**: Building on lessons learned, we propose best practices and future research direc-

2025 IEEE Symposium on Security and Privacy (SP) | 979-8-3315-2236-0/25/$31.00 ©2025 IEEE | DOI: 10.1109/SP61157.2025.00165

DOI 10.1109/SP61157.2025.00165

tions that lead to accurate, generalizable and real-world applicable network traffic classifiers.

Scope of the Paper: This paper concentrates on raw-information based NTC [28], [50], [55], [91], which extracts information directly from network traffic, such as packet headers and payloads. While we exclude side-channel based approaches [18], [46], [60], [64] and multimodal methods [2], [15], [48], [49], we recognize that a thorough understanding of raw data is fundamental to NTC. By focusing on how raw data can be meticulously used to classify traffic, we aim to provide insights that can enhance raw information-based classifiers and inform and improve multimodal approaches. The increasing adoption of raw-information-based NTC [61] further underscores the relevance and timeliness of this study.

This paper is organized as follows. In Section 2, we present a taxonomy and classification of existing NTC studies, detailing their design choices, benchmarking suites, and prevalent assumptions. Section 3 provides a systematization of these studies, synthesizing insights and identifying common trends. In Section 4, we outline the unresolved challenges in NTC by formulating specific research questions that address the issues of outdated datasets, design oversights, and unsubstantiated assumptions. Section 5 presents our empirical investigations, where we prove our conjectures and answer the research questions through extensive experiments. Finally, in Section 6, we discuss the implications of our findings and highlight emerging research directions.

2. Taxonomy and Classification

In this section, we lay the groundwork for our network traffic classification (NTC) study using raw information. We introduce a taxonomy that organizes the various aspects of classifier development, focusing on the key choices made throughout the process. Broadly, these decisions fall into two main categories: design choices and benchmarking choices, which are discussed in Section 2.1 and Section 2.2 respectively. This framework serves as a basis for understanding and analyzing the development and evaluation of network traffic classifiers in this paper.

2.1. Design Choices

The design of a Network Traffic Classification (NTC) system involves several ancillary decisions that impact its functionality, accuracy, and suitability for specific tasks. Key design considerations include the granularity of data, the methods used to extract relevant information, and the selection of features for analysis. For instance, MAC addresses, IP addresses, and protocol ports are commonly referred to as Strong Identification Information (SII) [50]. Incorporating SII constitutes an important design choice within the system's overall framework.

As illustrated in Figure 1, each choice impacts how the classifier processes network data, which is crucial in refining

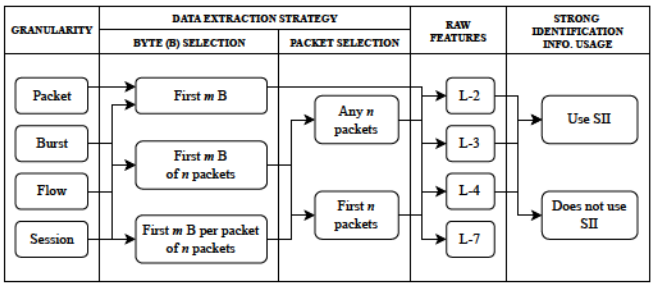


Figure 1. Taxonomy of design choices in raw information based NTC

the classifier's capability. This section delves into these design considerations, structured to reflect the decision-making paths highlighted in Figure 1.

2.1.1. Traffic Granularity. Traffic granularity defines the level of detail at which network traffic is analyzed and classified. To formalize, we represent the set of all traffic packets as $P = \{p^1, ..., p^{|p|}\}$, where each packet $p^i = (x^i, y^i, t^i)$ for $i \in [1, |P|]$. In this notation, x^i represents the 5-tuple consisting of the source IP, destination IP, source port, destination port, and protocol; y^i denotes the packet size in bytes, with $y^i \in (0, \infty)$; and t^i stands for the packet's transmission timestamp, measured in seconds, with $t^i \in [0, \infty)$. The most prominent traffic granularities found in the literature include:

Packet Granularity represents the finest level of analysis, where each packet $p^i = (x^i, b^i, t^i)$ is individually examined.

Burst Granularity analyzes traffic in bursts, or clusters of packets transmitted within short intervals, separated by idle periods. A burst B^n consists of packets that arrive within a specified time window defined by the burst duration threshold Δt. Formally, let B represent the set of all bursts, and each burst b^n can be defined as: $b^n = \{p^i \in P \mid t^i - t^{i-1} \leq \Delta t\}$, where $n \in [1, |B|]$.

Flow Granularity groups all unidirectional packets with the same 5-tuple x^i. Let F represent the set of all flows, where each flow f^k is a collection of packets with the same x^i. Formally, the flow f^k can be defined as:
$f^k = \{p^i \in P \mid x^i = x^j \; \forall \, p^i, p^j \in f^k\}$, where $k \in [1, |F|]$ and $i, j \in [1, |P|]$.

Session Granularity (or *Bi-Flow Granularity*) aggregates traffic flows between two endpoints into a single granularity by treating flows in both directions as part of one conversation. In this approach, the 5-tuple x^i is treated as bi-directional, meaning that flow $A \rightarrow B$ and $B \rightarrow A$ are combined. Formally, let S represent the set of all sessions, where each session s^m consists of paired flows f^k that capture the full interaction between two endpoints

2.1.2. Data Extraction Strategy. Once the traffic granularity is determined, the next essential design choice involves selecting a data extraction strategy. This strategy defines how data is captured from the chosen granularity level and how it is represented. Referring to Figure 1, we categorize data extraction strategies into the following types:

Type 1: First *m* Bytes of Selected Granularity This strategy involves extracting the first m bytes from the entire data segment of the selected granularity level, whether it is packet, burst, flow, or session. This approach treats the selected granularity as a continuous data stream without regard to individual packet boundaries or the number of packets it contains. As shown in Figure 1, this method is the default—and, effectively, the only—option available for packet granularity, where each packet's raw information is extracted for classification.

Type 2: First *m* Bytes of *n* Packets In this method, the classifier extracts the first m bytes collectively from a sequence of n packets within the selected granularity (e.g., burst, flow, or session). If the total size of these n packets is less than m, padding is applied to reach the desired byte length. Conversely, if the total size exceeds m, it is truncated to match the specified size.

Type 3: First *m* Bytes Per Packet of *n* Packets This approach extracts the first m bytes from each of the first n packets within the selected granularity, resulting in a total of $m \times n$ bytes. By preserving packet boundaries and treating the initial bytes of each packet separately, this strategy retains the structural integrity of network traffic.

As Figure 1 depicts, an additional choice for the *Type 2* and *Type 3* strategies is the selection of n packets from which data will be extracted, offering two primary options:

First *n* Packets: This option selects the initial n packets of the chosen granularity (burst, flow, or session) under the assumption that the earliest packets often contain critical information, such as protocol handshakes or initial data exchanges.

Any Consecutive *n* Packets: Alternatively, the n packets can be any consecutive sequence within the granularity, providing flexibility to capture patterns that may occur at various stages of the communication.

2.1.3. Raw Features. Building on the selected traffic granularity and data extraction strategy, the next aspect in designing a network traffic classification (NTC) system is the choice of raw features, as illustrated in Figure 1. Raw features refer to the unprocessed data extracted directly from network traffic without prior aggregation or manipulation. These features play a critical role in NTC as they often capture unique patterns essential for accurate classification. The choice of traffic granularity and data extraction strategy shapes the availability, scope, and type of raw features that can be used. Broadly, these features can be categorized into packet header information and application layer payloads.

Packet headers provide essential metadata about the packet's path through the network.

Layer-2 (L2): The Ethernet layer comprises attributes such as source and destination MAC addresses, which identify devices on the local network.

Layer-3 (L3): The Network layer mainly includes attributes like source and destination IP addresses, which are critical for routing data among networks.

Layer-4 (L4): The Transport layer primarily consists of source and destination ports, along with other flow control and error recovery features.

Layer-7 (L7): The Application layer payloads contain the actual data transmitted by applications. However, with the growing prevalence of encryption protocols, much of this payload data is encrypted, limiting accessibility to plaintext for analysis.

2.1.4. Strong Identification Information (SII). Following the selection of raw features, another critical consideration in designing network traffic classification (NTC) systems is the use of Strong Identification Information (SII), as outlined in Figure 1. SII refers to features directly identifying a user, device, or application, providing highly discriminative information that can significantly enhance classification accuracy.

According to literature, typical examples of SII include: MAC addresses in L2 (ethernet layer), which are unique identifiers for network interfaces; IP addresses in L3 (network layer), serving as numerical labels assigned to devices on a network; and protocol ports in L4 (transport layer), which are numeric identifiers for specific applications or services.

While some studies opt to obfuscate SII to mitigate overfitting, others choose to incorporate SII to leverage its discriminative power.

2.2. Downstream Tasks and Benchmarking

Network traffic classification covers a range of downstream tasks focused on identifying specific network activities, applications, or security threats. Benchmarking in this context involves selecting suitable datasets to rigorously test and validate the classifier's effectiveness in performing the chosen downstream task. Several public datasets are available as benchmarks, each addressing a specific classification task.

A significant area of focus as a downstream task is detecting malicious activities, including malware communications, network intrusions, and botnet traffic. The USTC-TFC2016 dataset [87] provides samples of malicious and benign traffic, enabling the development of classifiers capable of detecting malware within network flows. Similarly, the CIC-IDS2018 dataset [72] contains a range of intrusion scenarios alongside benign traffic. Additionally, CIC-IoT2022 [14] and Bot-IoT [40] datasets focus on IoT environments, offering traffic data that includes both normal device activities and malicious behaviours associated with botnets.

Another critical task involves classifying encrypted and anonymized traffic, such as VPN and Tor communications. The ISCXVPN2016 dataset [19] includes traffic data from both VPN and non-VPN connections, assisting researchers in creating models that can identify the use of VPNs. Similarly, the ISCXTor2016 dataset [27] focuses on Tor traffic classification, offering both Tor and non-Tor traffic samples.

In application identification, web and mobile traffic classification has gained prominence. The CSTNET-TLS1.3

dataset [50] supports web traffic classification by providing samples of web communications using TLS 1.3. The Cross-Platform Application dataset [65] offers network traffic generated by various mobile applications across different platforms, including Android and iOS.

In addition to these public datasets, private datasets collected from operational networks or controlled experiments are often employed to evaluate network traffic classifiers.

At its core, ML in NTC transforms raw network data into meaningful insights by identifying patterns and relationships that may not be explicitly defined. The interplay between data granularity, feature selection, and benchmarking choices shapes how models learn and perform, ultimately determining their effectiveness in real-world scenarios.

3. Systematization

To provide a comprehensive overview of the existing research on Network Traffic Classification (NTC), we conduct a systematic literature survey focusing on studies that utilize raw network traffic information. Our objective is to identify and analyze key literature that discusses the design choices and methodologies pertinent to our scope, particularly in the context of encrypted communications.

3.1. Methodology

We initiate the survey by conducting an extensive search across multiple academic databases, including JSTOR, SCOPUS, EBSCO, and Google Scholar. The search is guided by the following query: *("network traffic" OR "encrypted traffic") AND ("classification" OR "analysis" OR "detection") AND ("review" OR "survey" OR "sok").* This search string was designed to capture a broad spectrum of literature related to NTC, emphasizing works that offer a critical overview or synthesis of the field. Since the first raw-information based network traffic classifier was published in 2015 [1], [10], [33], we filtered our search to studies from 2015 onward.

The initial search yields approximately 152 articles. To refine this pool of literature, we exclude studies that were (a) not directly related to the subject matter, such as those focusing on unrelated networking topics or other domains (e.g., road traffic analysis, NTC studies that do not leverage ML); (b) non-peer-reviewed, to ensure the academic rigour and credibility of the sources; and (c) duplicate studies, to maintain a unique set of references. From the refined list of papers, we selected raw-information based NTC studies that detail their design choices and the benchmarking suites used (see Section 2).

