Deep Packet Inspection at Scale: Search Optimization Through Locality-Sensitive Hashing

1st Maya Kapoor
Defense & Intelligence Sector
Parsons Corporation
Charlotte, USA
maya.kapoor@parsons.com

2nd Siddharth Krishnan
College of Computing & Informatics
UNC Charlotte
Charlotte, USA
skrishnan@uncc.edu

3rd Thomas Moyer
College of Computing & Informatics
UNC Charlotte
Charlotte, USA
tmoyer2@uncc.edu

***Abstract*—Deep packet inspection is a primary tool for security specialists, surveillance analysts, and network engineers to lawfully intercept and analyze network traffic. In order to process this data or select streams of interest from the large amount of data flowing in today's internet, solutions must be capable of identifying network traffic as quickly and accurately as possible. The ever-increasing diversity of data as well as sheer size has rendered the current regular expression matching and filtering solutions ineffective. We propose locality-sensitive hash embedding techniques ALPINE and PALM for packet analysis. The fixed size of hashes as well as the adaptability of distance measures is proven to address the network traffic classification problem in our experiments and improves scalability over current state-of-the-art, automata-based search engines. In this paper, we analyze the system's ability to classify network traffic by many data layer protocols and traffic types with over 99% accuracy. The model is also proven effective in areas where the regular expressions are inapplicable, such as traffic profiling. Finally, we provide real benchmarks of the system's ability to scale to large signature and hash sets with much improved performance, demonstrating real-world applicability and generalizability of locality-sensitive hashing to deep packet inspection technology.**

***Index Terms*—Deep Packet Inspection, Data Mining, Locality Sensitive Hashing, Network Forensics**

I. INTRODUCTION

Forensic and real-time classification of network traffic to particular applications, protocols, content types, user profiles, and other identities provides vital information to many communities across the Internet. For law enforcement agencies, network packets contain information which can be routed back to crimes and cyber threats, and big data analytics can help reconstruct these traces into useful evidence [1]. The intelligence community can utilize traffic flowing between user devices, or across servers and hosts that are part of a foreign supply chain, or formulate further analytics capability on computer and social networks. In cybersecurity, deep packet inspection is used to detect and prevent intrusions. Analyzing packets at the data layer is used in real networks today as a method of detecting cyber attacks or malware embedded in content. Furthermore, traffic statistics and anomaly detection can be used to discover and mitigate Denial-of-Service attacks. One network forensic analysis tool for traffic profiling which is used in a variety of application domains is a deep packet inspection (DPI) middleware box. These devices inspect network traffic flows and can passively intercept, re-route, or collect information from network traffic. These devices must be capable of identifying traffic on a network and performing any necessary parsing, additional processing, duplication, and/or re-direction with minimal extra latency.

979-8-3503-9730-7/22/$31.00 ©2022 IEEE

The current industry standard for application layer DPI is to use regular expressions to match expected values in payloads. Protocols or applications may have multiple signatures to match request, response, or message types. Some protocols also do not have strong signatures. Furthermore, different versions of protocols or updated standards and RFCs often require separate signature sets and the continual update of older patterns [2]. Some protocols are also so non-standard that it is difficult to track them on a regular expression signature alone; for example, the payload of an RTP packet is raw data and assigned ports are dynamic [3]. Current state-of-the-art scanners suffer from variable and data-dependent performance impact [4]. With the increasing size and complexity of rulesets and the growing scale of the network environment, regular expression scanning using the current systems is more and more impractical. Regex methods are also limited by what classes they can be used for in network traffic. For example, there is no real regular expression signature which can identify Tor-routed traffic from non-Tor-routed traffic; the data is too diverse. Other desirable classes for traffic, such as traffic type, application, and user or device type present similar issues to regular expression scanners. Each possiblity must have its own, human-engineered regular expression to match precisely which is both slow and prone to error. In addition to scalability and machine performance, text-based regular expression matching no longer meets the needs of the environment as over 90% of browsing data today is encrypted [5].

In order to meet these challenges and advance the current state of the art, we propose a combinatorial, new deep packet inspection tool: ALPINE, **A** **L**ocality-Sensitive **P**acket **In**spection **E**ngine, and PALM, **P**ayload **A**nalysis using **L**ocality-Sensitive **M**easurements. These systems work together using locality-sensitive hashing techniques on packet data in order to accurately classify network traffic in multiple ways such as traffic type or application-layer protocol. The

complexity of our system outperforms state-of-the-art regular expression matching software while maintaining high accuracy and can also classify encrypted traffic in the tested scenarios, improving analysts' capability to perform the necessary packet inspection.

