Received XX Month, XXXX; revised XX Month, XXXX; accepted XX Month, XXXX; Date of publication XX Month, XXXX; date of current version XX Month, XXXX.

Digital Object Identifier 10.1109/TMLCN.2025.XXXXXXX

Fast Real-World Classification of ECH-Enabled Applications

Yizhak Kahana[1,2], Yaakov (J) Stein[1], Oren Glickman[2]

[1]Allot, Israel
[2]Bar Ilan University, Ramat Gan, Israel

Corresponding author: Yizhak Kahana (email: ikahana@allot.com).

ABSTRACT Encrypted Client Hello (ECH) reduces information leakage in encrypted packet connections, thereby complicating traditional Deep Packet Inspection (DPI) methods. While various machine learning approaches have been proposed for classifying ECH network connections, prior research typically requires lengthy observation times and focuses on a limited set of applications in constrained environments. This study demonstrates a highly accurate classification mechanism capable of identifying a large number of applications in real-world traffic within one second. To train our models we created a novel dataset derived from operational network traffic, ensuring relevance and diversity in training and evaluation. Furthermore, we propose a novel hierarchical approach that leverages protocol-specific knowledge, enabling scalable and efficient classification in operational settings. Our findings highlight the effectiveness of leveraging short Per Packet Information (PPI) sequences and domain-specific features to overcome the challenges posed by ECH, achieving performance superior to that of existing methods.

INDEX TERMS Encrypted Client Hello (ECH), Transport Layer Security (TLS), Encrypted Traffic Classification, Deep Packet Inspection (DPI), Machine Learning in Networking, Network Traffic Analysis, Real-Time Systems, Per Packet Information (PPI).

I. Introduction

SERVICE Providers require visibility into network traffic for a variety of critical tasks, including security, Quality of Service (QoS) monitoring, policy enforcement, congestion management, differentiated charging, regulatory compliance, and user sentiment analysis. Traditionally, Deep Packet Inspection (DPI) has been the primary method for enabling these capabilities, leveraging packet contents to accurately classify network traffic.

In the Internet protocol (IP) suite, packets are crafted to be self-describing, comprising a sequence of packet headers before the user content. This structure facilitates standards-based recursive parsing by the intended recipient, but has the unintended consequence of revealing information about the packet to potentially malicious parties observing the packet on its way from source to destination. This situation has been rectified by adding various cryptographic mechanisms, and it is now often stated that the great majority of Internet traffic is encrypted.

However, this statement regarding the ubiquity of encryption actually only signifies that user content is encrypted, while Internet packet headers still include unencrypted metadata. This metadata can be exploited by Deep Packet Inspection (DPI) to identify the application being used.

A notable (but far from the only) example of unencrypted fields are those in the Transport Layer Security (TLS) handshake. The TLS handshake is used by two parties wishing to securely communicate, before they have established their secure channel, and consists of a Client Hello (CH) message followed by a Server Hello (SH) message. The CH specifies the cipher suites the client supports, various crypto-related parameters, and optionally the name of the server (SNI) the client wants to reach (since there may be many virtual servers hosted behind a single IP address!). Since the cipher suite has not yet been agreed, the CH message is unencrypted, passing SNI and other information in the clear.

Over the past few years Internet traffic has become more fully encrypted, with sophisticated methods of encrypting overhead fields, including encrypted Client Hello (ECH) [1]. ECH, as its name implies, encrypts the TLS CH message using a public key previously obtained.

While encrypting metadata enhances privacy, it introduces significant challenges for DPI systems, impacting essential functions like security, QoS monitoring, and regulatory compliance. However, encrypted traffic still exhibits observable statistical patterns in packet sequences. Characteristics such as packet directionality (client-to-server vs. server-to-client), packet sizes, and inter-packet timing may form unique signatures for different applications. For example, video streaming traffic consists of sequences of client requests followed by server responses consisting of many large packets interspersed with acknowledgments, while voice calls exhibit more symmetric behavior with intermediate sized packets in both directions.

Previous machine learning approaches have explored classification using packet size and timing statistics. However, they suffered from limitations: (1) high computational and energy costs compared to rule-based DPI systems, (2) reliance on long packet sequences, making them unsuitable for scenarios such as blocking, and (3) only classifying small numbers of applications (as compared to thousands of applications recognized by DPI systems)..

This paper makes the following key contributions:

- We introduce a novel, real-world dataset capturing encrypted traffic from an operational network, ensuring diverse and representative training and evaluation conditions.
- We develop a high-accuracy classification mechanism capable of distinguishing a large set of applications with only one second of real-world traffic observation.
- We propose a hierarchical classification framework that leverages domain-specific knowledge, grouping traffic based on visible server IP addresses to enhance scalability and efficiency under ECH constraints.
- We demonstrate that short packet sequences (as few as 10 packets) can be sufficient for classification, enabling fast decision-making with minimal input data.

Our findings highlight the effectiveness of leveraging packet-level behavioral features to overcome the challenges posed by ECH. The proposed hierarchical model further improves classification efficiency by assigning smaller, specialized classifiers to subsets of applications, maintaining high accuracy while reducing computational overhead.

II. Background and Related Work

Recent research has demonstrated the potential of machine learning (ML) techniques for classifying encrypted traffic. Early attempts were based on the byte content of packets of the connection, such as IP addresses and SNI [2]–[7], and are thus rendered ineffective in the presence of ECH. Subsequent studies exploited statistical features of traffic patterns, such as histograms of packet sizes and inter-arrival times. However, these methods often require long observation windows (e.g., 30 seconds or more) [8], [9], rendering them unsuitable for many scenarios. A recent paper from Huawei France [10] achieves 90% accuracy across the top 200 applications of a proprietary dataset by analyzing the Per Packet Information (PPI) of the first 100 packets in TCP connections or 10 packets in UDP connections. However, requiring 100 packets significantly increases classification delay and limits real-time applicability.

To date, most studies have focused on a small number of applications (fewer than 20) [2]–[6], [11], [12], limiting their applicability in operational settings. For instance, FlowFormers [9] employs attention-based Transformers for classification within 10 seconds but has only been evaluated on a 5-class problem.

