Fast Real-World Classification of ECH-enabled Applications

Yizhak Kahana[1,2,*], Yaakov (J) Stein[1], Gabriel Zigelboim[1], and Oren Glickman[2]
[1]*Allot*
[2]*Bar Ilan University*
[*]*correspondence should be addressed to ikahana@allot.com*

December 9, 2024

Abstract

Encrypted Client Hello is plugging information leakage in encrypted packets, in order to thwart Deep Packet Inspection. Various machine learning mechanisms have been proposed to classify ECH network connections, but previous research relied on unlimited classification time and mostly dealt with classifying a small number of classifications in highly constrained settings. We demonstrate highly accurate classification in no more than one second of a large number of applications for a real-network dataset, and propose a novel hierarchical mechanism that leverages protocol-specific knowledge.

1 Introduction

For various reasons, Service Providers need to know characteristics of the network traffic their networks are forwarding. Perhaps the most obvious reason is security, but visibility, QoS monitoring and assurance (including congestion mitigation), policy enforcement, differentiated charging, ensuring QoE levels, fault root cause analysis, user analysis for *upsell* opportunities, meeting regulatory and compliance requirements, and many other use cases are also important. And DPI is conventionally the heart of all systems that need to know characteristics of Internet traffic.

In the Internet protocol (IP) suite, packets are crafted to be self-describing, comprising a sequence of packet headers before the user content (e.g., the encoded audio). 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.

DPI systems comprise carefully crafted rule sets that map logical combinations of metadata to applications. During operation, DPI associates packets to connections (sequences of packets between the same endpoints and sharing identifiers), and parses each individual packet exploiting the self-describing nature of IP packets to extract per-packet metadata. The DPI system then compares metadata collected from packets belonging to a connection to application signatures stored in an application signature rule-set. In this way, state-of-the-art DPI systems are able to recognize thousands of different applications from dozens of application categories, and accurately classify connections after observing a very small number of packets.

A notable (but far from the only) example of unencrypted fields are those in the 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 cyber 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 cyber suite has not yet been agreed upon, 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.

Encrypting metadata is an effective weapon in the battle against pervasive monitoring and revealing private information to malicious parties, but as collateral damage, it renders the job of DPI systems more complicated or even impossible. This directly impacts the aforementioned crucial functions for which DPI is employed, including security, visibility, QoS monitoring and assurance, etc.

However, even encrypting all packet header fields does not truly remove all distinctive metadata [2]. At least three per-packet characteristics remain. First, the sequence of directions of packets comprising the connection (e.g., whether from client to server or from server to client) may be indicative of the service type. For example, video streaming is characterized by a small number of request (GET) packets from client to server followed by a large number of response packets from server back to the client. On the other hand, voice or video calls are on the whole symmetric with approximately equal numbers of packets in each direction. Second, the size of the packets is generally preserved by metadata concealment techniques, or at most modified by the addition of additional header(s). Thus, even a fully encrypted connection can be characterized by a tell-tale sequence of packet sizes. For example, video streaming will consist of a large number of very large response packets from server to client, interspersed with small 'ACK' packets from client to server acknowledging their receipt. Thirdly, packet timings are generally not significantly perturbed by metadata obscuration, although they are influenced by the underlying latencies in the network. From here on we shall use the term PPI (Per Packet Information) to mean the time series of time, size, and direction of the initial packets in a connection.

There is a rich literature on how to use packet size and timing statistics to classify tens of different applications from a small number of application categories, and accurately classify connections after observing a relatively large number of packets.

However, these machine learning methods suffer from at least three drawbacks. First, they necessitate greater computational resources and consume more energy than DPI methods. Second, in contrast to DPI which classifies after observing a few packets, obtaining reliable statistics requires observing large numbers of packets, making these methods unsuitable for many scenarios (e.g., blocking). Third, they are limited to a small number of applications, as compared to the thousands of applications recognized by DPI.

Here we describe an alternative machine learning paradigm that avoids these drawbacks. Instead of statistics we directly use the PPI time series, without needing access to per-packet header field values. Classifying using only the PPI time series is known as *behavioral* classification. Only a short PPI time series, corresponding to a small number of packets, need be used. Furthermore, instead of building a single large classifier, we build a number of small classifiers, each model classifying a subset of applications. Since ECH leaves the IP layer visible, only applications that share the same server address are potentially confused and need to be distinguished using machine learning. These small models require short time series and have extremely high classification accuracy and recall.