II. Background

Deep packet inspection is the process of analyzing traffic data as it comes across the network. Packet headers contain information which can tell what type of application layer data is being transported in the packet.

1) Port-based Detection: Port numbers are assigned by the Internet Assign Number Authority (IANA) [6] and originally were used to identify the application layer protocol payload content belonged to. In modern architectures that employ nonstandard port usage and translation, classification by port number can be inaccurate. Especially in tunneled environments, network address port translation (NAPT) can be used to map applications to unused port numbers which do not match the standard assignment [7].

2) Signature-based Detection: In order to more accurately determine what type of traffic is flowing through the network, industry standard has evolved past the header to search for signatures in packet payloads. For example, the regular expression signature *^(\+ok .*pop)* can be used to match POP3 payloads [8]. While widely used for text scanning, regular expression matching requires the construction of complex data structures such as deterministic and/or non-deterministic finite automata (DFA/NFA) and can be both memory and computation-intensive depending on the size of the regular expression dictionary and the complexity of the signatures themselves. DFAs are fast to search, but are prone to state explosion as every possibility is explicity constructed. NFAs provide linear storage space to the size of the regular expression ruleset, but it is relatively easy to arrive at worst-case search performance with rule complexity [9]. Table I shows the processing complexity and storage costs of these structures where m is a number of regular expressions of length n.

TABLE I
Worst-Case Space and Time Complexities for NFA and DFA [10]

Data Structure	Complexity	Storage Cost
NFA	$O(n^2m)$	$O(nm)$
DFA	$O(1)$	$O(\Sigma^{nm})$

Recently, hybrid NFA/DFA engines have been proposed to improve search performance; however, improvements on computational complexity can still be prone to memory issues in the case of large regular expression rulesets [11]. Parallel computing may alleviate some of the computational stress, but is acknowledged as a brute force approach [9]. Another research innovation in regular expression matching aims to automatically generate regular expressions via encoding mechanisms and unsupervised learning of data. For example, SigBox used Apriori tokenization to find common n-grams across packets [12]. RExACtor uses both Apriori and genetic sequencing alignment to encode payloads into regular expression [2]. Other works like AutoSig [13], LASER [14] and VinothGeorge et al [8] use similar automation approaches. These regular expressions are still subject to the performance limitations of regular expression scanners, including what they can detect signatures for.

3) Stream-based Detection: In addition to per-packet solutions, there is also an abundance of research in traffic classification using machine-learning techniques that study time and flow-based features of packet streams [15]–[21]; however, in real-time network engineers may not be able to wait for or buffer entire packet streams before they can classify packets and send them to follow-on processing. In forensics, analysts may not have access to full streams. Many machine learning methods are not applicable for stateless packet inspection as they require time-based or flow-based features. Thus, per-packet solutions for both encrypted and non-encrypted data are a pressing research need regardless of flow-based solutions.

Locality Sensitive Hashing

Locality-sensitive hashing uses a distance metric to preserve the similarity of original inputs when they are converted to locality-sensitive hashes (LSH). The benefit of locality-sensitive hashing here is that it is capable of reducing high-dimensionality data into a low-dimensional, space-effective, computationally efficient representation of the same data while still preserving similarity metrics between the data points [22]. Hash families may be defined for many distance measures such as Hamming distance, L1 and L2 norm [23], and Jaccard similarity [24]. One way in which locality-sensitive hashing has been applied in the network security domain is in user-level browser fingerprinting. In this method, web-based browser fingerprints created by extracting multiple values from the browser API may be hashed using MinHash or similar functions to generate signatures of high entropy, where the data is uniquely identifiable as a particular device or host [25]. We apply this concept similarly to network packet data, aiming for a locality-sensitive hash using features which should also evidence values unique to the class type. LSH clustering has also been used to create signatures for identifying malware at scale [26], but has rarely been applied to the DPI problem.

Similarity Search

Similarity search is a generalized term in data mining which refers to searching for objects where the available comparator is some common pattern or similarities among them. Examples of similarity search mechanisms include Nearest Neighbors searching, link-based similarity searches, duplicate detection, and defining object representations [27]. Applications for this in the cyber crime space include finding individuals in the same criminal network, finding content online similar to known evidence, querying to find images similar to another, and detecting fingerprints across network data. The best indexes for similarity search have efficient querying and storage mechanisms, minimum memory footprint, and minimal interference required by the developer and maintainer [28].