Existing datasets often lack the diversity and relevance needed for operational use. For instance, [13]–[15] are already outdated, while others rely on simulated traffic [16] or advanced data augmentation techniques to compensate for limited data diversity [17]. Proprietary datasets further hinder reproducibility and fair benchmarking [10], [11].

The CESNET datasets [18], [19] are valuable sources of encrypted traffic data but also exhibit several limitations. A first dataset, limited to TLS/TCP traffic, was collected over two weeks in 2021, while a QUIC-only dataset was collected over a month in 2022. The former encompasses some 200 applications, while the latter has far fewer due to QUIC still having been in nearly stages of adoption. Both include the PPI of 30 packets for each connection and label connections based on the SNI.

Despite these advancements, a significant gap remains in the literature. To the best of our knowledge, no prior work has conducted a comprehensive, large-scale study targeting the classification of a substantial number of applications under stringent time constraints[1]. Our combination of real-world data, real-time applicability, and hierarchical classification represents a significant step forward in encrypted traffic classification, overcoming the limitations of existing approaches.

III. Problem Formulation

The goal of this study is to classify encrypted network traffic into a predefined set of applications under strict time and data constraints. We limit ourselves to features observable in the presence of ECH.

Formally, let $\mathcal{C} = (\mathrm{PCI}, \mathrm{PPI})$ represent a single network connection comprising: Per Connection Information $\mathrm{PCI} = (\mathrm{DestIP}, \mathrm{DestPort}, \mathrm{protocol})$ and a sequence of k packets with corresponding per packet information $\mathrm{PPI} = \langle \mathrm{PPI}_1, \mathrm{PPI}_2, \ldots, \mathrm{PPI}_k \rangle$ where $\mathrm{DestIP}, \mathrm{DestPort}$ are the destination IP address and port number, protocol is the layer 4 protocol (e.g., TCP or UDP) and each PPI_i includes: $(\Delta t_i, s_i, d_i)$, where Δt_i is the arrival time of the i-th packet relative to the first packet (in milliseconds), s_i is the size of the i-th packet (in bytes) and d_i is the direction of the i-th packet (client-to-server or server-to-client). The task is to classify a connection $\mathcal{C}$ into one of M application classes: $\mathrm{Apps} = \{\mathrm{App}_1, \mathrm{App}_2, \ldots, \mathrm{App}_M\}$.

The classification task must furthermore adhere to the following constraints: the number of packets per connection,

[1]A preliminary version of this work was presented at CSCML 2024 and is available as a technical report [20].

k, is limited to a predefined maximum (e.g., 50), and the total observation time for any connection, $\max(\Delta t_i)$, must not exceed a specified threshold (e.g., 1 second).

Formally, given a training dataset of annotated connections $\mathcal{X}_{\text{train}} = \{c_i\}_{i=1}^n, \mathcal{Y}_{\text{train}} = \{\text{app}_i\}_{i=1}^n$, the objective is to learn a classification model $f : \mathcal{X} \rightarrow \mathcal{Y}$ that maps a connection c to a predicted application $\text{App}_k \in \text{Apps}$, minimizing the overall classification error.

The model performance is evaluated on a separate test dataset of connections collected over a later time span $\mathcal{X}_{\text{test}}, \mathcal{Y}_{\text{test}}$ to assess its generalization capacity.

IV. Dataset Construction

A. Data Collection

The dataset for this study was derived from live network traffic captured in a large mobile network starting in September 2024. To capture variability in network conditions and reflect a diverse range of usage patterns, the raw traffic was partitioned into two subsets, each spanning approximately four days (covering both weekdays and weekends). The first subset was reserved for model training, while subsequent subset, collected at later dates, served as the test set.

Traffic data was collected by enhancing a commercial real-time DPI system from Allot Ltd.[2] operating in tap mode on an $N \times 100$ Gbps bidirectional line. This DPI system is highly fine-grained and can differentiate between sub-applications from the same provider. For example, it distinguishes between Facebook browsing, Facebook video streaming, and Facebook video uploads as distinct applications. Such granularity ensures detailed and accurate labeling.

The DPI system assigns application labels (`AppID`) using a proprietary rule set that is updated biweekly. The system achieves approximately 95% coverage of network connections (leaving only about 5% unclassified) and exhibits an extremely low misclassification rate compared to simpler labeling approaches, such as relying solely on Server Name Indication (SNI) [11] or server IP addresses. These attributes make it particularly well-suited for generating high-quality labeled datasets for machine learning tasks.

Additionally, connections are sampled by hashing the client IP address, enabling us to vary the sampling rate. Overall, the captured data encompasses 195 hours of traffic, comprising 313 million raw connections (averaging 445 connections collected per second).

We further filtered out the following uninformative connections:

- 146 million (47%) unencrypted DNS sessions,
- 31 million (10%) connections insufficiently identified by the DPI system.

The fact that half of the connections were DNS is to be expected. Of the insufficiently specified connections, over half were truly unknown applications (to be expected since the DPI coverage is on the order of 95%) and the rest were only partially identified. In any case such connections are uninteresting for training purposes and were removed, leaving 136 million nontrivial labeled connections.

[2]https://allot.com/network-intelligence/technology/traffic-identification/

B. The Dataset Format

Each network connection is represented by connection-level identifiers (the 5-tuple consisting of two IP addresses, the layer-4 protocol, and two TCP/UDP port numbers), an array of up to 50 PPI entries (collected under the one-second constraint), and an application label. To preserve user privacy, all IP addresses are anonymized using the prefix-preserving CryptoPAN algorithm [21].

The following table (Table 1) summarizes the fields for each connection.