2 Related work

In recent years, researchers have proposed using Machine Learning and Neural Network Learning models to deal with encrypted traffic classification. However, many of the conventional methods are based on the byte content of packets of the connection [3, 4, 5, 6]. These models learn patterns in unencrypted metadata, such as SNI, which will not be relevant once ECH is widely adopted. Another drawback in current studies is that the classification task was over a very few number of applications (12-19) [3, 4, 5, 6]. Many articles train and test their models on small limited data sets, old data sets such as the ISCX VPN/non-VPN dataset [7], or network simulations [8].

The main drawback of current methods of behavioral classification is lengthy classification times. FlowPic [9], for example, requires 30-second images and degrades with significantly shorter times. Studies that do attempt to minimize time to classification, tend to classify only a limited number of apps. For example, FlowFormers [10], which use attention-based Transformers to achieve high accuracy classification within 10 seconds, have only been demonstrated on 5-class problems. A recent paper from Huawei France [11] uses the PPI of the first 100 packets for TCP connections and 10 packets for UDP ones (unless the entire connection has fewer packets). Although this paper achieves approximately 90% accuracy for a large

number (the top 200) of applications, the analysis of 100 packets can be time-consuming, and the paper does not impose any time limit.

Despite the limitations mentioned above, new avenues for research are emerging with the availability of more relevant and extensive datasets. For example, the CESNET dataset [12, 13] provides a rich source of encrypted traffic data that has not been sufficiently embraced by other studies.

To the best of our knowledge, no comprehensive large-scale study has been carried out for a large number of applications with strongly limited classification time. As compared to the literature our approach innovates in three areas:

Operational Network Data Training and test data are collected from a large operational mobile network and labeled using a state-of-the-art commercial DPI system.

Real-Time Applicability Our approach supports immediate actions required in operational networks, by limiting PPI duration to 1 second.

Domain Knowledge Rather than blindly throwing a large machine learning model at the connection classification problem, we build a hierarchical structure of smaller classifiers based on clustering of server IP addresses.

3 The dataset format

Our aim is to identify the application supported by a network connection from among a large number of distinct applications and to accomplish this identification within stringent constraints — specifically ensuring that the total time elapsed from the reception of the first packet to the last does not exceed one second. Hence our data consists of network connections, each comprising connection identifiers (the so-called 5-tuple of 2 IP addresses, the layer-4 protocol number, and 2 TCP or UDP port numbers), up to 50 PPIs (but imposing the one-second constraint), and an application identifier (label). In order to preserve user privacy the IP addresses were anonymized using prefix-preserving CryptoPAN [14].

More precisely, for each connection we have the per-connection fields:

- time of first packet (not used for classification)
- obfuscated source IP address (not used for classification)
- obfuscated destination IP address
- layer 4 protocol number (UDP, TCP, other)
- source port number (not used for classification)
- destination port number (443/8443, 80/8080, other)
- AppID, a numeric value (label) identifying the application
- length of the PPI array (maximum of 50 packets)

and for each packet:

- packet arrival time relative to the initial packet in the connection (in milliseconds)
- packet direction (client-to-server or server-to-client)
- packet payload size (bytes in TCP/UDP payload)
- flag byte (see below).

The flag byte merits special attention. For TCP it contains the eight TCP flags, from which we use the SYN flag only for classification. For UDP packets it contains the first byte of the UDP payload, which for QUIC packets contains the 4 unencrypted bits (header format, fixed bit, long packet type field, etc.), and we utilize only the header format bit.

The AppID label is supplied by a state-of-the-art commercial DPI system, as will be discussed in the next subsection.

3.1 Data collection

Our dataset was constructed from live traffic observed in a large mobile network starting in September 2024. We divided the dataset into several subsets, each containing approximately four days of network traffic including both weekdays and weekends in order to ensure variability in network conditions. The first subset was designated for model training, while the rest were reserved for model testing, at several later dates.

Collection was performed by enhancing a commercial real-time Deep Packet Inspection (DPI) system operating in tap (non-inline) mode on an N*100Gbps bidirectional line. This DPI system is relatively fine-grained, distinguishing between sub-applications from the same provider. As an example, Facebook browsing, Facebook video, and Facebook video upload are considered three distinct applications.