One of the challenges in data engineering is determining how to construct suitable, unique *tokens* which characterize the data meaningfully. The sentence, "Palm trees are native to the Pacific", may be split on whitespace into the set of tokens $T = \{$"palm", "trees", "are", "native", "to", "the", "pacific"$\}$. It is obvious that tokens like "the" are far too generic in the English language to be characteristic of this sentence, but a token like "palm" or "pacific" will be much more unique and indicate a stronger similarity. This idea may be more solidly quantified through taking the inverse document frequency (IDF) of tokens which minimizes the importance of terms which appear frequently in the document set. When combined with term frequency (TF) as the TF-IDF value [29], the most relevant tokens may be found which are both common and characteristically unique of the document set.

$$IDF(t, D) = log\left(\frac{N}{count(d \in D : t \in d)}\right) \tag{1}$$

TF-IDF is typically used to convert texts into vector representations whose cosine similarity may be computed and compared. The TF-IDF model has been used to successfully distinguish between benign data and malicious network traffic carrying worms, executing DoS attacks, and spreading spam and nefarious content [30], [31]. The method has also been used to distinguish darknet traffic [32], [33].

III. Related Work

The ALPINE strategy was originally inspired by locality-sensitive, hash-based recommendation systems used in document duplication detection and search. The Jaccard similarity of sets is commonly used to recommend products or next steps in a workflow, or detect plagiarism, or make predictions. These sets must be made up of individual items, often tokens or shingles of strings as done in SigBox [12] and RExACtor [2]. Both these works use the tokens as part of regular expression generation.

A limited amount of work has been done in applying locality-sensitive hashing in order to generate application fingerprints. Tang et al proposed HSLF [34], an HTTP header sequence-based LSH fingerprint generator for classifying applications in HTTP traffic. While this work is limited to only cleartext HTTP traffic, results show the ability of their SimHash-based method to accurately distinguish between data such as Firefox, VMWare, and WeChat. Naturally, this work has limitations due to encryption and the scope of traffic which only makes up a fraction of real-world data.

Jiang and Gokhale [35] show the ability of locality-sensitive hashing to accurately classify network traffic in their research using packet-level features exclusively in order to achieve stateless packet inspection. Their research focused solely on multimedia applications versus legacy web-browsing; they did not expand into encapsulation architectures and more traffic classes as well as application-level identification. They also use K-Nearest Neighbors clustering for classification, which is known to scale sub-optimally. We expand these works into a much more complex and diverse set of experiments and applications.

IV. Contributions

As a forward-thinking approach, we propose PALM and ALPINE. PALM identifies packets based on their payload contents using locality-sensitive hashing. It provides a more data-independent approach to searching and a more bounded performance impact. ALPINE also performs locality-sensitive hashing with the same optimizations over header features, making it a reasonable approach even for encrypted traffic. It requires less human-engineering expertise to configure and can be adapted to a myriad of network packet problems. These methods provide the following contributions to the state of the art:

- A process for generating multiple locality-sensitive hash embeddings from packets which is highly accurate for identification of many classes, including protocol type, traffic type, application, and more,
- A generalizable framework which applies to many network traffic classification problems, and whose model can be quickly trained and adapted to suit new problems,
- An alternative to regular expression scanning for DPI which scales sublinearly and requires a linear amount of storage space,
- A diverse and unique application of the traffic classification problem with experiments classifying multiple classes of protocols across many traffic types,
- and public datasets and source code for experimental reproducibility.

V. System Design

In order to reduce packet data to a set intersection problem, contents must first be *shingled* into items which will make up the set. An object is w-shingled when a sub-sequence of length w of contiguous tokens is cut from it. In ALPINE, we use the port information, transport layer protocol, flag values, and packet length to make up our shingles. In PALM, word tokens are created by delimiting packet payloads by whitespace. Thus, each original packet P is now associated with some set S_P of shingles as depicted in Figure 1.

$$J(P_A, P_B) = \frac{|S_A \cap S_B|}{|S_A \cup S_B|} \tag{2}$$

Fig. 1. Jaccard Similarity of Shingle Sets

MinHash may be used to quickly approximate the Jaccard similarity between two sets. The hash signatures take up a fixed and smaller amount of space than large shingle sets and can be used to normalize the data in cases where the shingle set features are of varying lengths and types. The MinHash of a given column is the number of the first row in permuted order whose characteristic matrix value is 1. This sequence is continued down the columns and repeated for k permutations. The probability that the MinHash function for a

random permutation of rows produces the same value for two sets is a close approximation to their Jaccard similarity [36]. We use the LSHForest algorithm developed by Bawa et al [27] to index the locality-sensitive hashes and optimize the search process. It improves upon nearest-neighbor searches by both reducing complexity and eliminating the need to know the distance r from the query to the nearest neighbor of a given data point. This optimization is achieved through the use of a specific set of hash functions which ensures that any nearest neighbor query returns ϵ-approximate neighbors so long as a suitable l number of trees is chosen. Compared to previous work which uses nearest neighbor searches [35], LSHForest scales much more efficiently. The build time of a KNN model can have a complexity of $O(n^2)$, where n is the number of items. In contrast, LSH construction is done in linear time [27]. The theoretical complexities of operations on LSHForest are provided from the original paper in Table II. When compared with time and space complexities for the automata in regular expression searching in Table I, the logarithmic improvement and fixed storage space is clear.