TABLE 1. Dataset Fields for Each Network Connection

Per-Connection Fields	
Field	**Description**
Time of first packet	For reference only Not used for classification
Obfuscated source IP	
Source port number	
Obfuscated destination IP	Destination IP address after anonymization
Layer 4 protocol number	Protocol used (e.g., UDP, TCP)
Destination port number	Port number (e.g., 443/8443, 80/8080)
`AppID`	Numeric label identifying the application
Length of PPI array	Up to 50 packets
Per-Packet Fields	
Field	**Description**
Packet arrival time	Arrival time relative to the first packet in milliseconds
Packet direction	Client-to-server or server-to-client
Packet payload size	Size in bytes (corresponding to TCP/UDP payload)
Flag byte	Encodes protocol-specific information

The **flag byte** contains additional information and is treated differently based on the protocol: For TCP packets, it encodes the eight TCP flags, of which only the SYN flag is used for classification. For UDP packets, it includes the first byte of the UDP payload. In the case of QUIC packets, this byte holds four unencrypted bits (e.g., header format, fixed bit, long packet type field), but only the header format bit is used for classification.

The AppID label is supplied by the commercial DPI system and is further detailed in the following subsection.

C. Dataset Statistics

Table 2 summarizes key statistics of the dataset, which comprises a total of 136,201,280 connections. The table details the number of connections, client IPs, and server IPs for both the training and test sets, collected over two non-overlapping periods in September 2024.

A total of 823 applications were reported by the DPI in the dataset. Of the 763 applications in the training set, 115 appear in over 0.05% of the connections. Applications that attracted at least 1% of connections are presented in Table 3.

TABLE 2. **Key Statistics of the Dataset**

Field	Train	Test	All
Date span (Sep. 2024)	5-9	13-17	5-9,13-17
Total number of connections	63,067,349	73,133,931	136,201,280
Total number of client IPs	227,296	278,729	489,586
Total number of server IPs	521,011	521,017	924,755

TABLE 3. **Most popular applications with their appearance percentages.**

GoogleServices	14.56 %	WhatsApp Transfer	2.08 %
TikTok	10.55 %	Google Search	2.01 %
Facebook	6.98 %	Advertisements	1.98 %
AppleServices	6.66 %	Computing sites	1.76 %
WhatsApp	4.37 %	Telegram	1.61 %
Instagram	3.79 %	YouTube	1.43 %
Analytics	3.33 %	iTunes	1.36 %
Google Play	3.33 %	Banners	1.29 %
Other (< 0.05%)	3.03 %	Snapchat	1.27 %
iCloud	2.90 %	Generic CDN	1.19 %
YouTube Browsing	2.22 %	Facebook Chat	1.14 %

The reader will note that applications are identified by the DPI system in an *actioanable* manner. For instance, the most popular application, entitled GoogleServices, is actually an aggregate of multiple Google-provided services that are mostly transparent to the user, including push notifications, time services, website optimizations, translation, etc. AppleServices includes similar miscellaneous facilities related to Apple products. Analytics and advertisements are as their name implies from many different servers. We note that these application popularities are highly dependent on the network being a mobile network, and fixed networks would have different statistics. For example, Netflix accrues only about 0.1 % in this cellular network, while it is usually one of the more popular applications in fixed networks. Furthermore, application popularity is highly dependent on locality.

In terms of the relationship between server IP addresses and applications, over the training set, $\sim$91% of the server IP addresses are associated in the data to a single application, 6% to two applications and 1% to three. However, in terms of connections, only $\sim$10% of the connections are to a server IP which is mapped in the training set to a single application.

D. Dataset Availability

A sample of the dataset in CSV format has been made publicly available via a GitHub repository[3]. The full dataset is available for academic purposes and can be obtained by contacting the corresponding author.

V. Methodology

In this study, we develop and evaluate three baseline models — a Random Forest classifier, an LSTM neural network, and an LSTM with Attention — to classify encrypted network traffic based on PPI features. In addition, we propose a novel hierarchical classification model that groups connections based on server IP addresses and trains separate ML models on each group, thereby leveraging domain-specific knowledge to improve scalability and accuracy.

[3]https://bit.ly/40oluem

A. Baseline Models

We trained two types of classifiers (a Random Forest and a Long Short-Term Memory (LSTM) neural network) and employed two stratagems (*hierarchical* and *flat*). Comparing the resulting four architectures enables assessing the effectiveness of conventional machine learning approaches versus deep learning methods, as well as the value of exploiting domain-specific knowledge.

In addition, we trained an additional LSTM model augmented with an *attention* mechanism to explore its potential for enhancing sequence modeling and improving classification accuracy. The attention mechanism allows the model to focus selectively on the most relevant parts of the input sequence, potentially improving performance for PPI sequences with complex temporal dependencies.

We first evaluated the models under two configurations: one that excluded protocol-specific knowledge, aligning with the flat (non-hierarchical) methods typically described in the literature, and another that incorporated this knowledge by including the server's IP address as a feature. This approach enabled a comparative analysis to determine the impact of domain-specific features on classification performance.

1) Random Forest Classifier

A Random Forest classifier was employed as a baseline model for classifying encrypted network traffic. Given that the connections are of variable number of packets, we first converted the PPI data into fixed-length feature vectors by zero-padding each sequence to the maximum packet length we allow for training. The individual PPI components, namely, packet time, direction, size, and flags, were extracted and padded accordingly, and then concatenated with per-connection features (including the Layer-4 protocol number, destination port, and the PPI sequence length).

We also tested a configuration in which the server's IP address was incorporated as an additional per-connection feature. To achieve this, the server IP was transformed into a numeric representation: IPv4 addresses were converted into their 32-bit integer equivalents, while IPv6 addresses were converted into their full 128-bit integer representations. This conversion allowed the Random Forest to leverage the server IP as a meaningful numerical feature without requiring further normalization, given the model's inherent scale invariance.

2) LSTM Neural Network

To better capture the temporal dependencies in the packet-level data, we developed a deep LSTM neural network that processes variable-length sequences. The architecture inputs the same per-connection and per-packet information as the Random FOrest model, but does not pad the PPI.

Initially, the PPI sequence is processed using a *Pack Padded Sequence* operation to handle variable-length inputs. The packed sequence is then fed into a stack of four LSTM layers (each with 128 hidden units), which capture the temporal dynamics of the packet sequences. After processing, the sequence is *unpacked* and the final hidden state corresponding to the last time step (of size 128) is extracted as a compact representation of the sequence.