The DPI system provides the AppID label based on a rule set which is updated biweekly. This DPI system has approximately 95% coverage (i.e., it does not recognize about 5% of the connections) and an extremely low misclassification rate, far surpassing simplistic labeling methods used in much of the literature, such as relying solely on Server Name Indication (SNI) or server IP addresses.

3.2 The raw data

To enable the reader to better envision our data we will describe the file selected for training. The connections sampled are selected by hashing the client IP address, thus enabling varying the sampling rate. The training data file encompassed 195 hours of traffic and contained 313 million raw connections, implying an average rate of 445 connections per second.

Of these connections

- 146 million (47%) were unencrypted DNS sessions,
- 31 million (10%) were not completely 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 identified connections (194 per sec).

From previous research (based on querying multiple public DNS resolvers) we know that about 80% of popular server IP addresses correspond to essentially a single domain name. Here we observe a long tail of many less popular IP addresses that have a low probability of being hosted alone on an IPv4 address!

This cleaned data set was found to obey the following statistics:

- About 69% of the server IP addresses represent a single application;
  - 18% were observed to host two distinct applications;
  - 6% were observed to host three distinct applications.
- About 52% of the connections map to a server IP with *essentially* a single application (where *essentially* means that 95% of the connections were identified as belonging to this application);
  - 23% map to a server IP address with essentially two distinct applications;
  - 15% map to a server IP address with essentially three distinct applications.

So, about half of the sampled connections collected can be mapped to a unique application purely by the server IP address! Such connections are thus trivially classified and do not require further machine learning mechanisms.

In addition, while we observe more than 750 distinct applications in the training set, many of these are too rare to merit separate classification. We thus lumped together all applications that appeared in less than 0.05% of the connections into a single aggregate label *Other*.

The test sets were collected using the same setup and underwent identical clean-up, but were amassed one week, one month, and two months later.

A slightly restricted version of this dataset can be made available for academic research purposes. Interested parties should contact the corresponding author.

3.3 Applications

Of the 763 applications reported by the DPI in the training set, 115 passed the popularity criterion, namely were shared by at least 0.05% of the connections. Applications that attracted at least 1 % of connections are presented in Table 1.

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	3.03 %	Snapchat	1.27 %
iCloud	2.90 %	Generic CDN	1.19 %
YouTube Browsing	2.22 %	Facebook Chat	1.14 %

Table 1: Most popular applications with their appearance percentages.

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. Analytics and advertisements are what their name implies from many different servers.

Similarly, over 3% of the connections are labeled *Other*, which is not a true application but rather artificial aggregation of all the insufficiently popular ones.

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.

4 Model Training

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, and the value of exploiting domain-specific knowledge.

4.1 Random Forest Classifier

For the Random Forest classifier, it was necessary to transform the variable-length PPI array into fixed-length feature vectors. We did this by zero-padding to the length of the longest sequence in the dataset, denoted as $L_{\max}$. The PPI components (time, direction, size, and flags) were then separated into individual feature vectors of length $L_{\max}$. The padded feature vectors are then concatenated with per-connection features (we used IP protocol number, destination port, and PPI length).

The Random Forest classifier was trained using the following 10 trees with a maximum depth of 25 and a random state of 42 (for reproducibility). The classifier was trained on the entire dataset without a separate validation split.

4.2 LSTM Neural Network

Since the Random Forest classifier does not fully capture temporal dependencies, we trained an LSTM neural network capable of processing variable-length sequences and learning temporal patterns inherent in the packet-level data.

The LSTM-based architecture leverages both per-connection and per-packet information:

1. **per connection Information** protocol number, destination port, PPI length
2. **per packet information** packet time, direction, size, and flags.

The *PPI* is first passed through a *Pack Padded Sequence* operation, to prepare it for processing by the LSTMs layers that handle sequences of varying lengths. The packed sequence is then fed into four stacked **LSTM** layers which capture temporal dependencies within the packet sequences. After processing by the LSTMs, the output is *unpacked*, and the final hidden state corresponding to the *last time step* of each sequence is extracted. This hidden state encapsulates the sequential information from the packet data.

The extracted features from the LSTM are then *concatenated* with the connection information, effectively combining sequential and non-sequential features. This combined feature vector is passed through two **fully connected layers**.

The model outputs class scores corresponding to different application IDs.