3.2. Culmination

The collected articles are then meticulously analyzed and grouped based on the themes and design choices discussed in Section 2. These categories include traffic granularity, data extraction strategy, raw features used, Strong Identification Information (SII) and dataset usage. We present the culmination of this process in Table 1, which provides a concise reference to support our subsequent analysis and discussions.

4. Unresolved Challenges in NTC

Despite significant advances, several open problems inhibit the development of robust and generalizable classifiers. These issues impede the ability of models to perform effectively in real-world scenarios and limit the practical applicability of research findings. This section identifies and critically examines three primary snags affecting current NTC approaches: reliance on outdated datasets, unsubstantiated assumptions, and issues caused by oversights in design choices.

4.1. Snag 1: Legacy Datasets

The literature summarized in Table 1 reveals that many studies continue to rely on network traffic datasets collected before 2018. This reliance raises concerns since encryption protocols and cipher suites have evolved significantly since then. Specifically, the adoption of protocols such as TLS 1.3 has deprecated algorithms such as 3DES and RC4 and has mandated or strongly recommended the use of enhanced cipher suites. Furthermore, TLS 1.3 encrypts portions of the handshake process previously transmitted in plaintext [66].

These changes reduce the metadata and observable patterns available to network traffic classifiers. Consequently, models developed using older datasets and the assumptions they are based on, may not adequately account for these changes. Applying these models to contemporary network traffic could lead to decreased performance or even complete ineffectiveness.

To address this potential gap, we evaluate the validity and applicability of older datasets in the context of current network conditions. To this end, we formulate the following research questions to determine whether newer datasets are necessary for improving network traffic classification.

S1 - Research Questions

S1-RQ1: Do legacy datasets contain encrypted network traffic that accurately reflects modern protocols?

S1-RQ2: Do encryption algorithms used in legacy datasets remain valid and relevant for contemporary NTC?

4.2. Snag 2: Oversights in Design Choices

We suggest that NTC models often overfit due to a lack of careful feature selection. Feature choices are often

Table 1. RAW INFORMATION BASED NETWORK TRAFFIC CLASSIFICATION

Study		Raw Features Used				Data Extraction		Traffic Granularity				Datasets Used					
Ref.	Year	L2	L3	L4	L7	# Bytes	# Packets	Packet	Burst	Flow	Session	VPN'16	TOR'16	TFC'16	CP'17	Other	Private
[86]	2015	●	●	●	●	1000	-	-	-	-	T1	-	-	-	-	-	✓
[87]	2017	◐	◐	●	●	784	-	-	-	-	T1	-	-	✓	-	-	-
[85]	2017	◐	◐	●	●	784	-	-	-	-	T1	✓	-	-	-	-	-
[94]	2018	-	◐	◐	●	784	3 ✤	-	-	T2	-	✓	-	-	-	-	-
[35]	2018	●	●	●	●	1024	-	-	-	-	T1	✓	-	-	-	[23] ✡	-
[1]	2019	●	●	●	●	784	-	-	-	-	T1	-	-	-	-	-	✓
[89]	2019	-	-	-	●	784	-	-	-	-	T1	✓	-	-	-	-	-
[13]	2019	◐	◐	●	●	784	-	-	-	-	T1	✓	-	-	-	-	-
[32]	2020	●	●	●	●	784	-	-	-	-	T1	✓	-	-	-	-	-
[10]	2020	-	◐	●	●	1500	-	T1	-	-	-	✓	-	-	-	-	-
[70]	2020	◐	◐	●	●	900	-	-	-	-	T1	-	-	✓	-	-	-
[55]	2020	-	◐	●	●	1500	-	T1	-	-	-	✓	-	-	-	-	-
[30]	2020	◐	◐	●	●	128	5 ★	-	-	T3	-	✓	-	-	-	-	✓
[90]	2020	●	●	●	●	784	-	-	-	-	T1	-	-	-	-	[42] ✡	
[77]	2021	-	●	●	●	1480	-	T1	-	-	-	✓	-	-	-	-	-
[56]	2021	◐	◐	●	●	784	-	-	-	-	T1	✓	-	-	-	-	-
[33]	2021	-	-	-	●	256	10 ✤	-	-	T3	-	✓	-	-	-	-	-
[31]	2021	●	●	●	●	784	-	-	-	T1	-	-	-	✓	-	[36] ✡ [75] ✡	-
[26]	2022	◐	◐	●	●	784	-	-	-	T1	-	-	-	✓	-	-	-
[50]	2022	-	-	◐	●	128	5 ★	T1	-	-	T3	✓	✓	✓	✓	[50]	-
[54]	2022	●	●	●	●	300	-	-	-	-	T1	-	-	✓	-	-	-
[57]	2023	◐	◐	●	●	3072	-	-	-	-	T1	✓	-	-	-	-	✓
[45]	2023	-	-	◐	●	128	5 ★	T1	-	-	T3	-	-	-	-	[40] ✡	-
[74]	2023	-	-	◐	●	128	-	T1	-	-	-	✓	-	-	-	-	-
[28]	2023	◐	◐	◐	●	1024	-	-	T1	-	-	✓	✓	✓	✓	[72] ✡	
[52]	2023	-	◐	●	●	300	30 ✤	-	-	-	T2	-	-	-	-	[43] ✡	
[92]	2024	-	◐	◐	●	320	5 ★	-	-	T3	-	✓	✓	✓	-	[14]	-

Raw Features Used: **L2**=Ethernet layer; **L3**=Network layer; **L4**=Transport layer; **L7**=Encrypted payload; ●=Use with SII; ◐=Use without SII;

Data Extraction Strategy: **# Bytes**=No. of bytes; **# Packets**=No. of packets; n ★=First n packets; n ✤=Any consecutive n packets;
T1=Type 1(First m bytes); **T2**=Type 2(First m bytes of n packets); **T3**=Type 3(First m bytes per packet of n Packets);

Datasets Used: **VPN'16**: ISCXVPN2016 [19]; **Tor'16**: ISCXTor2016 [27]; **TFC'16**: USTC-TFC2016 [87]; **CP'17**: Cross-Platform Application [65];
✡: Dataset collected on or before 2018

made without considering the selected traffic granularity and data extraction strategy. We categorize this tendency to overfit into three types: data leakage overfitting, contextual overfitting, and temporal overfitting.

4.2.1. Data Leakage Overfitting. This problem occurs when models inadvertently learn from features that should not be available during inference, which leads to an illusion of high performance that does not generalize to real-world scenarios. While some studies have made commendable efforts to prevent overfitting related to Strong Identification Information (SII) (see Table 1), some have not. Another issue arises from exposing Server Name Indication (SNI) [1] to ML models. As SNI is frequently used for labelling traffic in datasets [50], [51], [73], exposing it creates a potential shortcut that models can exploit.

Studies that perform classification based on the *first* m *bytes* of a *flow* or *session* granularity, where m is set to include the initial TCP and TLS handshakes frequently do not obfuscate the SNI. To capture the SNI, m must be large enough to encompass the cumulative sizes of the packets from the beginning of the TCP session up to and including the TLS Client Hello message. The TCP handshake involves three packets (SYN, SYN-ACK, ACK) with minimal payloads, typically totalling around 162 bytes (without TCP options) [20]. The TLS Client Hello message, which includes the SNI, follows the TCP handshake and varies in size but is typically around 600 bytes, depending on the number of supported cipher suites and extensions [66], [67]. Therefore, choosing an m value greater than 700 bytes generally ensures that the SNI is included in the training data. According to Table 1, m ranges from 764 to 3072 in literature, raising concerns about data leakage overfitting.

Similarly, studies that consider the *first* n *packets* of a *flow* or *session* granularity, where n includes the packets containing the TLS Client Hello message, are likely to capture the SNI. Since the TCP handshake consists of three packets and the TLS Client Hello is typically sent in the fourth packet, setting $n \geq 4$ is likely to include the SNI [66].

If the SNI is not obfuscated, models may end up using SNI values as shortcuts for classification rather than learning meaningful patterns in the underlying traffic. However, in practice, the presence of the TLS handshake and the SNI is not always guaranteed. For example, in TLS 1.3, mechanisms such as Encrypted Client Hello (ECH) encrypt the SNI to enhance privacy [68]. Further, the presence of TLS handshake is not always guaranteed due to session reestablishment [51].

1. The SNI extension in the TLS handshake reveals the hostname the client is attempting to connect to

Consequently, models that rely on the SNI may struggle with accurate classification in real-world conditions where this information is either encrypted or absent.

4.2.2. Contextual Overfitting. Contextual overfitting refers to the phenomenon where a model learns to exploit irrelevant features or context-dependent patterns present in the data that do not reflect the intrinsic nature of the target class. In the NTC domain, contextual overfitting arises when models exploit features that are artifacts of the network protocols or specific implementations rather than intrinsic properties of the traffic generated by target applications.

In contrast to data leakage overfitting, contextual overfitting impacts classifiers of studies that split a single TCP/UDP session into multiple training and testing samples. As shown in Table 1, they are (1) *packet* and *burst* granularity-based classifiers, and (2) *flow* and *session* granularity-based classifiers which extract information from *any consecutive n packets*.

For example, according to RFC 791 [69], RFC 6864 [81] and RFC 6274 [25], the IP Identification (IP ID) field in *L3* (IP header), is initialized for each session with a pseudo-random number generator and incremented sequentially for each packet transmitted. As a consequence, the high-order bits remain relatively consistent.

Similarly, the IP Header Checksum, which is a 16-bit one's complement of the one's complement sum of all 16-bit words in *L3* (IP header) [69], poses a risk for contextual overfitting. Due to the changing nature of attributes, such as the IP ID and Total Length, the Header Checksum varies with each packet. However, packets with similar Total Lengths and minor differences in IP IDs (i.e., adjacent packets) have similar checksum values within a session.

As per RFC 9293 [20], fields such as the Sequence Number and Acknowledgment Number in *L4* (TCP header) are specific to each TCP session, initialized using pseudo-random functions and incremented with data transfer. Due to gradual increments caused by uploads/downloads, the high-order digits of Sequence and Acknowledgement numbers remain mostly consistent across packets in the same TCP session.

As a consequence, models that see multiple packets from the same session may inadvertently learn patterns based on the stable high-order bits of fields like IP ID, IP header checksum, and TCP Sequence and Acknowledgment numbers, leading to overfitting. Since these features are either randomly generated per session or influenced by such random values, they should not be relied upon for classification, as they reflect session-specific artifacts rather than meaningful traffic characteristics.

4.2.3. Temporal Overfitting. Temporal overfitting occurs when models capture features that fluctuate over time due to dynamic network conditions, system configurations, or temporal changes unrelated to the application's behaviour. These features may provide superficial patterns specific to the time and environment of data collection, leading to models that do not generalize well under different conditions.

A notable example is the TCP Timestamp Option, introduced in RFC 1323 [9] to enhance TCP performance over high-speed networks. Each TCP segment can contain a 32-bit Timestamp Value (TSval) set by the sender and a 32-bit Timestamp Echo Reply (TSecr) from the remote host. The TSecr field reflects the TSval received from the sender in the previous segment, allowing for more accurate round-trip time (RTT) measurements. Since packets within a session are sent in quick succession, the most significant bits of TSval and TSecr remain consistent.