TABLE II
TIME COMPLEXITIES FOR LSHFOREST [27]

Operation	Complexity
Insertion	$O(l * log_B n)$
Deletion	$O(l * log_B n)$
Query	$O(l * log_B n) + O(l * logB) + O(M/B)$

In these equations, l represents the number of trees, B the branching factor of an internal node, and n the number of data points in the dataset. Storage is also optimized to be linear, $O(n)$, through the use of compressed PATRICIA tries [27]. The query for some point q for m nearest neighbors first performs a binary search for the longest matching prefix at a leaf node with the point s_i. Then some M points are collected synchronously across all trees and ranked by similarity score, with the top m being returned as an answer. We use the m number as votes to ultimately classify the sample by majority once the query is performed. This majority vote approach also allows for multiple classification results; for example, an adaptive system may be able to take the second closest result if the first choice turns out to be erroneous.

VI. DATASETS

Instead of using a single dataset or network environment to evaluate the system, we used several available public datasets. Our sources include the CDX 2009 captures [37], the Skynet Tor dataset [38], the Canadian Institute for Cybersecurity's ISCX VPN/non-VPN and Tor/non-Tor datasets [39], [40], and repositories from Wireshark, Cloudshark, and IEEE Dataport. Investigation has shown that using captures from only a single environment can lead to bias in results [41]. A machine learning algorithm might learn characteristics of the network or environment such as IP addresses rather than actual protocol features, which can hurt generalizability of the model. For this reason and to increase the scope of classes beyond what is available in any single public dataset, we merged PCAPs from these experiments and created our own repository of twenty-six different, common application layer protocol packet captures. For experimental reproducibility and general use we have made this new dataset available publicly [1].

TABLE III
DESCRIPTION OF COMBINED DATASET

Application Type	Protocol	# Samples
File Transfer	FTP	10,015
	FTP-DATA	4,000
	BitTorrent	20,648
Voice-over-IP	MGCP	1,568
	SIP	1,112
	H.225	1,300
	RTP	15,552
	RTCP	1,626
Mail	SMTP	5,981
	POP	1,675
	IMAP	3,318
Authentication & Access	LDAP	1,354
	SMB	3,554
	Telnet	1,888
Tunneling	GPRS	9,981
	PPTP	1,288
	SSH	3,039
Web & Chat	DNS	10,563
	HTTP	21,298
	XMPP	1,553
	TLS	4000
Networking	DHCP	1,444
	NBNS	1,216
	GQUIC	1,740
	NTP	1,940
	SSDP	8,504

VII. RESULTS

We posed the following research questions and performed related experiments in order to assess the effectivity of the ALPINE and PALM models:

- **R1.** Given labeled training data, can packets be classified by their application-layer protocol through the construction and index of locality-sensitive hashes?
- **R2.** Can we also distinguish traffic types (such as email or VoIP data) from one another using this model? Does it generalize to other kinds of network traffic classes besides protocol?
- **R3.** For payload analysis, does using only high-value tokens as characterized by TF-IDF score improve distance measure results by isolating important features and reducing noise effectively in the data?
- **R4.** Does the combination of the ALPINE and PALM models, i.e. multiple hash embeddings and concatenating those results, improve classification performance over any single hash embedding of a feature set?
- **R5.** Does the number of classes, hashes, or sample size affect throughput or accuracy significantly?