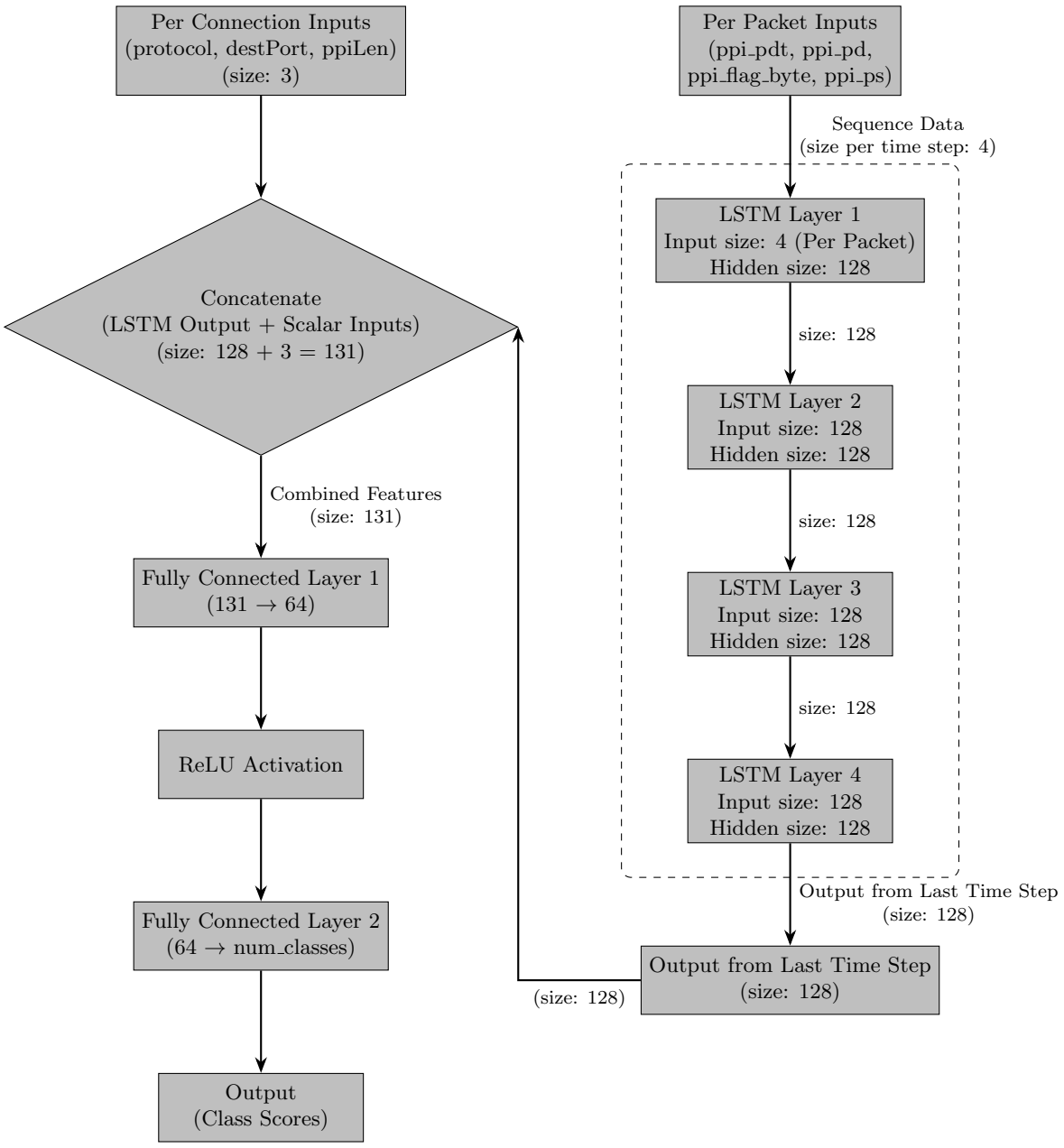

Figure 1: Architecture of the LSTM

The complete LSTM model (see Figure 1) has the following architecture:

- **LSTM Input Layer**: variable length PPI.
- **LSTM network**:
  - Input size: 4 (number of sequence features) times PPI length
  - Number of LSTM layers: 4
  - Number of LSTM units per layer: 128 units
  - Output size: 128
- **Fully Connected Layers**:
  - **Input Layer** per connection features (protocol number, destination port, PPI length).
  - **concatenation layer**
  - fully connected feedforward layer
    - Input size: 131 (128 from LSTM output + 3 scalar features)
    - Output size: 64
  - ReLU activation layer (size: 64)
  - fully connected feedforward layer
    - Input size: 64
    - Output size: C (the number of application classes)

The model was trained using the following settings:

- **Loss Function**: Cross-Entropy Loss, suitable for multi-class classification.
- **Optimizer**: Adam optimizer with an initial learning rate of 1×10^{-3}.
- **Learning Rate Scheduler**: StepLR scheduler reduces the learning rate by a factor of 0.5 every 5 epochs to facilitate convergence.
- **Epochs**: 20 epochs, determined empirically to balance training time and performance.
- **Batch Size**: 128 samples per batch.
- **Regularization**: Gradient clipping with a maximum norm of 5 to prevent exploding gradients.

Model performance was monitored using weighted precision, recall, and F1 metrics on a small validation set.

The model was trained using GPU acceleration.

5 Hierarchical Classification

In the current literature, the predominant method for classifying encrypted network traffic involves constructing a single, monolithic model that attempts to identify all applications. We'll call this the *flat classification* method. For example, in the previously cited Huawei study [11], a single large model is developed to categorize a multitude of applications by analyzing encrypted packet characteristics.

In contrast to the flat method, we introduce a hierarchical approach that incorporates knowledge of the server IP address, which remains unencrypted under ECH. We group network traffic based on server IP address and build unique models that distinguish between the relatively small number of applications specific to each group. This allows integrating domain-specific knowledge directly into the classification process. Instead of relying on a single monolithic model to classify all applications, we construct many smaller specialized models, each focusing on traffic associated with specific server IP addresses. Not only is it easier and faster to train these smaller models, but classification errors are less severe as misclassification is only possible between related applications.

The first step in our hierarchy singles out server IP addresses that were identified in the training set as hosting essentially a single application. A server is deemed to host *essentially* a single application if the

proportion of training connection associated with its dominant application exceeds 95%. For such server IP addresses, classification is trivial.

The next level of our hierarchy locates sets of popular server IP addresses that host multiple applications. We may partition such server IP addresses into distinct groups that share the same applications, using one of three methods:

1. ASN-based grouping
2. application-based grouping
3. application distribution clustering

Connections with server IP addresses not in a defined group are classified using a flat model.

ASN-based grouping maps server IP addresses to their Autonomous System Numbers (ASNs) to leverage the fact that servers within the same Autonomous System are likely to host applications related to the entity owning that AS. This is especially the case where the AS belongs to a *gorilla*, that is, one of the dominant technology giants such as Google, Meta, Amazon, Apple, or Microsoft. Thus, a server IP address belonging to Google's ASN 15169, might host GoogleServices, Google Search, Gmail, Google Maps, Youtube, etc., but would not support Facebook, Instagram, or WhatsApp, which are hosted in Meta's ASN 32934. The disadvantage of this grouping is that a gorilla may subdivide its IP address resources, not hosting all services on all of its addresses.

Application-based grouping partitions servers based on a particular application constituting a significant portion of their activity. All server IP addresses with the same dominant applications are grouped together. Application distribution clustering compares the probability distributions and performs clustering based on the distribution similarity. These grouping methods will be more fully explained in a future publication. The overall performance did not significantly differ depending on the grouping method.

Here we will report on ASN-based grouping, wherein we map popular server IP addresses to their ASN using publicly available information. In our training set we found 11 major ASN groups (each encompassing over 1% 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. For each ASN group we train a separate classification model that classifies only applications found in that group. Connections belonging to server IP addresses that are not in a major ASN group are classified using a flat model trained on the complete dataset comprising all applications.