Further, TCP Window Size is a dynamic value negotiated between the sender and receiver to manage flow control and optimize data transmission. According to RFC 7323 [8] and RFC 9438 [88], the Window Size is influenced by several factors independent of the traffic class, such as the available receive buffer, network congestion levels, bandwidth-delay product (BDP), and the Path Maximum Transmission Unit (MTU). For instance, during periods of high network congestion, the Window Size may be reduced to prevent packet loss. If models rely on the TCP Window Size as a feature, they could overfit to transient network conditions upon which the dataset is collected.

Since these features fluctuate based on time-sensitive factors such as timing within a session, network congestion, and buffer availability, they should not be relied upon for classification. These elements reflect temporary network states rather than intrinsic characteristics of the traffic itself, thus limiting the model's ability to generalize across varying network environments.

Table 2 summarizes the overfitting tendencies associated with the discussed raw features. It details the types of overfitting each feature may introduce, along with the affected traffic granularities and data extraction strategies. Notably, none of the studies listed in Table 1 take steps to obfuscate or mask these session-specific attributes. As a consequence, while these features may contribute to higher performance on familiar data, models trained on them risk failing to generalize to unseen traffic where these learned patterns are absent. To validate our suppositions, we formulate the following research questions.

S2 - Research Questions

S2-RQ1: Does data leakage from SII and SNI affect the generalizability and robustness of network traffic classifiers?

S2-RQ2: Do session-specific contextual artifacts contribute to overfitting in NTC?

S2-RQ3: Do time-specific temporal artifacts contribute to overfitting in NTC?

4.3. Snag 3: Unsubstantiated Assumptions

Examining the literature reveals several unsubstantiated and often contradictory assumptions regarding design

Table 2. OVERFITTING TYPE AND AFFECTED TRAFFIC GRANULARITY

Feature	Type	Affected Granularity			
		Packet	Burst	Flow	Session
Src. MAC Addr.	DL	✓	✓	✓	✓
Dst. MAC Addr.	DL	✓	✓	✓	✓
IP ID	C	✓	✓	T2✤ T3✤	T2✤ T3✤
IP Header Checksum	C	✓	✓	T2✤ T3✤	T2✤ T3✤
Src. IP Addr.	DL	✓	✓	✓	✓
Dst. IP Addr.	DL	✓	✓	✓	✓
Src. Port	DL	✓	✓	✓	✓
Dst. Port	DL	✓	✓	✓	✓
TCP Seq. No.	C	✓	✓	T2✤ T3✤	T2✤ T3✤
TCP Ack. No.	C	✓	✓	T2✤ T3✤	T2✤ T3✤
TCP Window Size	T	✓	✓	T2✤ T3✤	T2✤ T3✤
TCP Options - TSval	T	✓	✓	T2✤ T3✤	T2✤ T3✤
TCP Options - TSerc	T	✓	✓	T2✤ T3✤	T2✤ T3✤
TLS SNI	DL	-	-	T1 T2★ T3★	T1 T2★ T3★

DL=Data Leakage; **C**=Contextual; **T**=Temporal;
★=First n Packets; ✤=Any consecutive n Packets;
T1=Type 1(First m bytes); **T2**=Type 2(First m bytes of n packets);
T3=Type 3(First m bytes per packet of n Packets);
✓=Affects regardless of the data extraction strategy.

choices.

While all studies shown in Table 1 utilize the encrypted payload (*L7*), their justifications vary.

Some studies suggest that encrypted payloads contain inherent patterns resulting from the imperfect randomness of encryption algorithms [10], [28], [33], [50], [55], [74], [92].

However, TLS 1.3 guarantees that the same plaintext will always produce different ciphertexts due to the use of Authenticated Encryption with Associated Data (AEAD) ciphers like AES-128-GCM, AES-256-GCM and ChaCha20-Poly1305, and unique initialization vectors (IVs) [66]. Given these robust security measures, the claim that machine learning models can still learn and exploit patterns directly from encrypted payloads is concerning. It suggests potential vulnerabilities in the cipher suites, implying that encrypted communications may not be as secure or random as intended.

There is also inconsistency in the perceived sufficiency of encrypted payloads for classification: several studies argue that both metadata (*L2*, *L3*, and *L4*) and encrypted payloads (*L7*) are necessary for accurate classification. In contrast, others assert that the encrypted payload alone is adequate [33], [89].

Furthermore, all studies listed in Table 1 use truncation and padding techniques to create a consistent, fixed-length embedding for training proposed ML models. However, some studies argue that altering payload lengths to fit a fixed-size representation can result in losing important information, negatively impacting the model's classification performance [78].

To address these discrepancies, we formulate the following research questions:

S3 - Research Questions

S3-RQ1: Can state-of-the-art network traffic classifiers detect meaningful patterns in encrypted payloads?

S3-RQ2: Is encrypted payload alone sufficient for accurate network traffic classification?

S3-RQ3: Does truncating or padding traffic payloads impact classification accuracy?

5. Decoding the NTC Fidelity

In the previous section, we identified research questions related to: (*Snag-1*) limitations of widely used datasets, (*Snag-2*) oversights in design choices, and (*Snag-3*) contradicting assumptions in literature. To address the *Snag-1* questions, we begin with an evaluation of widely used datasets, assessing their relevance and content in light of these concerns. Following this, we conduct 348 strategic experiments using state-of-the-art classifiers to answer the questions posed in *Snag-2* and *Snag-3*.

5.1. NTC Dataset Evaluation

In Section 4.1, we raised concerns about the credibility and applicability of network traffic classification models that rely on datasets collected before 2018. This section addresses these doubts by empirically evaluating datasets to uncover their true potential and suitability for contemporary NTC tasks.

5.1.1. Dataset Selection. To conduct meaningful analysis, we select public datasets used in more than one study presented in Table 1 or those claimed to include TLS 1.3 traffic. This selection criterion ensures that we focus on datasets with significant influence in the research community and those containing modern encryption protocols. By scrutinizing these datasets, we aim to answer research questions *S1-RQ1* and *S1-RQ2*, thereby justifying their use in developing effective NTC models.

5.1.2. Methodology. To systematically evaluate the encryption protocols and encryption algorithms present in the selected datasets, we process packet capture (PCAP) files and extract relevant encryption information using the Tshark tool[2]. A simplified pseudocode of our analysis script is presented in Algorithm 1.

We extract all unique session identifiers for each PCAP file in the dataset, which represent individual communication sessions in the network traffic *(Line 2, 3)*. These session IDs encompass TCP and UDP sessions, allowing for a comprehensive dataset analysis.

We then iterate over each session ID to determine whether the session is encrypted *(Line 4, 5)*. This task

2. Tshark: https://tshark.dev/

Algorithm 1 Dataset evaluation

```
1: Initialize pcap_stats dictionary for the dataset
2: for each pcap_file in dataset do
3:     session_ids ← Extract session IDs from pcap_file
4:     for each s_id in session_ids do
5:         type ← Check if session s_id is encrypted
6:         if type is "encrypted" then
7:             Increment pcap_stats["encrypted"] by 1
8:             c_suite ← Get cipher suite of session s_id
9:             if c_suite is "unknown" then
10:                Increment pcap_stats["unknown"] by 1
11:            else
12:                e_algo ← Get enc. algo from c_suite
13:                Increment pcap_stats[e_algo] by 1
14:            end if
15:        else
16:            Increment pcap_stats["unencrypted"] by 1
17:        end if
18:    end for
19: end for
```

is carried out by checking for the presence of encryption protocols such as TLS for TCP sessions or DTLS/QUIC for UDP sessions *(Line 6)*.

Encrypted sessions: If a session is encrypted (*type* is *encrypted*), we increment the count of encrypted sessions of the dataset being analyzed. We then attempt to identify the specific cipher suite used in the session by examining the "Server Hello" packet of the TLS handshake packets *(Line 7, 8)*. Suppose the cipher suite cannot be identified due to missing handshake packets. In that case, we increment the count of sessions with *unknown* to account for sessions with unidentified encryption methods *(Line 9, 10)*. If the cipher suite can be determined, we extract the encryption algorithm and increment the count of sessions encrypted by it *(Line 12, 13)*. This helps us understand the distribution and representation of different encryption algorithms within a dataset.

Unencrypted sessions: If a session is not encrypted (*type* is *unencrypted*), we increment the count of *unencrypted* sessions. This category includes sessions either transmitted in plaintext or unrelated to encryption protocols *(Line 16)*.

After processing all sessions within a PCAP file, we continue to the next file in the dataset *(Line 18)*. By aggregating the statistics across all PCAP files, we construct a detailed overview of encryption usage and cipher suite distribution within the dataset.

5.1.3. *S1-RQ1*: Encryption in Legacy Datasets. The results of the encryption usage analysis of the selected datasets are shown in Figure 2a. This indicates a substantial variation in the proportion of unencrypted versus encrypted traffic across different datasets. The results underscore the prevalence of unencrypted traffic in several widely used datasets, undermining their applicability in evaluating modern encrypted NTC models.

Among the datasets analyzed, the CSTNET-TLS1.3 dataset [50] stands out as it exclusively contains encrypted network traffic, as claimed.

In stark contrast, several legacy datasets reveal a significant presence of unencrypted traffic. The ISCXVPN2016 [19], and USTC-TFC2016 [87] datasets, for example, exhibit notably high percentages of unencrypted traffic, at 98.9% and 94.7%, respectively. This suggests a dominant presence of plaintext communications in these datasets, which limits their applicability in scenarios that require encrypted traffic analysis. Similarly, the ISCXTor2016 [27] dataset, which is intended to capture traffic from the Tor network, still shows 89.3% of its traffic as unencrypted. This indicates that a considerable portion of the dataset lacks the encryption expected from a privacy-focused network like Tor. The Cross-Platform Application dataset [65], which is relatively more balanced, still reveals that 69.7% of its traffic remains unencrypted.

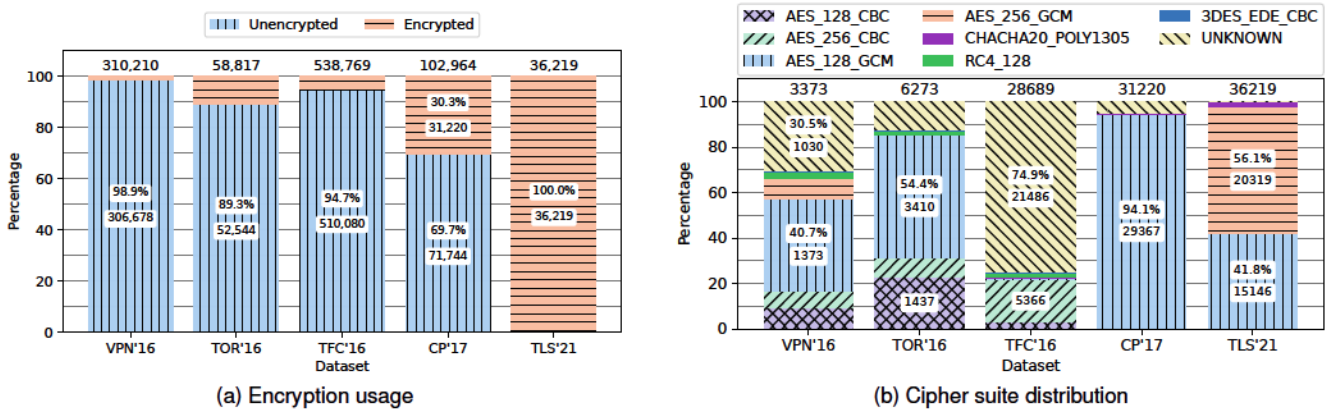
These proportions are widely misaligned with real-world conditions. According to Google's Transparency Report, as of October 2024, more than 93% of web pages loaded in Google Chrome are secured with SSL/TLS encryption [37]. This discrepancy highlights a critical gap between the encryption distributions within these datasets and the actual state of internet traffic, where encryption has become the norm.