[1] https://github.com/mayakapoor/protocol-dataset

TABLE IV
PROTOCOL AUTODETECTION RESULTS

	Jaccard			TF-IDF		
Accuracy	0.996			0.841		
Cohen's κ	0.997			0.841		
Protocol	**P**	**R**	**F1**	**P**	**R**	**F1**
BitTorrent	1.00	0.99	0.99	0.89	0.85	0.86
DHCP	1.00	1.00	1.00	0.91	1.00	0.95
DNS	1.00	0.99	1.00	0.95	0.97	0.96
FTP	0.99	1.00	0.99	0.83	0.90	0.86
FTP-DATA	1.00	1.00	1.00	0.93	0.96	0.94
GPRS	1.00	1.00	1.00	0.91	0.98	0.95
GQUIC	1.00	0.99	0.99	0.87	0.75	0.81
HTTP	1.00	0.99	0.99	0.82	0.48	0.61
H.225	1.00	1.00	1.00	0.83	0.97	0.90
IMAP	1.00	0.99	0.99	0.86	0.79	0.82
LDAP	1.00	0.99	0.99	0.87	1.00	0.93
MGCP	1.00	1.00	1.00	0.90	0.65	0.75
NBNS	0.99	1.00	0.99	0.94	1.00	0.97
NTP	1.00	1.00	1.00	0.91	1.00	0.95
POP	0.98	1.00	0.99	0.91	0.47	0.62
PPTP	0.95	1.00	0.98	0.84	0.90	0.87
RTCP	1.00	1.00	1.00	0.96	1.00	0.98
RTP	1.00	1.00	1.00	0.89	0.88	0.89
SIP	1.00	1.00	1.00	0.54	0.98	0.69
SMB	1.00	0.97	0.98	0.95	0.99	0.97
SMTP	1.00	0.99	0.99	0.92	0.93	0.93
SSDP	1.00	1.00	1.00	0.72	0.28	0.40
SSH	1.00	0.98	0.99	0.90	0.86	0.88
Telnet	0.99	1.00	0.99	0.86	0.99	0.92
TLS	1.00	1.00	1.00	1.00	1.00	1.00
XMPP	1.00	1.00	1.00	0.79	0.90	0.84

- **R6.** Does our system outperform state-of-the-art regular expression scanning in terms of memory usage and testing/training time? What is the rate of scalability?

For all classification experiments, we record traditional measures of precision, recall, and F1-score. We also measure Cohen's kappa (κ) which describes how often multiple raters agree on a vote in ensemble classification adjusted for the times they agree by chance. In Equation 4, the value p_o is the relative observed agreement, and p_e is the hypothetical probability that they agree by chance.

$$k = \frac{p_o - p_e}{1 - p_e} \tag{3}$$

For systems experiments, we measure the throughput by taking into account the number of Megabits per second of packets the system is capable of processing. Memory usage during testing is also profiled by mebibytes using Python's native memory profiler. We also record the milliseconds per classification and training/test times for varying sample sizes.

A. Protocol Auto Detection

The autodetection of application-layer protocols is a crucial functionality for lawful interception devices like DPI middleware boxes in order to properly assess, parse, and process the traffic they are checking. The traffic environment on any given signal can be wildly diverse; thus, any classification system must be capable of scaling to a large, multiclass problem. Previous systems [34], [35] have not attempted protocol detection to the scale of a 26-class problem, making this experiment and our model a true and fresh pioneer in the DPI field. To address the first question, we split the data into training and test sets and extracted the features from each of the protocols. We generated hash values for the training samples, adding them to the classifier. The results of the testing samples are provided in Table IV. The evidence shows that the combined model has high accuracy and precision/recall for protocol identifications. Cohen's kappa also shows a strong agreement among the ensemble voters. The method also performs equally well across application types, indicating generalizability. Figure 3 shows the confusion matrix results of the classifier's ensemble voting system.

B. Classification by Traffic Type

RFCs and standardized documentation necessitate that there are some detectable similarities between protocols. It is reasonable to think that detection of more abstract class types like traffic type (i.e. email, file transfer, web data) would be more heterogeneous and thus more difficult to embed. We experimented with classification by traffic type by amalgamating the dataset into classes as described in Table III in the datasets section of this work. Results in Figure 2 show a confusion matrix of the classification results which are also highly accurate.

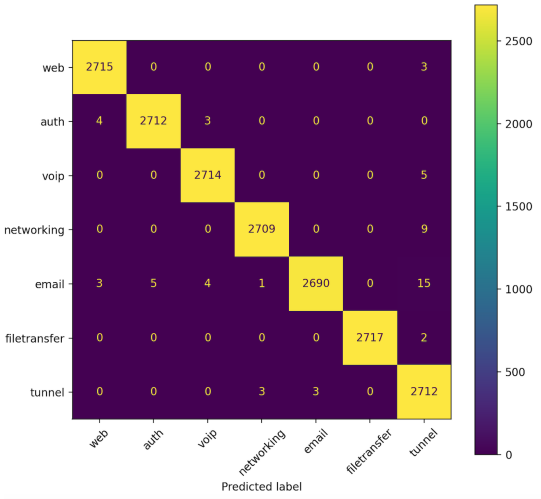

Fig. 2. Confusion matrix of traffic type classification results.