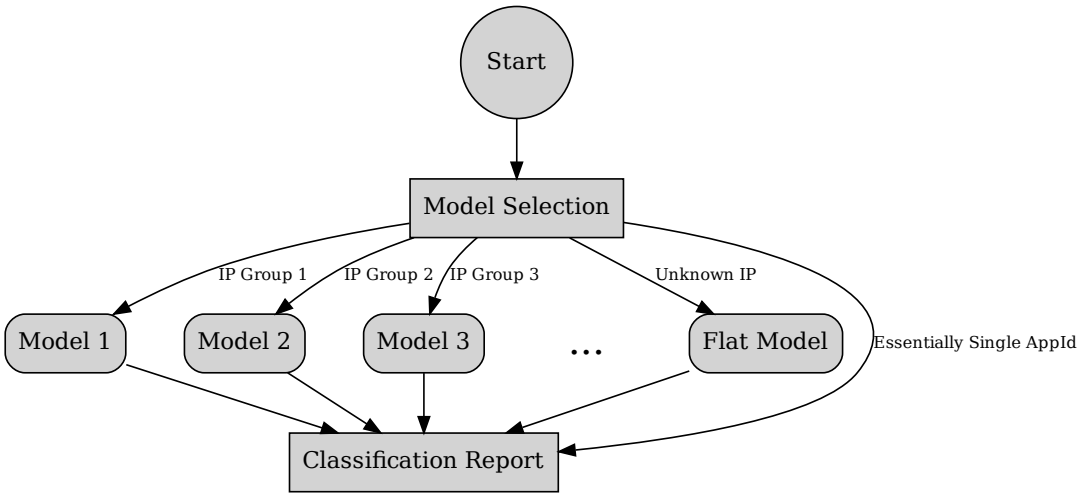

Figure 2: Hierarchical Decision Flow

6 Results

Here compare our hierarchical classification scheme to a flat classifier that does not exploit server IP address at all. We have additionally explored the flat model with server IP address as input; although this improved the model's accuracy, its overall performance was still significantly inferior to that of our proposed hierarchical method.

First, two classifiers, a Random Forest classifier and a deep LSTM, were trained with up to 50 packets per connection, using the flat model and our hierarchical model (with ASN-based grouping). The performance on a test set collected a week after the training are presented in Table 2.

In order to reduce the time to classification, we explored the same methods but with a stricter limit of no more than 10 packets per connection. The performance metrics suffered, but not very significantly, as shown in Table 3.

In a rapidly evolving network landscape, models trained on data from a specific time frame may experience performance degradation when applied to data collected at a later date. Factors such as updates to existing services and changes in user behavior can alter the statistical properties of network traffic, leading to distributional drift.

To assess the time stability of model we conducted a drift evaluation using two additional datasets collected a month later and two months later. Results are presented in Tables 4, 5, 6 and 7. We observe that our hierarchical model maintained close to 90% accuracy, recall, and F1-score, even after two months. Assessment of geographical stability will be reported elsewhere.

7 Discussion

The introduction of ECH strongly impacts DPI systems, and requires the introduction of more sophisticated machine learning techniques. Previous research has mostly ignored the real-world issues of time-to-classification (allowing 30 seconds or a minute before classifying) and the large number of potential applications (typically assuming tens of applications suffice).

We have presented what we believe to be the first research into replacing DPI in real-world settings to counter the influence of ECH. We find that, based solely on packet size, direction and time of arrival, we can classify over 100 applications in less than 1 second, with accuracy of approximately 90%. Indeed, even a mere ten packets suffice for high-accuracy classification. We find little advantage to using deep LSTM classifiers as compared to legacy Random Forest ones.

We furthermore proposed a hierarchical model that integrates domain-specific knowledge through the utilization of server IP addresses, which are not hidden in the ECH scenario. This hierarchical approach not only reduces computational complexity of training and inference, but noticeably improves performance as compared to the flat model. Furthermore, any residual errors are limited to within server IP groups, which are less impactful in practice.

Our methods demonstrate robust generalization when tested on a real-world mobile network, and do not degrade significantly after a month or even two months have gone by.

Table 2: Performance Comparison with up to 50 PPIs

Classification Method	Model	Precision	Recall	F1 Score
Flat Model	Random Forest	88.42	88.26	88.07
	LSTM	90.06	89.93	89.84
ASN-Based Grouping	Random Forest	93.58	93.54	93.42
	LSTM	94.68	94.64	94.57

Table 3: Performance Comparison with up to 10 PPIs

Classification Method	Model	Precision	Recall	F1 Score
Flat Model	Random Forest	88.15	88.11	87.92
	LSTM	86.34	86.35	86.05
ASN-Based Grouping	Random Forest	92.71	92.71	92.58
	LSTM	92.15	92.12	91.97