5.1.4. *S1-RQ2*: Encryption Algorithm Usage. As shown in Figure 2b, CSTNET-TLS1.3 dataset stands out by exclusively containing traffic encrypted with current, secure AEADs, in line with the recommendations of TLS 1.3 [66]. In contrast, pre-2018 datasets like ISCXVPN2016, ISCXTor2016, and USTC-TFC2016 contain traffic sessions encrypted using outdated algorithms such as AES with CBC mode (both 128-bit and 256-bit), 3DES, and RC4, which are deprecated in contemporary security standards due to their known vulnerabilities [6], [59], [62]. This reliance on outdated encryption algorithms further limits the effectiveness of legacy datasets for developing and benchmarking classifiers.

Moreover, another notable gap is the limited representation of traffic encrypted with the ChaCha20-Poly1305 cipher suite [7]. Despite being a recommended algorithm in TLS 1.3 and the default AEAD in several widely-used secure communication protocols like OpenSSH, WireGuard, and OTRv4, none of the datasets include considerable proportions of ChaCha20-Poly1305-encrypted traffic. This lack of representation is a critical omission, considering the growing adoption of this algorithm in modern security protocols [16], [66].

***Takeaway 1:** Substantial portions of public datasets contain unencrypted traffic.*

***Takeaway 2:** Widely-used datasets include sessions encrypted by vulnerable and deprecated ciphers, potentially misleading machine learning models.*



(a) Encryption usage

(b) Cipher suite distribution

VPN'16: ISCXVPN2016; **Tor'16**: ISCXTor2016; **TFC'16**: USTC-TFC2016; **CP'17**: Cross-Platform Application; **TLS'21**: CSTNET-TLS1.3;

Each segment of the stacked bars represents a percentage of sessions. The total number of sessions relevant for the analysis is displayed on top of the corresponding stacked bar. Additionally, segments indicating the number of sessions are labeled if they constitute more than 15% of the total.

Figure 2. Public network traffic dataset evaluation.

Takeaway 3: *Although secure ciphers like ChaCha20-Poly1305 are seeing growing adoption, they remain under-represented in public datasets.*

Takeaway 4: *Given the widespread adoption of TLS 1.3 and QUIC [24], [44], public datasets should be critically evaluated and updated to reflect contemporary conditions.*

5.2. CipherSpectrum

The analysis in the previous sections highlighted several limitations in existing public datasets, particularly their unencrypted nature and usage of outdated encryption protocols. Moreover, the literature indicates that many publicly available datasets are flow features (i.e., extracted statistics) instead of raw traffic [77], [87]. While these datasets are valuable for statistics-based NTC, they are unsuitable for raw information-based or multimodal methods. As a result, many researchers are compelled to rely on self-collected or private traffic datasets, which compromises the reproducibility and credibility of their results [13], [77], [85], [94]. Further, some publicly available datasets, such as ISCXVPN2016, exhibit significant class imbalances [55], which can be detrimental to the performance of deep-learning models that are known to be sensitive to such imbalances [74].

To collectively address these issues and to support our subsequent experiments, we develop a new traffic dataset: *CipherSpectrum*. CipherSpectrum includes network traffic encrypted with modern cipher suites mandated/strongly-recommended by TLS 1.3 [66], providing a robust foundation for research in raw information-based NTC.

5.2.1. Composition. CipherSpectrum consists of encrypted TCP/UDP sessions for 40 distinct domains (classes), each represented by traffic encrypted with the three major TLS 1.3 cipher suites. Specifically, for each class and cipher suite, CipherSpectrum contains 1,000 TCP/UDP session samples, totalling 120,000 sessions (40 classes $\times$ 3 cipher suites per class $\times$ 1,000 sessions per suite). For instance, traffic for *example.com* includes 1,000 sessions with TLS-AES-128-GCM-SHA256, 1,000 with TLS-AES-256-GCM-SHA384, and 1,000 with TLS-CHACHA20-POLY1305-SHA256. To support the development of robust and generalizable NTC models and facilitate research that reflects a range of encrypted traffic scenarios, we make CipherSpectrum publicly available[3].

The data collection and development methodology for CipherSpectrum is provided in Appendix A to maintain focus on the primary discussion.

Takeaway 5: *CipherSpectrum provides a comprehensive dataset by uniformly representing traffic encrypted using the three mandated and recommended cipher suites of TLS 1.3.*

Takeaway 6: *Public availability of CipherSpectrum supports the development of robust and generalizable NTC models, facilitating cipher-agnostic network traffic classification.*

5.3. Design Choices and Assumptions Validation

In this section, we address the research questions related to (*Snag-2*) oversights in design choices and (*Snag-3*) unsubstantiated assumptions, as discussed in Section 4.2 and Section 4.3 respectively.

We conduct 348 systematic occlusion experiments to substantiate the speculative claims in which the features in question are masked, removed, or obfuscated from the input. This approach allows us to assess the impact of these features on model performance and to prove our conjectures empirically.

5.3.1. Dataset Selection. For our evaluation, we focus on encrypted traffic adhering to TLS 1.3 by utilizing two

3. CipherSpectrum: https://cspectrum.web.cse.unsw.edu.au

datasets: CipherSpectrum and CSTNET-TLS1.3 [50]. Although both models demonstrate their multi-class classification abilities, we randomly select 10 classes from each dataset. This decision is motivated by several factors: (1) preliminary tests on larger class sets confirmed the same pitfalls, reinforcing the generalizability of our findings; (2) the pitfalls we highlight are independent of the class count, making a more extensive selection unnecessary; (3) restricting the class set aligns with YaTC's proposed capabilities, which max out at 20 classes, ensuring a fair and unbiased comparison; and (4) conducting 348 occlusion experiments is computationally intensive, and limiting the number of classes allows for a thorough yet efficient analysis, enabling deeper scrutiny and reliable reproducibility.

In CipherSpectrum, the selected classes include an equal mix of traffic encrypted using AES-128-GCM, AES-256-GCM, and CHACHA20-POLY1305, ensuring a balanced representation of encryption algorithms. In CSTNET-TLS1.3, we address the class imbalance problem by randomly selecting 10 classes containing more than 400 TCP/UDP sessions. Furthermore, we randomly select 400 TCP/UDP sessions from each selected class. By adopting a dual-dataset approach, we aim to confirm the generalizability of our findings across different traffic characteristics and encryption schemes.

5.3.2. Model Selection. To substantiate our claims, we selected ET-BERT [50] and YaTC [91] for our evaluation due to their open-source nature, credibility from publication in top-ranked venues (A* by CORE), and proven superiority over traditional ML methods [63], [84], [91]. ET-BERT, regarded as state-of-the-art [28], [91], serves as a foundation for numerous studies [17], [45], [47], [53], [74], while both models effectively address a range of NTC challenges. However, our goal is not to evaluate or compare their performance as an end objective. Instead, we use these models as illustrative tools to demonstrate how design choices (see Section 2.1) influence model behaviour, leading to overfitting and other methodological pitfalls. Appendix B explains the model-specific preprocessing steps applied to represent each level of granularity and data extraction strategy for ET-BERT and YaTC.

5.3.3. Occlusion Strategies. The occlusion strategies used in our analysis are summarised in Table 3. The *A1* occlusion, which includes all data, provides a baseline performance metric. The *D1* occlusion, designed to test Strong Identification Information (SII) based data leakage overfitting (see Section 4.2.1), provides a secondary baseline. Our observations indicate that the presence of SII can overshadow other features, preventing models from learning additional information. Therefore, subsequent occlusions are compared against the performance of *D1*. In *D2*, we eliminate Server Name Indication (SNI) by replacing relevant bytes with random hex values, allowing us to examine the extent of SNI-based data leakage overfitting.

We further employ *C* and *T* occlusions to assess contextual and temporal overfitting (see Sections 4.2.2 and 4.2.3). We randomize context- and time-dependent header features in these occlusions to identify the model's reliance on these session-specific artifacts. To evaluate the combined impact of all these overfitting shortcuts, the *CTD* occlusion randomizes or removes all features associated with *D1, D2, C* and *T* occlusions.

To examine the distinct roles of header and payload in NTC, we implement *H1* and *P1* occlusions. All header information is preserved in *H1* while the payload is eradicated and padded to a uniform length with $0x00$. This ensures that the model's reliance on header features alone can be observed without interference from payload variations. Conversely, all header information is removed in *P1*, eliminating all identifying metadata from the headers. We also strip TCP options from the representation to avoid unintended impacts from variable header lengths, allowing the analysis to focus solely on the payload's contribution to classification performance.

The *E1*, *E2*, and *E3* occlusions assess whether state-of-the-art classifiers can identify patterns in the encrypted payload or if they rely solely on payload length. In *E1*, we isolate the encrypted payload by removing all plaintext information (e.g., header details) to measure its standalone impact. In *E2*, we mask the encrypted payload with $0xFF$, removing inherent randomness and leaving only payload length as a potential feature. Finally, *E3* replaces the encrypted payload with pseudo-random values independent of class labels, simulating *perfect randomness*. This configuration further helps us determine whether any discernible patterns are artifacts of payload length rather than encryption-induced imperfections. All three occlusion techniques were applied to data extraction strategies that do not depend on the *first* m *bytes* or first n packets as they do not exclusively contain encrypted payloads due to handshake packets.

As shown in Table 2, not all overfitting tendencies affect every traffic granularity. Therefore, we run occlusion experiments only on the relevant design choices. To ensure a fair and unbiased evaluation, we iteratively train and test the selected models for each occlusion separately (as opposed to training once and testing against different occluded data). Drawing on the results presented in Tables 4 and 5, Sections 5.3.4 to 5.3.10 offer an in-depth analysis of the experimental findings. However, to maintain conciseness, we report the average accuracy of each occlusion across different design choices. Specifically, for each occlusion type, the accuracy values discussed (e.g., for ET-BERT under A1 occlusion) represent the mean accuracy computed across the 12 design choices presented in Tables 4 and 5.

5.3.4. S2-RQ1: Data leakage overfitting. To address the data leakage concerns discussed in Section 4.2, we use the *A1*, *D1*, and *D2* occlusions, as outlined in Table 3. For *A1*, we conduct 48 experiments across two classifiers and 12 design choices, with results presented in Tables 4 and 5. This configuration establishes baseline performance, with ET-BERT and YaTC achieving average accuracies of 0.96 and 0.90, respectively.

Table 3. FEATURE OCCLUSION STRATEGIES

ID	Occlusion Strategy	L1	L2			L3				L7		Implications of the results
		MAC A.	IP A.	IP ID	Checksum	Ports	Seq & Ack	Window S.	Options	Payload	SNI	
$A1$	All Data	-	-	-	-	-	-	-	-	-	-	Baseline performance
$D1$	Anonymized SII	R	R	-	-	R	-	-	-	-	-	Reliance on SII
$D2$	Anonymized SNI	R	R	-	-	R	-	-	-	-	R	Reliance on SNI
C	w/o Contextual O.	R	R	R	R	R	R	-	-	-	-	Contextual overfitting proof
T	w/o Temporal O.	R	R	-	-	R	-	R	R	-	-	Temporal overfitting proof
CTD	w/o Overfitting	R	R	R	R	R	R	R	R	-	R	Absolute performance
$H1$	Header Only	R	R	-	-	R	-	-	-	E	R	Contribution of the header
$P1$	Payload Only	E	E	E	E	E	E	E	E	-	-	Contribution of the payload
$E1$	Encrypted Payload Only	E	E	E	E	E	E	E	E	ENC	E	E. payload's contribution
$E2$	$E1$ - Masked	E	E	E	E	E	E	E	E	MSK	E	E. payload's length's contribution
$E3$	$E1$ - Obfuscated	E	E	E	E	E	E	E	E	OBF	E	Implicit pattern's contribution

L2=Ethernet layer; **L3**=Network layer; **L4**=Transport layer; **L7**=Encrypted payload; (Only features in question are shown in the table)

A.=Address; **Ports**=Source and destination ports; **Seq & Ack**=Sequence & Acknowledgment numbers; **Window S.**=Window size;

O.=Overfitting; **E**=Eradicate (replace relevant bytes with $0x00$); **R**=Randomize (replace relevant bytes with $0xnn$, where n is a random hexadecimal value);

ENC=Encrypted; **MSK**=Replace encrypted bytes with $0xFF$; **OBF**=Randomize encrypted bytes with $0xnn$, where n is a random hexadecimal value;

Impact of SII: To examine the impact of Strong Identification Information (SII), we repeat the 24 experiments for each classifier using the *D1* occlusion. On average, ET-BERT achieved an accuracy of 0.51 (↓ 0.45), while YaTC reached 0.62 (↓ 0.28). The average accuracy drop of 0.36 highlights the influence of SII on classifier performance, exposing the risk of overfitting. Besides the resultant poor generalizability, obfuscation practices, such as MAC address randomization and dynamic IP allocation, reduce the reliability of SII, as these features change frequently or can be intentionally altered to protect privacy [58], [93].