To incorporate the server's IP address into the model, the IP is first converted into an integer representation. For compatibility with the LSTM, this numeric value is normalized to the range $[0, 1]$ and then concatenated with other per-connection features (protocol number, destination port, and PPI length), resulting in a combined feature vector.

The combined feature vector is then passed through two fully connected layers: first, FC Layer 1 reduces the input (of size 131, comprising 128 from the LSTM output plus 3 scalar per-connection features, or 4 if the server IP is included) to 64 dimensions; next, a ReLU activation is applied to introduce non-linearity; finally, FC Layer 2 transforms the 64-dimensional vector into C class scores, where C is the number of application classes.

Figure 1A illustrates the overall architecture of the LSTM model.

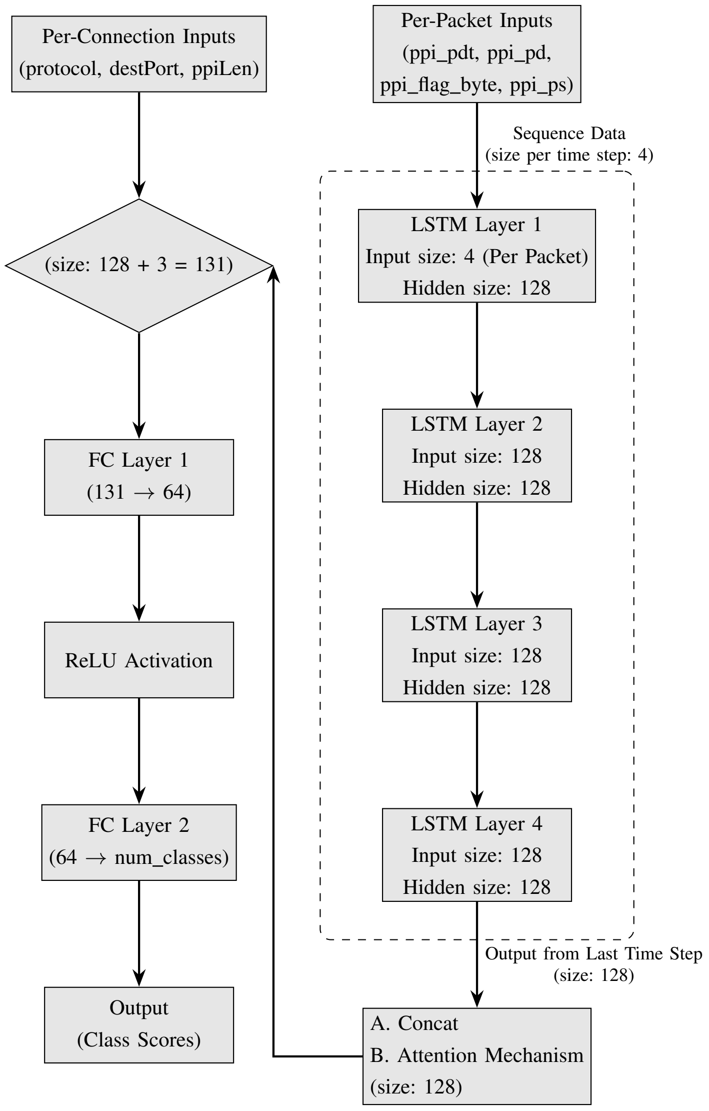

FIGURE 1. **Architecture of the LSTM. (A) without and (B) with Attention.**

3) LSTM with Attention

In addition to the basic LSTM architecture, we enhanced our model by incorporating an attention mechanism to further improve exploitation of temporal dependencies in the packet-level data. Unlike the standard LSTM, which relies solely on the final hidden state to summarize the entire sequence, the attention mechanism dynamically computes a weighted sum of the hidden states from all time steps. This *context vector* highlights the most informative parts of the sequence, allowing the model to focus on critical patterns that may otherwise be diluted in a single, compressed representation. After the LSTM layers produce a sequence of outputs, the attention mechanism computes the context vector (of size 128), which is then concatenated with the per-connection features (e.g., protocol number, destination port, PPI length, and optionally the server's IP address). The resulting combined feature vector is subsequently fed through two fully connected layers that output class scores for each application. Figure 1B illustrates the overall architecture of the LSTM with Attention model.

B. Hierarchical Classification

In contrast to the standard *flat* classification approach, which employs a single monolithic approach to classify all network connections, we propose a hierarchical classification model that leverages domain-specific knowledge inherent in the server IP address left not encrypted even under ECH. In our hierarchical approach, the training data is first partitioned into groups based on server IP, and then a dedicated classifier is trained for each group. This approach not only simplifies and accelerates the training process but also reduces the impact of misclassification errors by limiting them to a subset of related applications.

The overall flow of our proposed hierarchical classification model is illustrated in figure 2.

The first level of our hierarchy identifies server IP addresses that predominantly host a single application. Formally, let S denote the set of all server IP addresses in the

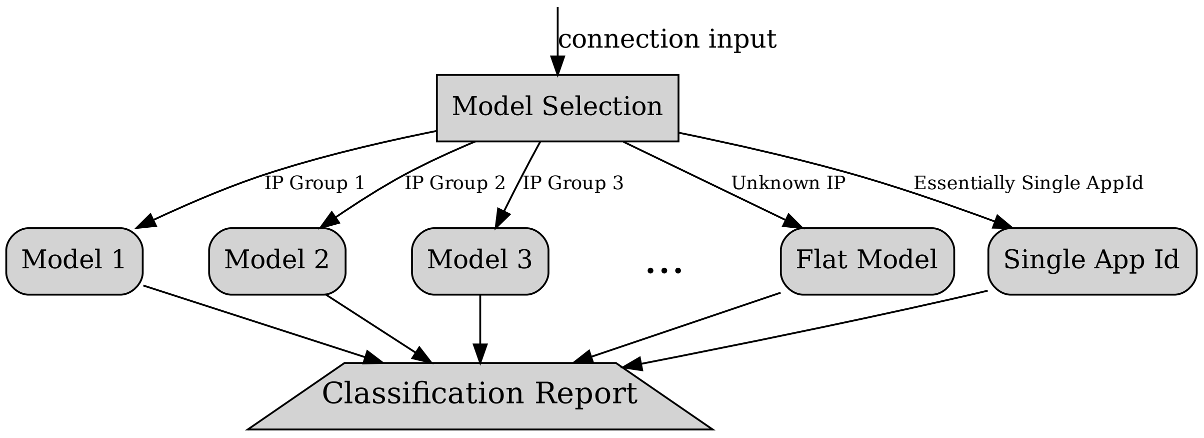

FIGURE 2. **Hierarchical Decision Flow**

training data. For each address $s \in S$, let N_s be the total number of connections associated with s and $N_s(\text{App})$ the number of connections labeled with application App. We define the dominant application for server s as