Table 4: Performance Comparison with up to 50 PPIs one month after training

Classification Method	Model	Precision	Recall	F1 Score
Flat Model	Random Forest	86.62	86.21	85.95
	LSTM	88.16	87.81	87.64
ASN-Based Grouping	Random Forest	92.16	92.01	91.85
	LSTM	93.13	93.05	92.93

Table 5: Performance Comparison with up to 10 PPIs one month after training

Classification Method	Model	Precision	Recall	F1 Score
Flat Model	Random Forest	86.21	86.00	85.70
	LSTM	84.19	84.11	83.65
ASN-Based Grouping	Random Forest	91.10	91.05	90.85
	LSTM	90.25	90.23	89.98

Table 6: Performance Comparison with up to 50 PPIs two months after model training

Classification Method	Model	Precision	Recall	F1 Score
Flat Model	Random Forest	82.03	80.79	80.38
	LSTM	83.19	82.22	81.90
ASN-Based Grouping	Random Forest	90.48	90.31	90.06
	LSTM	91.27	91.17	90.98

Table 7: Performance Comparison with up to 10 PPIs two months after training

Classification Method	Model	Precision	Recall	F1 Score
Flat Model	Random Forest	81.43	80.34	79.95
	LSTM	79.25	78.15	77.59
ASN-Based Grouping	Random Forest	89.63	89.58	89.35
	LSTM	88.75	88.69	88.43

References

[1] Eric Rescorla, Kazuho Oku, Nick Sullivan, and Christopher A. Wood. TLS Encrypted Client Hello. Internet-Draft draft-ietf-tls-esni-22, Internet Engineering Task Force, September 2024. Work in Progress.

[2] Martino Trevisan, Francesca Soro, Marco Mellia, Idilio Drago, and Ricardo Morla. Attacking doh and ech: Does server name encryption protect users' privacy? *ACM Transactions on Internet Technology*, 23(1):1–22, 2023.

[3] Weiping Zheng, Jianhao Zhong, Qizhi Zhang, and Gansen Zhao. Mtt: an efficient model for encrypted network traffic classification using multi-task transformer. *Applied Intelligence*, 52(9):10741–10756, 2022.

[4] Hong Ye He, Zhi Guo Yang, and Xiang Ning Chen. Pert: Payload encoding representation from transformer for encrypted traffic classification. In *2020 ITU Kaleidoscope: Industry-Driven Digital Transformation (ITU K)*, pages 1–8. IEEE, 2020.

[5] Peng Lin, Yishen Hu, Yanying Lin, Kejiang Ye, and Cheng-Zhong 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)*, pages 267–274. IEEE, 2021.

[6] Jee-Tae Park, Chang-Yui Shin, Ui-Jun Baek, and Myung-Sup Kim. Fast and accurate multi-task learning for encrypted network traffic classification. *Applied Sciences*, 14(7):3073, 2024.

[7] Gerard Draper-Gil, Arash Habibi Lashkari, Mohammad Saiful Islam Mamun, and Ali 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)*, pages 407–414, 2016.

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

[9] Tal Shapira and Yuval Shavitt. Flowpic: Encrypted internet traffic classification is as easy as image recognition. In *IEEE INFOCOM 2019-IEEE conference on computer communications workshops (INFOCOM WKSHPS)*, pages 680–687. IEEE, 2019.

[10] Rushi Babaria, Sharat Chandra Madanapalli, Himal Kumar, and Vijay Sivaraman. Flowformers: Transformer-based models for real-time network flow classification. In *2021 17th International Conference on Mobility, Sensing and Networking (MSN)*, pages 231–238. IEEE, 2021.

[11] Lixuan Yang, Alessandro Finamore, Feng Jun, and Dario Rossi. Deep learning and zero-day traffic classification: Lessons learned from a commercial-grade dataset. *IEEE Transactions on Network and Service Management*, 18(4):4103–4118, 2021.

[12] Jan Luxemburk, Karel Hynek, Tomáš Čejka, Andrej Lukačovič, and Pavel Šiška. Cesnet-quic22: A large one-month quic network traffic dataset from backbone lines. *Data in Brief*, 46:108888, 2023.

[13] Jan Luxemburk and Tomáš Čejka. Fine-grained tls services classification with reject option. *Computer Networks*, 220:109467, 2023.

[14] Jun Xu, Jinliang Fan, Mostafa H. Ammar, and Sue 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.