We commend the studies shown in Table 1 that avoid using SII for classification, emphasizing that MAC addresses, IP addresses, and protocol ports should not be relied upon as features for classification.

Impact of SNI: The *D2* occlusion is applied to design choices that rely on the initial portion of a TCP/UDP session. Specifically, this includes data extraction strategies that focus on the *First m bytes*, *First m bytes of n packets*, and *First m bytes per packet of n packets*, as shown in Tables 4 and 5. When using the *D1* occlusion on these strategies, ET-BERT and YaTC achieved average accuracies of 0.17 and 0.44, respectively. Under *D2*, ET-BERT's accuracy remained largely unchanged at 0.16 (↓ 0.01), while YaTC's accuracy decreased to 0.38 (↓ 0.06). This difference is potentially due to the data representation sizes each classifier uses. ET-BERT represents *T1* and *T2* granularities with 640 bytes and *T3* with 128 bytes per packet across 5 packets, which is unlikely to capture the SNI. In contrast, YaTC uses 1600 bytes for *T1* and *T2* granularities and 320 bytes per packet for 5 packets in *T3*, making it more likely to include the SNI.

The influence of SNI on classification accuracy is minimal in our experiments due to its inconsistent capture across different TLS implementations. The position of the SNI extension within the ClientHello message varies, appearing earlier in Firefox and later in Chromium. Since data extraction strategies operate on fixed-length byte segments, the SNI may or may not be included, showcasing minimal overall impact on accuracy. However, this positional variance highlights the importance of considering implementation-specific factors, as unintended SNI exposure can still introduce bias. As hypothesized in Section 4.2.1, classifier accuracy is affected by SNI data leakage, though the extent of this impact depends on design choices and data representation sizes.

***Guideline 1**: Avoid using Strong Identification Information (SII) features such as MAC addresses, IP addresses, and protocol ports, as they contribute to overfitting and reduce model generalizability.*

***Guideline 2**: Obfuscate or exclude SNI data in initial portions of sessions to prevent data leakage overfitting, mainly when using large data representations that may capture the SNI unintentionally.*

5.3.5. S2-RQ2: Contextual overfitting. To assess the presence of contextual overfitting, we implemented the *C* occlusion, which randomizes session-specific fields such as the IP Identification (IP ID), IP header checksum, and TCP sequence and acknowledgment numbers.

As discussed in Section 4.2.2, contextual overfitting occurs when a single TCP/UDP session is split into multiple samples used in training and testing sets. To demonstrate, we focused on packet and burst-based granularities and employed data extraction strategies that select *any consecutive n packets*, as these are more susceptible to contextual overfitting.

Our experimental results, summarized in Tables 4 and 5, show that under the *C* occlusion, the average accuracies of ET-BERT and YaTC decreased to 0.57 (↓ 0.06) and 0.55 (↓ 0.05), respectively. This is a noticeable drop from their average accuracies of 0.63 and 0.60 under the baseline

Table 4. FEATURE OCCLUSION RESULTS - CLASSIFICATION ACCURACIES AGAINST CIPHERSPECTRUM DATASET

ID	Model	Packet	Burst	Flow					Session				
				T1	T2 ★	T2 ✤	T3 ★	T3 ✤	T1	T2 ★	T2 ✤	T3 ★	T3 ✤
A1	ET-BERT	0.99	0.98	0.96	0.96	0.97	0.90	0.98	0.93	0.94	0.96	0.95	0.99
	YaTC	0.94	0.90	0.84	0.84	0.80	0.87	0.89	0.86	0.89	0.80	0.90	0.91
D1	ET-BERT	0.79	0.41	0.10	0.12	0.31	0.08	0.29	0.11	0.10	0.23	0.57	0.79
	YaTC	0.42	0.74	0.51	0.49	0.36	0.54	0.40	0.36	0.36	0.31	0.42	0.41
D2	ET-BERT	-	-	0.08	0.10	-	0.08	-	0.10	0.09	-	0.52	-
	YaTC	-	-	0.43	0.38	-	0.50	-	0.34	0.29	-	0.37	-
C	ET-BERT	0.68	0.35	-	-	0.29	-	0.20	-	-	0.18	-	0.68
	YaTC	0.37	0.69	-	-	0.35	-	0.33	-	-	0.30	-	0.37
T	ET-BERT	0.76	0.38	-	-	0.30	-	0.23	-	-	0.21	-	0.71
	YaTC	0.40	0.73	-	-	0.36	-	0.38	-	-	0.29	-	0.39
CTD	ET-BERT	0.62	0.33	-	-	0.26	-	0.19	-	-	0.17	-	0.63
	YaTC	0.33	0.69	-	-	0.35	-	0.32	-	-	0.28	-	0.33
H1	ET-BERT	0.80	0.37	0.10	0.09	0.29	0.10	0.28	0.10	0.12	0.25	0.58	0.79
	YaTC	0.44	0.72	0.27	0.27	0.22	0.48	0.28	0.11	0.26	0.23	0.30	0.36
E1	ET-BERT	0.12	0.12	-	-	0.14	-	0.13	-	-	0.11	-	0.11
	YaTC	0.30	0.31	-	-	0.29	-	0.26	-	-	0.28	-	0.25
E2	ET-BERT	0.12	0.13	-	-	0.14	-	0.13	-	-	0.11	-	0.11
	YaTC	0.36	0.39	-	-	0.42	-	0.37	-	-	0.39	-	0.33
E3	ET-BERT	0.11	0.12	-	-	0.14	-	0.13	-	-	0.10	-	0.11
	YaTC	0.31	0.30	-	-	0.31	-	0.27	-	-	0.28	-	0.26
$\hat{E}2$	ET-BERT	0.12	0.12	-	-	0.11	-	0.12	-	-	0.13	-	0.12
	YaTC	0.34	0.32	-	-	0.37	-	0.33	-	-	0.35	-	0.3
$\hat{\hat{E}}2$	ET-BERT	0.12	0.11	-	-	0.12	-	0.11	-	-	0.13	-	0.11
	YaTC	0.31	0.29	-	-	0.34	-	0.3	-	-	0.29	-	0.27

Table 5. FEATURE OCCLUSION RESULTS - CLASSIFICATION ACCURACIES AGAINST CSTNET-TLS1.3 DATASET

ID	Model	Packet	Burst	Flow					Session				
				T1	T2 ★	T2 ✤	T3 ★	T3 ✤	T1	T2 ★	T2 ✤	T3 ★	T3 ✤
A1	ET-BERT	0.99	0.98	0.95	0.96	0.99	0.95	0.99	0.93	0.92	0.99	0.92	0.98
	YaTC	0.98	0.83	0.89	0.90	0.93	0.94	0.96	0.95	0.95	0.93	0.96	0.97
D1	ET-BERT	0.97	0.74	0.50	0.53	0.79	0.57	0.79	0.66	0.69	0.76	0.71	0.73
	YaTC	0.90	0.61	0.81	0.79	0.70	0.80	0.81	0.92	0.92	0.68	0.95	0.86
C	ET-BERT	0.96	0.75	-	-	0.72	-	0.68	-	-	0.74	-	0.71
	YaTC	0.86	0.58	-	-	0.66	-	0.75	-	-	0.63	-	0.81
T	ET-BERT	0.94	0.68	-	-	0.64	-	0.63	-	-	0.70	-	0.68
	YaTC	0.86	0.58	-	-	0.61	-	0.75	-	-	0.62	-	0.83
CTD	ET-BERT	0.83	0.67	-	-	0.49	-	0.51	-	-	0.61	-	0.56
	YaTC	0.69	0.52	-	-	0.51	-	0.63	-	-	0.45	-	0.73
H1	ET-BERT	0.96	0.77	0.68	0.69	0.78	0.58	0.79	0.81	0.79	0.78	0.72	0.77
	YaTC	0.90	0.62	0.68	0.68	0.73	0.69	0.80	0.80	0.79	0.71	0.86	0.85
E1	ET-BERT	0.13	0.13	-	-	0.11	-	0.11	-	-	0.11	-	0.09
	YaTC	0.38	0.34	-	-	0.35	-	0.26	-	-	0.37	-	0.22
E2	ET-BERT	0.13	0.14	-	-	0.12	-	0.11	-	-	0.11	-	0.10
	YaTC	0.46	0.40	-	-	0.47	-	0.33	-	-	0.50	-	0.34
E3	ET-BERT	0.13	0.14	-	-	0.12	-	0.11	-	-	0.11	-	0.10
	YaTC	0.38	0.33	-	-	0.35	-	0.27	-	-	0.37	-	0.22
$\hat{E}2$	ET-BERT	0.11	0.13	-	-	0.12	-	0.12	-	-	0.13	-	0.11
	YaTC	0.24	0.39	-	-	0.47	-	0.31	-	-	0.48	-	0.33
$\hat{\hat{E}}2$	ET-BERT	0.12	0.14	-	-	0.13	-	0.11	-	-	0.12	-	0.13
	YaTC	0.22	0.35	-	-	0.37	-	0.28	-	-	0.39	-	0.31

★=First n Packets; ✤=Any consecutive n Packets;
T1=Type 1(First m bytes); **T2**=Type 2(First m bytes of n packets); **T3**=Type 3(First m bytes per packet of n Packets);

condition *D1*.

The consistent decline in accuracy across all evaluated design choices when applying the *C* occlusion indicates that the models rely on session-specific contextual features as shortcuts for classification.

This reduction in performance highlights the importance of mitigating context-dependent, session-specific artifacts in the dataset.

***Guideline 3**: Avoid uninformative, session-specific fields such as IP ID, IP header checksum, and TCP sequence/acknowledgment numbers to reduce contextual overfitting risks.*

5.3.6. S2-RQ3: Temporal overfitting. We applied the *T* occlusion strategy to assess temporal overfitting, which involves randomizing time-variant fields such as the TCP option timestamps.

Mirroring our approach for contextual overfitting, we conducted 12 experiments for each model, as reflected in Tables 4 and 5. Consistent with the previous findings, both ET-BERT and YaTC showed a decrease in average accuracy under the *T* occlusion. Their accuracies dropped to 0.57 ($\downarrow$ 0.06) and 0.56 ($\downarrow$ 0.04), respectively, compared to their baseline accuracies of 0.63 and 0.60.

This decline in performance indicates that the models were leveraging time-dependent features as shortcuts for classification, confirming the presence of temporal overfitting.