$$\text{App}_s^* = \arg\max_{\text{App}} N_s(\text{App}),$$

and consider s to host essentially a single application if

$$\frac{N_s(\text{App}_s^*)}{N_s} \geq \tau.$$

where τ is a threshold parameter (set to 0.95 in our experiments). For these servers, classification is straightforward: the predicted application is simply App_s^*.

The second layer of our hierarchy involves popular server IP addresses hosting multiple applications. We partition such server IP addresses into distinct groups. Once the groups are defined, a separate classifier is trained for each group. Three grouping strategies were considered: ASN-based grouping, application-based grouping, and application distribution clustering.

Connections corresponding to server IP addresses not identified in the first or second levels are handed off to a single flat classifier trained on the entire dataset.

Several grouping strategies were considered, including ASN-based grouping, application-based grouping, and application distribution clustering.

ASN-based Grouping: This method maps server IP addresses to their corresponding Autonomous System Numbers (ASNs), under the assumption that servers within the same AS are likely to host the same applications. For instance, server IP addresses associated with Google's ASN (15169) typically host Google services (e.g., Google Search, Gmail, Google Maps, or YouTube), whereas those in Meta's ASN (32934) generally host Facebook, Instagram, or WhatsApp. A drawback of this approach is that large companies may subdivide their IP address resources, meaning not all addresses within an ASN necessarily host the same services.

Application-based Grouping: In this approach, servers are partitioned based on the dominant application that constitutes a significant portion of their activity. All server IP addresses with the same dominant application are grouped together, facilitating targeted model training on a more homogeneous subset of traffic.

Application Distribution Clustering: This method groups servers by analyzing the probability distribution of applications they serve. For each server, the algorithm computes the distribution of application labels, and servers with similar distributions are clustered together. This approach captures nuanced similarities in application usage patterns, potentially leading to more effective grouping.

In the following we report results for ASN-based grouping; we map popular server IP addresses to their ASN using publicly available information. In our training set we found 11 popular ASN groups (encompassing over 45% of the connections) with between 10 and 30 applications per group. As expected, ASN groups included Google, Apple, Amazon, Microsoft, Akamai, Cloudflare, and several CDNs.

VI. Experimental Setup

A. Experimental Configuration and Evaluation

Our dataset contains over 800 distinct applications; however, many of these occur so infrequently that they are not practical for separate classification. Therefore, we aggregate all applications appearing in less than 0.05% of the connections into a single label, *Other*. This aggregate label is the expected output for applications appearing in the test set but that were absent from the training set.

We evaluate our models using standard classification metrics: Precision, Recall, and F1-score. We compute these metrics for each class individually and then combine them using weighted averaging in order to account for potential class imbalance.

B. Hyperparameter Settings and Implementation Details

We utilized scikit-learn to train the Random Forest classifier and PyTorch for the LSTM-based models. For the Random Forest classifier, we used 15 trees with a maximum depth of 35 and a random state of 42 to ensure reproducibility. The entire training dataset was utilized without a separate validation split.

For the LSTM models, input sequences were first processed using the Pack Padded Sequence operation to accommodate variable-length PPI data. The LSTM architecture consists of four stacked layers, each with 128 hidden units. Following the LSTM layers, the extracted features are concatenated with per-connection features and passed through two fully connected layers: the first reduces the combined feature vector from 131 to 64 dimensions (with ReLU activation), and the second maps the 64-dimensional vector to C class scores, where C is the number of application classes. The LSTM-based models were trained using the Adam optimizer with an initial learning rate of 10^{-3} and a StepLR scheduler that reduces the learning rate by a factor of 0.5 every 5 epochs. Training was performed for 20 epochs with a batch size of 128, and gradient clipping (maximum norm of 5) was applied to prevent diverging gradients.

The hierarchical classification approach utilizes the same configurations as the baseline models. However, it trains separate classifiers for each server group, enabling a more tailored and scalable approach.

To comprehensively evaluate our methodology, we conducted experiments under multiple configurations. For the baseline (flat) models, each model was evaluated both with and without the inclusion of the server IP as an additional per-connection feature, and under two PPI length limits: 10 and 50 packets. In parallel, the hierarchical classification approach employing ASN-based grouping was tested using all three underlying baseline architectures (Random Forest, LSTM, and LSTM with Attention) under both 10 and 50 PPI length limits. This experiment design enables a detailed analysis of the impact of protocol-specific features and sequence length on classification performance.

VII. Results and Analysis

Table 4 presents a comprehensive comparison of the three baseline models (Random Forest, LSTM, and LSTM with Attention) evaluated under two packet length restrictions (10 and 50 packets) and with two baseline configurations (with and without the inclusion of the server IP as an additional feature) as well as the proposed hierarchical approach. The table shows that performance, as measured by Precision, Recall, and F1 Score, is highly sensitive to both the packet restriction and the use of protocol-specific features. In general, reducing the packet restriction hampers results, due to the reduced amount of input information available for classification. Notably, the Random Forest model benefits significantly from incorporating the server IP feature, particularly when using the shorter (10-packet) limit, where it actually achieved the best performance overall. For the 50-packet restriction no clear winner emerges among the baseline models, suggesting that the additional input data mitigates the advantages of protocol-specific features. These findings underscore the importance of optimizing both the feature configuration and the packet length restriction to achieve accurate real-time classification of encrypted traffic.

Our experiments further show that our proposed hierarchical approach consistently outperforms the flat classification approach. Specifically, when using a packet restriction of 50, the best performing model is the hierarchical model using deep LSTM model with attention achieving approximately 95% accuracy - a 5% improvement over the equivalent flat model.