C. TF-IDF Score Evaluation

In order to determine the effectivity of Jaccard similarity versus more complex similarity metrics, we evaluated the results of using payload tokens which were considered *high-value* by TF-IDF score. We posed this question to address concern that large data payloads, for example HTML documents or email messages, may introduce noise into hashes which would reduce classification accuracy. Instead, isolating payload tokens to only the high value ones could reduce spurious extra data. In order to test this, we implemented a

TABLE V
MULTI-HASH EMBEDDING CLASSIFICATION RESULTS

	ALPINE			PALM			Multi-Model		
Accuracy	0.988			0.717			0.996		
Cohen's κ	0.988			0.705			0.997		
Protocol	**F1**	**P**	**R**	**F1**	**P**	**R**	**F1**	**P**	**R**
BitTorrent	0.98	0.97	1.00	0.74	0.75	0.73	0.99	1.00	0.99
DHCP	1.00	1.00	1.00	0.91	0.83	1.00	1.00	1.00	1.00
DNS	1.00	1.00	1.00	0.24	0.46	0.17	1.00	1.00	0.99
FTP	0.96	0.97	0.94	0.90	0.83	0.98	0.99	0.99	1.00
FTPDATA	1.00	1.00	1.00	0.20	0.37	0.13	1.00	1.00	1.00
GPRS	1.00	1.00	1.00	0.91	0.98	0.85	1.00	1.00	1.00
GQUIC	0.99	0.99	1.00	0.27	0.48	0.19	0.99	1.00	0.98
HTTP	0.99	1.00	1.00	0.61	0.65	0.55	1.00	1.00	0.99
H.225	1.00	1.00	1.00	0.63	0.68	0.95	1.00	1.00	0.99
IMAP	0.99	0.94	0.98	0.85	0.77	0.59	0.99	1.00	0.99
LDAP	0.96	1.00	0.98	0.66	0.75	0.59	0.99	1.00	0.99
MGCP	1.00	1.00	1.00	0.90	0.83	1.00	1.00	1.00	1.00
NBNS	0.99	0.98	1.00	0.87	0.82	0.92	0.99	0.99	1.00
NTP	1.00	1.00	1.00	0.90	0.82	0.99	1.00	1.00	1.00
POP	0.99	0.98	0.99	0.26	0.41	0.20	0.99	0.98	1.00
PPTP	0.99	0.98	0.99	0.49	0.33	1.00	0.98	0.95	1.00
RTCP	1.00	1.00	1.00	0.92	0.85	1.00	1.00	1.00	1.00
RTP	0.99	0.98	0.99	0.40	0.65	0.29	1.00	1.00	1.00
SIP	1.00	1.00	1.00	0.93	0.87	1.00	1.00	1.00	1.00
SMB	0.99	1.00	0.98	0.81	0.80	0.82	0.98	1.00	0.97
SMTP	0.98	0.99	0.98	0.81	0.80	0.82	0.99	1.00	0.99
SSDP	0.99	1.00	0.98	0.92	0.86	1.00	1.00	1.00	1.00
SSH	0.95	0.97	0.92	0.47	0.67	0.37	0.99	1.00	0.98
Telnet	0.99	0.98	1.00	0.80	0.75	0.85	0.99	0.99	1.00
TLS	0.99	0.99	1.00	0.61	0.51	0.75	1.00	1.00	1.00
XMPP	1.00	1.00	0.99	0.92	1.00	0.85	1.00	1.00	1.00

TF-IDF measurement of tokens in the training stage. This was done by modifying configurations of the PALM model to filter the added tokens. When training and test signatures were generated, only those tokens with TF-IDF score above the minimum threshold were kept and included in the actual LSH signature. The results in Table IV show that this actually had a negative impact on the overall result, indicating the basic Jaccard similarity was a better distance metric for this use case.

D. LSH Multi-Embeddings

The concatenation of multiple classifiers and agreeance among voters in an ensemble has yielded a reduction in error and at best correlation when there is a misclassification [42]. We hypothesized that enabling both ALPINE and PALM to run together would yield the most accurate and agreed-upon classification results. In order to test this, we ran the 26-class protocol identification test against configurations of the model with just ALPINE embedding, only PALM embedding, and a final combined run. Our results as shown in Table V demonstrate that the combined votes of both models turned out to be the most accurate in the final tests. Furthermore, the agreeance among voters was also highest using multiple hash embeddings.

E. Model Performance and Throughput

One critical requirement for applied machine learning in network processing is that systems must keep up with signal processing at very high rates. Any passive system must not interfere with legitimate traffic and service. Furthermore, the software must be capable of performing the necessary processing on as much of the traffic as possible (ideally, all of it). While thinning and load-balancing solutions as well as diverting and copying traffic for offline analysis can help levy this concern, there is still the desire to employ classification solutions at real-time rates. Thus, we perform experiments to see how the system scales based on model sizes and number of classes. We measured the training time, system throughput in millibits per second (Mbps) as well as memory usage during the testing phase in megibytes (MiB). We ran all performance tests on the combined dataset consisting of 140,157 total packets. In order to avoid bias we implemented random under-sampling to even out the distribution of data, causing our total number of packets after sampling to be a fraction from the original set. For testing the number of classes, we created experiments with binary RTP/non-RTP classification, the traffic-type multiclass experiment with 8 classes/types, and the largest 26-class experiment where we identify all possible data layer protocols. In Table VII, we detail the results of the experiments for a varied number of classes and sample sizes.