***Guideline 4**: Exclude or randomize time-variant fields, such as TCP timestamps, to prevent models from developing dependencies on temporal patterns.*

5.3.7. Consequences of Design Choice Oversights. To further illustrate the impact of design choice oversights, we evaluated ET-BERT and YaTC using their originally proposed data preprocessing and representation techniques (as opposed to preprocessing methods discussed in Appendix B). Under these original conditions, ET-BERT and YaTC achieved high average accuracies of 0.91 and 0.93 across the two datasets. However, when the features associated with data leakage, contextual overfitting, and temporal overfitting were randomized, the accuracy of ET-BERT dropped sharply to 0.59 ($\downarrow$ 0.32), while YaTC's accuracy fell to 0.68 ($\downarrow$ 0.25).

These significant accuracy declines underscore the extent to which both models relied on features prone to data leakage and overfitting. This highlights the need for careful design choices to avoid reliance on artifacts that compromise the model's generalizability and robustness.

***Guideline 5**: Ensure data extraction strategies minimize reliance on contextual and temporal features by avoiding overlap between training and testing samples drawn from the same sessions.*

5.3.8. S3-RQ1: Patterns in Cipher Texts. In this section, we investigate the existence of discernible patterns in encrypted payloads, some of which claim to be due to imperfect randomness in encryption ciphers. ET-BERT and YaTC have explicitly suggested their ability to exploit patterns in encrypted payloads, serving as ideal candidates for this analysis.

First, we perform the *E1* analysis, as outlined in Table 3. In this configuration, we use only the encrypted payload, removing any plaintext information (e.g., header details) to isolate its contribution to classification. As shown in Tables 4 and 5, we conducted 12 experiments per classifier and on average, ET-BERT achieved an accuracy of 0.12, while YaTC achieved an accuracy of 0.30.

Next, we apply a masking strategy in the *E2* occlusion configuration, replacing the encrypted payload bytes with $0xFF$. This test addresses the hypothesis of *imperfect randomness* by removing any underlying randomness in the payload, leaving only the encrypted payload length as a potential feature for the model. Under these conditions, ET-BERT maintained an accuracy of 0.12, while YaTC's accuracy increased to 0.39 ($\uparrow$ 0.09).

The increase in classifier accuracy under the *E2* configuration can be attributed to differences in data representation and the impact of masking, which removes noise introduced by encryption. ET-BERT processes network traffic as a token sequence without explicit packet boundaries, limiting its ability to leverage structural information. In contrast, YaTC's Matrix Flow Representation organizes traffic into structured segments, preserving per-packet boundaries. This structured approach allows YaTC to better exploit class-specific patterns in payload lengths that remain observable in TLS 1.3 despite encryption [66]. By eliminating encryption-induced randomness, the *E2* masking strategy enhances YaTC's ability to recognize these patterns, leading to improved classification accuracy.

Finally, in the *E3* occlusion configuration, we replace the encrypted payload bytes with pseudo-random values independent of the class labels, simulating a scenario of *perfect randomness*. This approach aims to eliminate any patterns that could arise from encryption, further testing whether observable patterns stem from payload length rather than cipher-related randomness. In this setup, ET-BERT again averaged an accuracy of 0.12, and YaTC maintained an accuracy of 0.30.

Collectively, the 72 experiments presented in Tables 4 and 5 demonstrate that state-of-the-art classifiers do not learn any intrinsic patterns from encrypted payloads beyond their length. This finding aligns with the guarantees provided by TLS 1.3, which asserts that the only observable characteristic in encrypted payloads is their length [66]. Thus, our results reinforce that any previously perceived patterns within encrypted payloads are likely artifacts from outdated datasets containing unencrypted data.

***Guideline 6**: Focus on encrypted payload length rather than content, as classifiers primarily rely on payload length for classification rather than intrinsic patterns in the cipher text.*

***Guideline 7**: Use structured representations (e.g., segmented matrices) to better capture payload length variations, especially when packet boundaries are relevant to the model.*

Guideline 8: *Be cautious in interpreting patterns observed in encrypted data, as they may result from unencrypted or outdated datasets rather than meaningful insights in modern encrypted traffic.*

5.3.9. S3-RQ2: Sufficiency of Encrypted Payload Alone. To evaluate whether encrypted payload alone is sufficient for classification, we applied the *D1*, *H1*, and *E1* occlusions, as outlined in Table 3. For reference, ET-BERT and YaTC achieved baseline accuracies of 0.63 and 0.60 under the *D1* condition. When isolated to encrypted payload only (*E1* occlusion), ET-BERT's accuracy dropped significantly to 0.12 ($\downarrow$ 0.51), and YaTC's to 0.30 ($\downarrow$ 0.30), suggesting that encrypted payload alone provides limited information for accurate classification.

Next, we isolated header information using the *H1* occlusion. ET-BERT matched its baseline *D1* accuracy of 0.63, indicating that its classification is entirely based on header information. In contrast, YaTC's accuracy under *H1* dropped slightly to 0.57 ($\downarrow$ 0.03), showing that it relies on both header and payload information for classification, though header information plays a dominant role.

These results suggest that claims regarding the sufficiency of encrypted payload alone for classification may stem from artifacts present in unencrypted network traffic datasets. While encrypted payload lengths contribute when appropriately represented, they are insufficient for accurate classification in contemporary networks where headers remain essential.

Guideline 9: *Avoid relying solely on encrypted payloads for classification, as they provide limited information; header data remains crucial for accuracy.*

Guideline 10: *Be cautious of assumptions about payload sufficiency, as these may originate from artifacts in outdated or unencrypted datasets.*

Guideline 11: *Leverage both header and payload length information to improve classification performance, especially in modern, encrypted network environments.*

5.3.10. S3-RQ3: Truncating, padding vs performance. To further examine the role of payload length in classification, we tested the impact of truncating and padding the payload. Recognizing that manipulating the only learnable characteristic—payload length—can reduce performance, we implemented two occlusion strategies: $\hat{E}2$ and $\hat{\hat{E}}2$.

In $\hat{E}2$, we truncate the payload by 25%, representing the traffic with only $0.75m$ bytes (where m is the original byte length used for the granularity). Similarly, in $\hat{\hat{E}}2$, we truncate the payload by 50%, using $0.5m$ bytes to represent traffic. For YaTC, additional padding of 25% and 50% is applied, respectively, to compensate for the reduced payload length. However, this padding adjustment does not impact ET-BERT due to its token sequence-based representation.

Compared to the baseline *E2*, YaTC's average accuracy decreased from 0.39 to 0.35 ($\downarrow$ 0.04) and 0.31 ($\downarrow$ 0.08) for $\hat{E}2$ and $\hat{\hat{E}}2$, highlighting its dependency on payload length for classification. In contrast, ET-BERT's average accuracy remained unchanged, further demonstrating that its linguistic representation does not leverage the encrypted payloads' length for classification.

As RFC 8446 states, "*TLS does not hide the length of the data it transmits, though endpoints can pad TLS records to obscure lengths.*" [66]. These experiments highlight the effects of truncation and padding on classification performance, reinforcing the significance of payload length as a critical feature in NTC.

Guideline 12: *Avoid arbitrary truncation or padding of payloads, as these modifications can significantly impact classifier accuracy. Especially for models that rely on payload length as a key feature.*

6. Discussion

This study systematically examined the current landscape of raw-information-based network traffic classification (NTC) identifying key challenges that impact its effectiveness. Our findings highlight the importance of prioritizing updated datasets that reflect modern encryption protocols. Similarly, we emphasize that design choices and features should be selected based on a thorough evaluation of RFCs (Request for Comments) to assess suitability and informativeness. Future work could broaden the scope of analysis to explore a broader range of models and datasets. Furthermore, we highlight the need for greater transparency and reproducibility in developing ML models for NTC.

Limited Scope: Researchers, particularly within the network security community, have emphasized the critical need for improved reproducibility [5], [38], a call to action we support. As discussed in Section 5.3.2, the scope of our analysis was confined to two fully reproducible models considered state-of-the-art and publicly available datasets. We emphasize that the focus of our analysis was not to evaluate the performance of classifiers but rather to uncover potential pitfalls in design choices. To achieve this, we focused on state-of-the-art models and ensured they represented a diverse range of design choices, enabling a representation of the challenges in the field. Although this may not capture the entirety of practices in the field, our reported findings expose pitfalls that are symptomatic of the broader literature. To maintain transparency and foster further research, we make all scripts related to our evaluation publicly available[4]. In this context, our work contributes to ongoing efforts that advocate for a more critical evaluation of developments in this area [3], [4], [38].

Validity of CipherSpectrum: Synthetic data collected using automated scripts without human involvement has faced scrutiny regarding its credibility and representativeness. However, CipherSpectrum addresses the immediate need for a contemporary dataset while remaining valid for evaluating raw-information-based NTC models designed for TLS 1.3 encrypted traffic. This validity stems from the dataset's inclusion of accurate header and payload information, verified by ensuring full rendering of web pages

4. https://github.com/nime-sha256/ntc-enigma

during data collection. In future work, we plan to extend CipherSpectrum by incorporating data collected with human involvement, enhancing its fidelity further. Additionally, we make our instrumented Chromium version publicly available to support and facilitate research in this domain[5].

7. Conclusion

Our systematization of knowledge uncovered drawbacks in raw information-based network traffic classification (NTC). We identified a widespread reliance on outdated datasets, oversights in design choices leading to overfitting, and the consequences of unsubstantiated assumptions.

Specifically, we demonstrated that popularly used datasets include substantial amounts of unencrypted network traffic and do not reflect contemporary security protocols and standards (e.g., TLS 1.3). To address this issue, we introduced CipherSpectrum, a contemporary dataset that embodies modern encryption practices, enabling the development of robust and generalizable NTC models.

Through 348 feature occlusion experiments on state-of-the-art classifiers, we achieved two main objectives: (1) we demonstrated how design oversights can lead to overfitting, hindering the classifiers' generalizability; (2) we validated/refuted prevailing assumptions by providing empirical evidence, thereby reducing confusion and inconsistency in the field.

Building upon our findings, we provided strategic insights and best practices to mitigate the identified issues.

In conclusion, our work underscores the necessity for updated datasets and careful design choices to make NTC more robust and applicable. Further, by reevaluating prevailing assumptions, we paved the way for more effective network traffic classification in today's encrypted landscape.

Acknowledgments

We sincerely thank the reviewers and our shepherd for their constructive feedback, which improved the quality of this work. We also extend our gratitude to Distinguished Professor Yvo Desmedt, Associate Professor Gustavo Batista, and Dr Mohammad Goudarzi for their valuable insights throughout this research.

5. https://github.com/nime-sha256/chromium-cipher-suite-customizer

References

[1] G. Aceto, D. Ciuonzo, *et al.*, "Mobile encrypted traffic classification using deep learning: Experimental evaluation, lessons learned, and challenges," *IEEE Transactions on Network and Service Management*, 2019.

[2] G. Aceto, D. Ciuonzo *et al.*, "Mimetic: Mobile encrypted traffic classification using multimodal deep learning," *Computer Networks*, 2019.

[3] G. Apruzzese, L. Pajola *et al.*, "The Cross-evaluation of Machine Learning-based Network Intrusion Detection Systems," *IEEE Transactions on Network and Service Management (IEEE TNSM)*, 2022.

[4] D. Arp, E. Quiring *et al.*, "Dos and don'ts of machine learning in computer security," in *31st USENIX Security Symposium (USENIX Security 22)*, 2022.

[5] V. Bajpai, A. Brunstrom *et al.*, "The dagstuhl beginners guide to reproducibility for experimental networking research," *SIGCOMM Comput. Commun. Rev.*, 2019.

[6] E. Barker and A. Roginsky, "Transitioning the use of cryptographic algorithms and key lengths," National Institute of Standards and Technology, Tech. Rep., 2018.