There seems to be no significant advantage for deep learning models in this problem. Although there is a slight performance improvement when operating on 50 packets, it remains insufficient to justify the complexity and computational overhead of deep learning approaches—especially in real-time systems that would require dedicated hardware such as GPUs.

A detailed comparison reveals that while the Random Forest model benefits significantly from the inclusion of the server IP feature—especially at the 10-packet restriction—the LSTM-based models exhibit varying performance trade-offs. In fact, the Random Forest classifier outperforms the LSTM model at the shorter packet restriction, likely because the LSTM's ability to capture temporal patterns is diminished when fewer packets are available. Additionally, incorporating an Attention layer did not yield noticeable improvements, suggesting that the features (e.g., packet sizes, directions, and inter-arrival times) do not present distinct key points that benefit from selective focusing.

The hierarchical approach consistently outperformed the flat architecture by more than 5% in accuracy. The 10 packet restriction led to some performance degradation, but the the impact was not substantial. We conclude that this approach more effectively leverages domain-specific information than merely adding server IP address to the input, in addition to being more scalable and maintainable.

VIII. Distributional Drift Analysis

Due to the dynamic nature of modern applications, machine learning models often experience a decline in performance when applied to data collected at later time periods. This

TABLE 4. **Model Performance for Different Configurations**

Packet Restriction	Model Config.	Random Forest			LSTM			LSTM w/ Attention		
		Precision	Recall	F1 Score	Precision	Recall	F1 Score	Precision	Recall	F1 Score
50	Flat	88.42	88.26	88.07	90.06	89.93	89.84	90.18	90.09	**89.99**
	+ Server IP	90.72	90.63	**90.47**	89.13	89.05	88.95	88.11	88.09	87.93
	Hierarchical	93.58	93.54	93.42	94.68	94.64	94.57	94.83	94.79	**94.73**
10	Flat	88.15	88.11	**87.92**	86.34	86.35	86.05	86.25	86.19	85.94
	+ Server IP	90.66	90.66	**90.50**	86.39	86.45	86.15	87.47	87.50	87.25
	Hierarchical	92.71	92.71	**92.58**	92.15	92.12	91.97	92.52	92.46	92.32

distributional drift arises from new application releases, modifications in communication protocols, shifts in user behavior, and the adoption of emerging technologies. Consequently, the accuracy of models trained on historical data is expected to decline over time.

Addressing distributional drift typically requires continuous monitoring and periodic model retraining. While techniques like online learning, domain adaptation, and transfer learning have been proposed, they often introduce additional computational overhead and complexity. To better understand the practical impact of drift on our model's performance, we conducted an experimental evaluation using test datasets collected at later time intervals.

A. Additional Dataset Collection and Evaluation

To assess the quantitative effects of distributional drift, we collected additional datasets from the same large mobile network operator at intervals of one and two months following the initial model training period. Each drift dataset spans approximately four days of network traffic, covering both weekdays and weekends, to capture a representative range of network conditions and user behaviors. The temporal spacing of one and two months was chosen to reflect potential shifts in traffic patterns due to seasonal changes, application updates, and evolving user trends. These drift-testing datasets underwent the same preprocessing pipeline, and were evaluated using the same metrics as the original training and testing datasets, enabling consistent performance comparisons.

B. Results

Table 5 presents the F1 performance of all model configurations evaluated on test data collected one week (the original test set), one month, and two months after training. Overall, the results indicate a consistent drop in F1 score over time. The performance drop varies with configuration, being generally less pronounced for hierarchical approach and more significant for the 10-packet PPI length restriction case.

The hierarchical approach, with its protocol-specific grouping, experiences an F1 score drop of approximately 2% after one month. Despite this moderate degradation, it maintains an F1 score above 90% even after two months, showcasing its robustness. In contrast, the flat models exhibit significantly higher performance drops, particularly in the second month. For example, the Random Forest model under the flat configuration drops by 2.41% at one month and 8.73% at two months, compared to the hierarchical model which only drops by 1.68% and 3.60%, respectively.

The inclusion of server IP information in the flat models does not prevent their performance from degrading over time. These results underscore the effectiveness of the hierarchical approach for dynamic environments, where distributional drift is inevitable. The ability to maintain performance well beyond the one-month mark validates the hierarchical approach's potential in long-term deployment scenarios.

Figure 3 shows the F1 score trends for selected IP groups when using the hierarchical model with a 50-packet PPI restriction and the Random Forest classifier applied to each group. Our analysis reveals distinct drift patterns across different groups. **Stable groups**—such as the ASN-based groups for Apple, Microsoft, and the single AppId group—exhibit only minor performance degradation (approximately a 2% drop in F1 score after one month) and maintain F1 scores above 90% even after two months. In contrast, **moderate shifts** are observed in some groups (e.g., the Google group), where the F1 score declines significantly within one month, indicating that these groups may require targeted updates. Additionally, **delayed degradation** is evident in groups such as Facebook and Amazon, where performance remains relatively stable after one month, but drops markedly by the two-month mark.

Further analysis shows that the number of PPIs required for the model to achieve optimal performance may vary across different groups. In some groups, such as Apple or Microsoft, the difference in performance between processing 50 and 10 packets is negligible. While in others, e.g., Amazon, comparable results are achievable even with only 10 packets over all time periods.

These findings support a selective retraining strategy, where only server groups that exhibit substantial drift are updated. By focusing retraining efforts on groups experiencing significant performance degradation, computational costs can be reduced while maintaining resilience against drift, thereby extending the system's usability in real-world, dynamic environments.