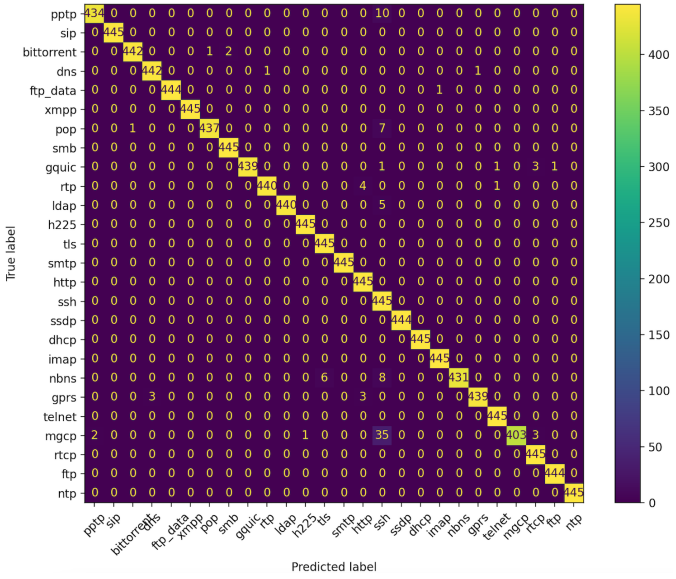

Fig. 3. Twenty-six class identification problem using the PALM-ALPINE combined model.

F. Comparison to Regular Expression Matching

In order to test system scalability indeployment and verify the theoretical complexity discussed in the background section of this work, we performed a series of experiments increasing the number of signatures for a model used to classify HTTP traffic. As a comparator, we wrote a DPI sniffer application in C++ using using the Hyperscan 5.4.0 regular expression matching library by Intel [11]. All experiments were performed on a single i9 Dell computer with Ubuntu 18.04 installed. In the trials, the number of signatures represented the number of LSH hashes our system or the number of

TABLE VI
PERFORMANCE RESULTS FOR VARIED NUMBER OF CLASSES

Classes	# Hashes	# Test Samples	Training Time	Test Time	Memory Usage	Throughput	Accuracy
RTP/non-RTP	20,613	13,743	4.273s	1.21ms	470.8 MiB	2.596 Mbs	1.000
Binary classification	2,400	1,600	0.738s	0.866ms	283.7 MiB	3.139 Mbs	0.997
	240	160	0.075s	1.093ms	353.4 MiB	2.317 Mbs	1.00
Traffic Type	28,543	19,029	7.136s	1.015ms	584.6 MiB	3.728 Mbs	0.997
8-class problem	12,600	8,400	3.31s	1.006ms	305.3 MiB	2.842 Mbs	0.997
	240	160	0.683s	2.397ms	290.6 MiB	2.54 Mbs	0.998
Protocol	17,347	11,565	5.312s	1.138ms	352.4 MiB	3.041 Mbs	0.997
26-class problem	15,600	10,400	3.13s	0.727ms	305.3 MiB	4.231 Mbs	0.998
	1,560	1,040	0.492s	0.901ms	301.8 MiB	3.212 Mbs	0.994

regular expressions added to the sniffer application. We used RExACtor [2] to generate HTTP signatures from the HTTP traffic in our dataset. Both models performed the classification with 100% accuracy on all trials.

System	# Signatures	Training	Testing	Memory
ALPINE/PALM	1,000	1.21s	1.23s	392.4 MiB
	10,000	2.75s	3.56s	419.9 MiB
	100,000	52.75s	4.2s	1047.4 MiB
	1,000,000	423.3s	4.52s	4717.9 MiB
Hyperscan	1,000	4.6s	156ms	70.7 MiB
	10,000	81.8s	1.3s	529.3 MiB
	100,000	39.9m	17.28s	4596.7 MiB
	1,000,000	153m	63.1s	-