[7] D. Bernstein, "Chacha, a variant of salsa20," *cr.yp.to*, 2008.

[8] D. Borman, R. T. Braden *et al.*, "TCP Extensions for High Performance," RFC 7323, 2014.

[9] D. A. Borman, R. T. Braden *et al.*, "TCP Extensions for High Performance," RFC 1323, 1992.

[10] Z. Bu, B. Zhou, *et al.*, "Encrypted network traffic classification using deep and parallel network-in-network models," *IEEE Access*, 2020.

[11] Chromium, "Chromium — chromium.org."

[12] CloudFlare, "Goodbye, Alexa. Hello, Cloudflare Radar Domain Rankings — blog.cloudflare.com."

[13] S. Cui, B. Jiang *et al.*, "A session-packets-based encrypted traffic classification using capsule neural networks," in *2019 IEEE 21st International Conference on High Performance Computing and Communications; IEEE 17th International Conference on Smart City; IEEE 5th International Conference on Data Science and Systems (HPCC/SmartCity/DSS)*, 2019.

[14] S. Dadkhah, H. Mahdikhani *et al.*, "Towards the development of a realistic multidimensional iot profiling dataset," in *2022 19th Annual International Conference on Privacy, Security & Trust (PST)*, 2022.

[15] J. Dai, X. Xu *et al.*, "Glads: A global-local attention data selection model for multimodal multitask encrypted traffic classification of iot," *Computer Networks*, 2023.

[16] J. P. Degabriele, J. Govinden *et al.*, "The security of chacha20-poly1305 in the multi-user setting," in *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, 2021.

[17] Q. Ding, Z. Zha *et al.*, "Multi-granularity feature fusion for enhancing encrypted traffic classification," *International Journal of Advanced Computer Science and Applications*, 2024.

[18] P. Dodia, M. AlSabah *et al.*, "Exposing the rat in the tunnel: Using traffic analysis for tor-based malware detection," in *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, 2022.

[19] G. Draper-Gil, A. H. Lashkari *et al.*, "Characterization of encrypted and vpn traffic using time-related features," in *International Conference on Information Systems Security and Privacy*, 2016.

[20] W. Eddy, "Transmission Control Protocol (TCP)," 2022.

[21] R. T. El-Maghraby, *et al.*, "A survey on deep packet inspection," in *2017 12th International Conference on Computer Engineering and Systems (ICCES)*, 2017.

[22] Firefox, "Firefox — mozilla.org."

[23] S. García, M. Grill *et al.*, "An empirical comparison of botnet detection methods," *Computers & Security*, 2014.

[24] K. Y. Gbur and F. Tschorsch, "Quicforge: Client-side request forgery in QUIC," in *30th Annual Network and Distributed System Security Symposium, NDSS*, 2023.

[25] F. Gont, "Security Assessment of the Internet Protocol Version 4," RFC 6274, 2011.

[26] J. Guo, M. Cui, *et al.*, "Global-aware prototypical network for few-shot encrypted traffic classification," in *IFIP Networking Conference (IFIP Networking)*, 2022.

[27] A. Habibi Lashkari., G. Draper Gil *et al.*, "Characterization of tor traffic using time based features," in *Proceedings of the 3rd International Conference on Information Systems Security and Privacy - ICISSP*. INSTICC, 2017.

[28] Z. Hang, Y. Lu *et al.*, "Flow-mae: Leveraging masked autoencoder for accurate, efficient and robust malicious traffic classification," in *Proceedings of the 26th International Symposium on Research in Attacks, Intrusions and Defenses*, 2023.

[29] J. Hayes and G. Danezis, "k-fingerprinting: A robust scalable website fingerprinting technique," in *25th USENIX Security Symposium (USENIX Security 16)*, 2016.

[30] H. Y. He, G. Yang *et al.*, "Pert: Payload encoding representation from transformer for encrypted traffic classification," in *2020 ITU Kaleidoscope: Industry-Driven Digital Transformation (ITU K)*, 2020.

[31] M. He, X. Wang *et al.*, "Deep-feature-based autoencoder network for few-shot malicious traffic detection," *Security and Communication Networks*, vol. 2021, 2021.

[32] Y. He and W. Li, "Image-based encrypted traffic classification with convolution neural networks," in *IEEE Fifth International Conference on Data Science in Cyberspace (DSC)*, 2020.

[33] X. Hu, C. Gu *et al.*, "Cbd: A deep-learning-based scheme for encrypted traffic classification with a general pre-training method," *Sensors*, 2021.

[34] G. Huang, C. Ma *et al.*, "Efficient and low overhead website fingerprinting attacks and defenses based on tcp/ip traffic," in *Proceedings of the ACM Web Conference 2023*, 2023.

[35] H. Huang, H. Deng *et al.*, "Automatic multi-task learning system for abnormal network traffic detection," *International Journal of Emerging Technologies in Learning (iJET)*, 2018.

[36] S. Iman, L. Arash Habibi *et al.*, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," in *International Conference on Information Systems Security and Privacy*, 2018.

[37] G. (Inc), "Google Transparency Report — transparencyreport.google.com."

[38] A. S. Jacobs, R. Beltiukov *et al.*, "Ai/ml for network security: The emperor has no clothes," in *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, 2022.

[39] M. Juarez, S. Afroz *et al.*, "A critical evaluation of website fingerprinting attacks," in *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, 2014.

[40] N. Koroniotis, N. Moustafa *et al.*, "Towards the development of realistic botnet dataset in the internet of things for network forensic analytics: Bot-iot dataset," *Future Gener. Comput. Syst.*, 2019.

[41] E. Kovacs, "Amazon's Shuttering of Alexa Ranking Service Hits Cybersecurity Industry — securityweek.com."

[42] S. Lab, "Malware Capture Facility Project: Mixed Captures — Stratosphere IPS — stratosphereips.org," https://www.stratosphereips.org/datasets-mixed, [Accessed 25-10-2024].

[43] A. H. Lashkari, A. F. A. Kadir *et al.*, "Toward developing a systematic approach to generate benchmark android malware datasets and classification," in *2018 International Carnahan Conference on Security Technology (ICCST)*, 2018.

[44] H. Lee, D. Kim, and Y. Kwon, "Tls 1.3 in practice:how tls 1.3 contributes to the internet," in *Proceedings of the Web Conference 2021*, 2021.

[45] S. Lei, X. Zhang *et al.*, "Rp-bert: An approach to detect and classify network intrusions based on a combination of transfer learning and rules," *Journal of Physics: Conference Series*, 2023.

[46] J. Li, S. Wu *et al.*, "Packet-level open-world app fingerprinting on wireless traffic," in *29th Annual Network and Distributed System Security Symposium, NDSS*. The Internet Society, 2022.

[47] Y. Liang, P. Li *et al.*, "Em-bert: A language model based method to detect encrypted malicious network traffic," in *Proceedings of International Conference on Image, Vision and Intelligent Systems 2023 (ICIVIS 2023)*, 2024.

[48] K. Lin, X. Xu *et al.*, "Mffusion: A multi-level features fusion model for malicious traffic detection based on deep learning," *Computer Networks*, 2022.

[49] P. Lin, K. Ye *et al.*, "A novel multimodal deep learning framework for encrypted traffic classification," *IEEE/ACM Transactions on Networking*, 2023.

[50] X. Lin, G. Xiong *et al.*, "Et-bert: A contextualized datagram representation with pre-training transformers for encrypted traffic classification," in *Proceedings of the ACM Web Conference 2022*, 2022.

[51] C. Liu, Z. Cao, Xiong *et al.*, "Mampf: Encrypted traffic classification based on multi-attribute markov probability fingerprints," in *2018 IEEE/ACM 26th International Symposium on Quality of Service (IWQoS)*, 2018.

[52] J. Liu, L. Wang *et al.*, "Spatial-temporal feature with dual-attention mechanism for encrypted malicious traffic detection," *Security and Communication Networks*, 2023.

[53] K. Liu, Y. Zhang, and S. Lu, "Stc-bert (satellite traffic classification-bert): A traffic classification model for low-earth-orbit satellite internet systems," *Electronics*, 2024.

[54] X. Liu, M. Shen *et al.*, "Fewfine: Few-shot malware traffic classification via transfer learning based on fine-tuning strategy," in *2022 IEEE Smartworld, Ubiquitous Intelligence & Computing, Scalable Computing & Communications, Digital Twin, Privacy Computing, Metaverse, Autonomous & Trusted Vehicles*, 2022.

[55] M. Lotfollahi, M. Jafari Siavoshani *et al.*, "Deep packet: a novel approach for encrypted traffic classification using deep learning," *Soft Computing*, 2020.

[56] B. Lu, N. Luktarhan *et al.*, "ICLSTM: Encrypted traffic service identification based on inception-lstm neural network," *Symmetry*, 2021.

[57] X. Ma, W. Zhu *et al.*, "Eetc: An extended encrypted traffic classification algorithm based on variant resnet network," *Computers & Security*, 2023.

[58] R. Meier, D. Gugelmann *et al.*, "itap: In-network traffic analysis prevention using software-defined networks," in *Proceedings of the Symposium on SDN Research*, 2017.

[59] K. Moriarty and S. Farrell, "Deprecating TLS 1.0 and TLS 1.1," RFC 8996, 2021.

[60] S. Oh, M. Lee *et al.*, "Appsniffer: Towards robust mobile app fingerprinting against vpn," in *Proceedings of the ACM Web Conference 2023*, 2023.

[61] W. Peng, L. Cui *et al.*, "Bottom aggregating, top separating: An aggregator and separator network for encrypted traffic understanding," *Trans. Info. For. Sec.*, 2025.

[62] A. Popov, "Prohibiting RC4 Cipher Suites," RFC 7465, 2015.

[63] C. Qian, X. Li *et al.*, " NetBench: A Large-Scale and Comprehensive Network Traffic Benchmark Dataset for Foundation Models ," in *2024 IEEE International Workshop on Foundation Models for Cyber-Physical Systems & Internet of Things (FMSys)*, 2024.

[64] J. Qu, X. Ma *et al.*, "An Input-Agnostic hierarchical deep learning framework for traffic fingerprinting," in *32nd USENIX Security Symposium (USENIX Security 23)*, 2023.

[65] J. Ren, D. Dubois *et al.*, "An international view of privacy risks for mobile apps," 2019.

[66] E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.3," RFC 8446, 2018.

[67] E. Rescorla and T. Dierks, "The Transport Layer Security (TLS) Protocol Version 1.2," RFC 5246, 2008.

[68] E. Rescorla, K. Oku *et al.*, "TLS Encrypted Client Hello," Internet Engineering Task Force, 2024.

[69] "Internet Protocol," RFC 791, 1981.

[70] C. Rong, G. Gou *et al.*, "Transnet: Unseen malware variants detection using deep transfer learning," in *Security and Privacy in Communication Networks*, 2022.

[71] Selenium, "WebDriver — selenium.dev."

[72] I. Sharafaldin, A. H. Lashkari *et al.*, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," in *International Conference on Information Systems Security and Privacy*, 2018.

[73] M. Shen, Z. Gao *et al.*, "Efficient fine-grained website fingerprinting via encrypted traffic analysis with deep learning," in *2021 IEEE/ACM 29th International Symposium on Quality of Service (IWQOS)*, 2021.

[74] Z. Shi, N. Luktarhan *et al.*, "Bfcn: A novel classification method of encrypted traffic based on bert and cnn," *Electronics*, 2023.

[75] A. Shiravi, H. Shiravi *et al.*, "Toward developing a systematic approach to generate benchmark datasets for intrusion detection," *Computers & Security*, 2012.

[76] P. Sirinam, M. Imani *et al.*, "Deep fingerprinting: Undermining website fingerprinting defenses with deep learning," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, 2018.