TABLE 5. Distributional Drift Performance Comparison

packet restriction	Model Config.	Method	One Week	One Month		Two Months		
			F1 (%)	F1 (%)	Drop (1m%)	F1 (%)	Drop (1m%)	Drop (2m%)
50	Flat Model	Random Forest	88.07	85.95	2.41	80.38	6.48	8.73
		LSTM	89.84	87.64	2.45	81.90	6.55	8.84
		LSTM w/ Attention	89.99	87.58	2.68	82.01	6.36	8.87
	Flat w/ Server IP	Random Forest	90.47	87.24	3.57	82.12	5.87	9.23
		LSTM	88.95	86.48	2.78	81.58	5.67	8.29
		LSTM w/ Attention	87.93	85.61	2.64	80.17	6.35	8.83
	Hierarchical	Random Forest	93.42	91.85	1.68	90.06	1.95	3.60
		LSTM	94.57	**92.93**	1.73	90.98	2.10	3.80
		LSTM w/ Attention	**94.73**	92.90	1.93	**90.91**	2.14	4.03
10	Flat Model	Random Forest	87.92	85.70	2.53	79.95	6.71	9.07
		LSTM	86.05	83.65	2.79	77.59	7.24	9.83
		LSTM w/ Attention	85.94	83.41	2.94	77.26	7.37	10.10
	Flat w/ Server IP	Random Forest	90.50	88.50	2.21	83.15	6.05	8.12
		LSTM	86.15	83.46	3.12	77.31	7.37	10.26
		LSTM w/ Attention	87.25	84.78	2.83	78.72	7.15	9.78
	Hierarchical	Random Forest	**92.58**	**90.85**	1.87	**89.35**	1.65	3.49
		LSTM	91.97	89.98	2.16	88.43	1.72	3.85
		LSTM w/ Attention	92.32	90.32	2.17	88.80	1.68	3.81

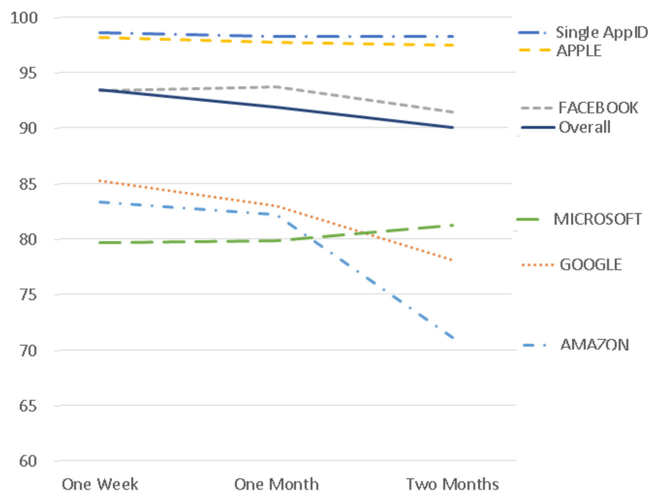

FIGURE 3. F1 score trends for selected IP groups over time using the hierarchical model with a 50-packet PPI restriction and a Random Forest classifier. The figure illustrates the degradation in performance from one week to one month to two months, highlighting groups with stable performance, moderate shifts, and delayed degradation.

IX. Conclusions

In this work, we have addressed the challenge of real-time classification of encrypted network traffic in the presence of Encrypted Client Hello (ECH). The introduction of ECH significantly impacts the effectiveness of traditional Deep Packet Inspection (DPI) systems, necessitating more sophisticated machine learning techniques. Unlike previous studies that mostly focus on small-scale datasets with limited application diversity and long observation delays, our work provides a practical, real-world solution capable of identifying over 100 applications with only one second of observation. We furthermore proposed a novel hierarchical classification approach that integrates domain-specific knowledge by leveraging server IP addresses, which remain unencrypted under ECH. Compared to conventional flat classification approaches, our hierarchical model offers multiple advantages:

- **Improved Scalability:** The hierarchical approach allows for structured classification by breaking down the problem into smaller, more manageable subproblems.
- **Higher Accuracy:** By focusing on groups of applications within known server IP ranges, the model improves prediction accuracy compared to a flat classification model.
- **Less Drift:** The hierarchical approach is observed to suffer less performance degradation over time.
- **Reduced Computational Complexity:** Compact submodels decrease overall model complexity, improving both training efficiency and inference speed.
- **Enhanced Maintainability:** When applications or network behaviors change over time, only specific submodels require updating.

The evaluation results demonstrate that the hierarchical model consistently outperforms traditional flat classifiers, particularly in handling evolving traffic patterns. Even under significant concept drift, our model maintains robust classification performance, with an F1 score remaining above 90% even after two months. Additionally, we observed that reducing the packet restriction negatively impacts performance, though hierarchical classification mitigates this effect better than flat models.

One key insight from our experiments is that Random Forest models performed comparably to deep LSTM-based approaches while being more computationally efficient. This suggests that complex sequence-based models may not always be necessary for encrypted traffic classification.

Looking ahead, there are several directions still open for future research:

- **Refining Hierarchical Grouping:** Exploring more granular grouping methods, such as dynamically clustering applications based on real-time traffic patterns, that may further improve classification accuracy and adaptability.
- **Selective Retraining Strategies:** Implementing adaptive learning techniques, which selectively retrain only the submodels experiencing drift, could reduce the computational cost of model updates while maintaining performance.
- **Generalization Across Networks:** Evaluating the model's applicability across different geographic regions and network environments to ensure its robustness in diverse real-world scenarios.
- **Adaptive Packet Processing:** Investigate dynamic models that determine the optimal number of packets for classification in real-time. By enabling early stopping after 10 packets in some cases while utilizing up to 50 in others, one could optimize the tradeoff between classification accuracy and inference efficiency.

Our findings highlight the usefulness of considered leveraging of structured domain knowledge, in contrast to blindly applying machine learning tools to input data. One may speculate as to the generality of this result is, and in which other cases such techniques may be advisable.

REFERENCES

[1] E. Rescorla, K. Oku, N. Sullivan, and C. A. Wood, "Tls encrypted client hello," Internet Engineering Task Force, Internet-Draft draft-ietf-tls-esni-23, Mar. 2025, work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/draft-ietf-tls-esni/23/

[2] W. Zheng, J. Zhong, Q. Zhang, and G. Zhao, "Mtt: an efficient model for encrypted network traffic classification using multi-task transformer," *Applied Intelligence*, vol. 52, no. 9, pp. 10 741–10 756, 2022.