[77] S. Soleymanpour, H. Sadr *et al.*, "Cscnn: Cost-sensitive convolutional neural network for encrypted traffic classification," *Neural Process. Lett.*, 2021.

[78] Z. Tang, J. Wang *et al.*, "Markov-gan: Markov image enhancement method for malicious encrypted traffic classification," *IET Information Security*, 2022.

[79] V. F. Taylor, R. Spolaor *et al.*, "Robust smartphone app identification via encrypted network traffic analysis," *IEEE Transactions on Information Forensics and Security*, 2018.

[80] V. Tong, H. A. Tran *et al.*, "A novel quic traffic classifier based on convolutional neural networks," in *2018 IEEE Global Communications Conference (GLOBECOM)*, 2018.

[81] D. J. D. Touch, "Updated Specification of the IPv4 ID Field," RFC 6864, 2013.

[82] K. Wang, J. Zhang *et al.*, "It's not just the site, it's the contents: Intra-domain fingerprinting social media websites through cdn bursts," in *Proceedings of the Web Conference 2021*, 2021.

[83] T. Wang and I. Goldberg, "On realistically attacking tor with website fingerprinting," *Proceedings on Privacy Enhancing Technologies*, 2016.

[84] T. Wang, X. Xie *et al.*, "Netmamba: Efficient network traffic classification via pre-training unidirectional mamba," in *2024 IEEE 32nd International Conference on Network Protocols (ICNP)*, 2024.

[85] W. Wang, M. Zhu *et al.*, "End-to-end encrypted traffic classification with one-dimensional convolution neural networks," in *2017 IEEE International Conference on Intelligence and Security Informatics (ISI)*, 2017.

[86] Z. Wang, "The applications of deep learning on traffic identification," in *BlackHat*, 2015.

[87] W. Wei, Z. Ming *et al.*, "Malware traffic classification using convolutional neural network for representation learning," in *2017 International Conference on Information Networking (ICOIN)*, 2017.

[88] L. Xu, S. Ha *et al.*, "CUBIC for Fast and Long-Distance Networks," RFC 9438, 2023.

[89] L. Xu, X. Zhou *et al.*, "A traffic classification method based on packet transport layer payload by ensemble learning," in *2019 IEEE Symposium on Computers and Communications (ISCC)*, 2019.

[90] J. Yang, G. Liang *et al.*, "A deep-learning- and reinforcement-learning-based system for encrypted network malicious traffic detection," *Electronics Letters*, 2021.

[91] R. Zhao, M. Zhan *et al.*, "Yet another traffic classifier: A masked autoencoder based traffic transformer with multi-level flow representation," *Proceedings of the AAAI Conference on Artificial Intelligence*, 2023.

[92] ——, "A novel self-supervised framework based on masked autoencoder for traffic classification," *IEEE/ACM Transactions on Networking*, 2024.

[93] Z. Zhao, G. Yuanbo *et al.*, "The design and research for network address space randomization in openflow network," *Journal of Computer and Communications*, 2015.

[94] Z. Zou, J. Ge *et al.*, "Encrypted traffic classification with a convolutional long short-term memory neural network," in *2018 IEEE 20th International Conference on High Performance Computing and Communications; IEEE 16th International Conference on Smart City; IEEE 4th International Conference on Data Science and Systems (HPCC/SmartCity/DSS)*, 2018.

Appendix A. Cipherspectrum

A.1. Collection Methodology

Our primary objective is to capture encrypted network traffic and facilitate the downstream task of *web traffic classification* (see Section 2.2). Therefore, to create CipherSpectrum, we focused on collecting web traffic encrypted with the three cipher suites mandated/strongly-recommended by TLS 1.3: TLS-AES-256-GCM-SHA384, TLS-CHACHA20-POLY1305-SHA256, TLS-AES-128-GCM-SHA256 [66]. This strategy ensures that the dataset aligns with modern encryption standards and contemporary network traffic.

A.1.1. Browser Selection and Customization. To browse websites and generate network traffic, we chose two open-source web browsers: Firefox [22] and Chromium [11]. Firefox inherently allows users to configure cipher suite preferences, enabling us to modify the encryption algorithms used in secure communications directly. However, since Chromium does not provide this capability natively, we customized its source code to add a feature for selecting specific cipher suites. This browser selection and customization enabled us to create a controlled and diverse environment for generating traffic.

A.1.2. Domain Selection and Automation. To generate realistic web traffic, we selected the top 2000 domains listed on Cloudflare Radar [12] to align with recent trends and reliable traffic sources following the practice of [39], [50], [76][6]. After identifying the top domains, we implemented an automated process to verify their accessibility as follows: if a domain is accessible via a web browser without errors, we extract five URLs belonging to web-pages of the same domain. For example, for the domain "example.com", the extracted URLs would include: https://example.com, https://example.com/contact-us, https://example.com/settings, https://example.com/privacy, and https://example.com/about-us. Any domain that was either inaccessible via a browser or lacked five URLs from the same domain was excluded from the final list. We selected five web pages per domain to capture a relatively diverse set of traffic patterns, and to include a variety of content types within each domain. This filtering process resulted in a refined set of 132 domains and a total of 660 URLs.

A.1.3. Traffic Collection Process. Further adhering methodologies from previous studies [29], [34], [39], [76], [79], [80], [82], [83], we automated the traffic collection process using Selenium [71] as shown in Algorithm 2.

For each iteration of the traffic collection process, the algorithm begins by iterating over the three target cipher suites: TLS-CHACHA20-POLY1305-SHA256

6. We opted for Cloudflare Radar instead of currently dormant Alexa [41] rankings.

Algorithm 2 Network traffic collection

```
1: urls ← load 660 urls of shortlisted domains
2: for each iteration in {1 to 100} do
3:     for each c_suite in {"c20", "a128", "a256"} do
4:         for each browser in {"chromium", "firefox"} do
5:             for each url in urls do
6:                 run browser with c_suite
7:                 start traffic capture with tshark
8:                 load url with browser
9:                 capture screenshot after page load
10:                end traffic capture with tshark
11:                quit browser
12:            end for
13:        end for
14:    end for
15: end for
```

("c20"), TLS-AES-128-GCM-SHA256 ("a128"), and TLS-AES-256-GCM-SHA384 ("a256") *(Line 3)*. This ensures that traffic is collected in a balanced manner for all three cipher suites mandated/strongly-recommended by TLS 1.3. Within each iteration, the algorithm proceeds to loop through two selected browsers, Chromium and Firefox, to generate diverse traffic patterns *(Line 4)*.

For each URL in the set of 660 shortlisted URLs *(Line 5)*, the algorithm follows a systematic process. First, the browser is launched with the specified cipher suite for the current iteration *(Line 6)*, ensuring that all traffic generated in this session uses the intended encryption algorithm. Traffic capture then begins using Tshark *(Line 7)*, which records all incoming and outgoing network packets. The algorithm then directs the browser to load the selected URL *(Line 8)*, simulating a user accessing the webpage and generating realistic network traffic. Once the webpage is fully loaded, a screenshot is taken to document the state of the page during the session *(Line 9)*.

After this, the algorithm ends the traffic capture process and saves the recorded packets for analysis *(Line 10)*. Finally, the browser is quit to clean up session data and free up system resources *(Line 11)*. This entire sequence is repeated for each URL *(Line 5)* across each browser *(Line 4)* and each cipher suite *(Line 3)* within every iteration *(Line 2)*. By repeating this procedure across 100 iterations, the algorithm effectively captures a diverse dataset, reflecting traffic encrypted with different cipher suites in varied browser environments.

The sequential website visits allowed for capturing different variants of each site, resulting in a more comprehensive and representative traffic patterns [39], [76]. All data was collected using a university network, providing a realistic environment from January 13, 2024, to March 10, 2024.

A.2. Traffic Lebelling

At the conclusion of the traffic collection process, we successfully gathered a total of 396,000 traffic traces, resulting from 100 iterations across three different cipher suites, two browsers, and 660 URLs. To prepare this raw traffic data for analysis, we utilized SplitCap[7] to segment the collected traces into individual TCP/UDP sessions. During this process, we discarded any unencrypted sessions or unrelated network conversations, such as DNS and ARP traffic, to maintain a focus on encrypted communications.

The remaining split sessions represent individual, encrypted connections that were established to request various resources from servers in order to load the 660 URLs. To provide meaningful labels for these encrypted TCP/UDP sessions, we employed a Server Name Indication (SNI)-based labeling strategy. This technique, inspired by practices from previous studies [50], [51], [73], uses the SNI field within the TLS handshake to identify and label each session with the corresponding server name. This labeling approach ensures that each session is accurately associated with its originating domain, allowing for precise and reliable classification of network traffic based on the requested web resources.

Appendix B. Evaluation - Data Preprocessing

To represent various design choices using ET-BERT and YaTC, we extracted raw data following the methods outlined in Table 6. For both models, we adhered to the originally proposed number of bytes and packets to accurately capture different granularities. We also maintained the train, test, and validation splits as defined in the original studies to ensure consistency.

For both ET-BERT and YaTC, data selection incorporates all layers—*L2, L3, L4*, and *L7*—unless stated otherwise.

Table 6. DATA PREPROCESSING FOR SELECTED MODELS

Model	Packet	T1	T2	Burst, T3
ET-BERT	128B	640B	640B of 5 pkt	128B per pkt of 5pkt
YaTC	1600B	1600B	1600B of 5 pkt	320B per pkt of 5pkt

B=Bytes; **pkt**=Packets; **T3**=First m bytes per packet of n Packets; **T1**=First m bytes; **T2**=First m bytes of n packets;

7. SplitCap: https://www.netresec.com/?page=SplitCap

Appendix C. Meta-Review

The following meta-review was prepared by the program committee for the 2025 IEEE Symposium on Security and Privacy (S&P) as part of the review process as detailed in the call for papers.

C.1. Summary

This paper systematizes the space of works using machine learning classifiers on network traffic in the context of encrypted traffic. The key motivation is that, given modern protocols like TLS 1.3, studies of ML classifier efficacy on old datasets or using improperly curated or censored datasets may dramatically overestimate the accuracy of the resulting classifiers. The authors survey these prior works and identify a variety of risk areas for generalizing to modern protocols.

Based on this survey, the authors design an experiment for evaluating whether these risks occur in practice. They take existing open datasets and create a new CipherSpectrum datasets, designed to be reflective of properly encrypted traffic. They evaluate two SOTA traffic evaluaton ML approaches and show that, when the dataset is reasonably curated and irrelevant parameters are censored, the accuracy of these techniques declines dramatically. Further, the authors show that, in prior datasets, at least one of these models learns trends from fully-encrypted data, implying that part of their classification performance is resulting from weak encryption. The authors conclude that traffic classification approaches must carefully consider the effect of traffic encryption, and provide their reproducible dataset and evaluation to enable this.

C.2. Scientific Contributions

- Independent Confirmation of Important Results with Limited Prior Research
- Provides a New Data Set For Public Use
- Creates a New Tool to Enable Future Science
- Addresses a Long-Known Issue
- Provides a Valuable Step Forward in an Established Field

C.3. Reasons for Acceptance

1) The paper addresses a long-known issue and provides independent confirmation of important results with limited prior research. While the limitations of applying learning to encrypted traffic are straightforward, the wealth of works incorrectly attributing performance to encrypted traffic characteristics justifies a systematic review of these limits.
2) The paper provides a new data set for public use. The CipherSpectrum dataset contains a variety of encrypted traffic. Whereas prior datasets and related works contributed to confounding factors from incorrect train-test splits, CipherSpectrum specifically takes steps to eliminate these confounding factors.
3) The paper creates a new tool to enable future science. The paper evaluation is a reproducible artifact that can be used as a baseline for future network traffic classification studies.