[3] H. Y. He, Z. G. Yang, and X. N. Chen, "Pert: Payload encoding representation from transformer for encrypted traffic classification," in *2020 ITU Kaleidoscope: Industry-Driven Digital Transformation (ITU K)*. IEEE, 2020, pp. 1–8.

[4] P. Lin, Y. Hu, Y. Lin, K. Ye, and C.-Z. Xu, "Pean: A packet-level end-to-end attentive network for encrypted traffic identification," in *2021 IEEE 23rd Int Conf on High Performance Computing & Communications; 7th Int Conf on Data Science & Systems; 19th Int Conf on Smart City; 7th Int Conf on Dependability in Sensor, Cloud & Big Data Systems & Application (HPCC/DSS/SmartCity/DependSys)*. IEEE, 2021, pp. 267–274.

[5] J.-T. Park, C.-Y. Shin, U.-J. Baek, and M.-S. Kim, "Fast and accurate multi-task learning for encrypted network traffic classification," *Applied Sciences*, vol. 14, no. 7, p. 3073, 2024.

[6] J. Cui, L. Bai, X. Zhang, Z. Lin, and Q. Liu, "The attention-based autoencoder for network traffic classification with interpretable feature representation," *Symmetry*, vol. 16, p. 589, 2024. [Online]. Available: https://api.semanticscholar.org/CorpusID:269733468

[7] T. Liu, X. Ma, L. Liu, X. Liu, Y. Zhao, N. Hu, and K. Z. Ghafoor, "Lambert: Leveraging attention mechanisms to improve the bert fine-tuning model for encrypted traffic classification," *Mathematics*, vol. 12, no. 11, p. 1624, 2024.

[8] T. Shapira and Y. Shavitt, "Flowpic: Encrypted internet traffic classification is as easy as image recognition," in *IEEE INFOCOM 2019-IEEE conference on computer communications workshops (INFOCOM WKSHPS)*. IEEE, 2019, pp. 680–687.

[9] R. Babaria, S. C. Madanapalli, H. Kumar, and V. Sivaraman, "Flowformers: Transformer-based models for real-time network flow classification," in *2021 17th International Conference on Mobility, Sensing and Networking (MSN)*. IEEE, 2021, pp. 231–238.

[10] L. Yang, A. Finamore, F. Jun, and D. Rossi, "Deep learning and zero-day traffic classification: Lessons learned from a commercial-grade dataset," *IEEE Transactions on Network and Service Management*, vol. 18, no. 4, pp. 4103–4118, 2021.

[11] D. Shamsimukhametov, A. Kurapov, M. Liubogoshchev, and E. Khorov, "Early traffic classification with encrypted clienthello: A multi-country study," *IEEE Access*, vol. 12, pp. 142 979–142 993, 2024.

[12] D. Shamsimukhametov, M. Liubogoshchev, E. Khorov, and I. F. Akyildiz, "Youtube, netflix, web dataset for encrypted traffic classification," 2021, available at https://dx.doi.org/10.21227/s7x7-wd58. [Online]. Available: https://dx.doi.org/10.21227/s7x7-wd58

[13] G. Draper-Gil, A. H. Lashkari, M. S. I. Mamun, and A. A. Ghorbani, "Characterization of encrypted and vpn traffic using time-related," in *Proceedings of the 2nd international conference on information systems security and privacy (ICISSP)*, 2016, pp. 407–414.

[14] R. Wang, Z. Liu, Y. Cai, D. Tang, J. Yang, and Z. Yang, "Benchmark data for mobile app traffic research," in *Proceedings of the 15th EAI international conference on mobile and ubiquitous systems: Computing, networking and services*, 2018, pp. 402–411.

[15] W. Shbair, T. Cholez, J. Francois, and I. Chrisment, "Https websites dataset," http://betternet.lhs.loria.fr/datasets, 2016.

[16] S. Ray, "Advancing packet-level traffic predictions with transformers," Master's thesis, ETH Zurich, Networked Systems, 2022.

[17] Y. Zion, P. Aharon, R. Dubin, A. Dvir, and C. Hajaj, "Enhancing encrypted internet traffic classification through advanced data augmentation techniques," *arXiv preprint arXiv:2407.16539*, 2024.

[18] J. Luxemburk, K. Hynek, T. Čejka, A. Lukačovič, and P. Šiška, "Cesnet-quic22: A large one-month quic network traffic dataset from backbone lines," *Data in Brief*, vol. 46, p. 108888, 2023.

[19] J. Luxemburk and T. Čejka, "Fine-grained tls services classification with reject option," *Computer Networks*, vol. 220, p. 109467, 2023.

[20] Y. Kahana, Y. Stein, and O. Glickman, "Fast real-world classification of ech-enabled applications," 8th International Symposium on Cyber Security Cryptography and Machine Learning (CSCML 2024), Tech. Rep., December 2024. [Online]. Available: https://www.cscml.org/_files/ugd/1ad9fa_8a1961559c3c4a93987df7925d3d3537.pdf

[21] J. Xu, J. Fan, M. H. Ammar, and S. B. Moon, "On the design and performance of prefix-preserving ip traffic trace anonymization," in *Proceedings of the First ACM SIGCOMM Workshop on Internet Measurement*. ACM, 2001.

Yizhak Kahana received his M.Sc. in Computer Science from Bar-Ilan University in 2025 and is currently employed at Allot. His primary research interests include machine learning applications in communications and networking.

Dr. Yaakov Stein received his B.Sc., M.Sc., and Ph.D. degrees in Theoretical Physics from the Hebrew University of Jerusalem. He is currently the CTO at Allot and an Adjunct Professor of Computer Science at Tel Aviv University. He is the author of the textbook "Digital Signal Processing, a Computer Science Perspective." His research interests include network security, communications protocols, and intelligent signal processing.

Oren Glickman holds a B.Sc. in Math and Computer Science (1993) from the Hebrew University Jerusalem, an M.Sc. in Computational Linguistics (1995) from Carnegie Melon University, and a Ph.D. (2006) in Computer Science from Bar-Ilan University. He has over 20 years of industrial experience, primarily in leading research and data science teams. Oren is currently an assistant professor in the Computer Science Dept. at Bar-Ilan University, specializing in natural language processing and applied data science.