TABLE VII
PERFORMANCE RESULTS FOR ALPINE/PALM VERSUS HYPERSCAN

VIII. DISCUSSION

The locality-sensitive hashing approach is an improvement upon the regular expression approach to DPI in several ways. First, the hash generation process reduces the need for expert knowledge about a particular protocol. Previously, engineers would have to employ unique signatures for different message types. It would not be uncommon, for example, to have regular expression signatures for every method for a given text-based protocol like FTP or IMAP. Furthermore, requests and response messages often require unique signatures. Regexes are difficult for humans to read and comprehend. The management of server-client side interactions is also a challenge to know which versions to apply. Instead, hash embeddings using our forward-thinking system create a unique value for each provided input and indexes those values into one bucket. This even more generally captures the broad diversity of possible payload and header contents while providing a new layer of abstraction and simplicity. Flows may be examined bi-directionally, and all message types and methods considered. We are also not limited to traffic flows and protocol types. As demonstrated by the experiments in this paper, the system is capable of distinguishing more heterogenous or abstract class types like traffic type, mechanisms like encryption, and traditionally weak-signatured traffic like RTP. In terms of performance, the LSHForest as implemented in our model further reduces the storage and computational complexity when compared with a regular expression, automata-searching approach.

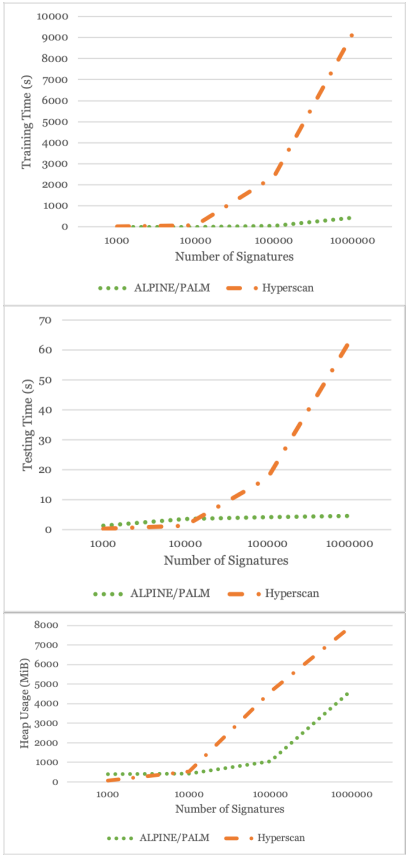

Fig. 4. Time and space complexity results for ALPINE/PALM versus Hyperscan.

R1. The results of our experiments show that for many application layer protocols, locality sensitive hashes may be constructed which are highly accurate for searching when indexed. Our model was 99.6% accurate in its identifications across a wide scale of protocols. In the real world, this sort of expandable model has applications in DPI technology for traffic surveillance, quality of service management and network management, traffic profiling, and content and copyright enforcement. There are also additional filtering applications for cybersecurity and private network management. Detecting traffic at the protocol level remains an important task for network engineers, and our hashing technique improves upon the state of the art in both accuracy and breadth of scope.

R2. Our experiment in classifying packets by traffic type tests both an additional definition of class for network traffic as well as the ability of the model to handle more heterogeneous classes. Even in this problem the hash embedding technique proved highly accurate. This demonstrates the generalizability of the models to real data and new problems.

R3. We bring to the forefront our metric choice of Jaccard similarity because machine learning for network traffic analysis has trended toward more and more technically advanced and often computationally intensive techniques to improve model accuracy or desired metrics. The results of our experiments in this report are evidence that a more complex metric or model is not always the best direction. Even within subfields of machine learning like natural language processing, there is not a silver bullet solution which works best for all kinds of problems. While TF-IDF decently diminishes tokens such as "a" and "the" from the English language, the fact that "HTTP\1.1" appears in nearly every HTTP request in our dataset is actually relevant considering the context. Especially in network traffic analysis where scalability and line-rate capability is highly desirable, simple techniques can still be effective and should not be disregarded.

R4. Multi-embeddings are a proposed solution particularly for more heterogeneous datasets. The idea is that capturing multiple angles or representations of the data and combining those results will yield a more full-scope picture and analysis. Regular expressions, on the other hand, are one-dimensional and incapable of such encoding without becoming increasingly complex. Furthermore, a single bad bit can cause a packet match to fail. Network traffic and packets in the wild are extremely diverse and can be prone to error, and thus are an ideal test set for the multi-embedding hash technique which is both more informative and resilient. The modular design of our system enables using one or more embeddings, which allows for a balance between performance impact and accuracy. Interestingly, though the PALM model was overall poorer-performing than the ALPINE model, votes from the PALM model helped improve the combined result regardless.

R5. One concern for applied machine learning in traffic analysis and cybersecurity in general is the performance impact such techniques have. Thus, we provide transparent numbers for our system performance. All experiments in this trial were run on a 1.6 GHz Dual-Core Intel Core i5 processor with 16GB DDR3 memory. In future work, leveraging an accelerated processor such as an FPGA would greatly enhance performance and throughput due to the repetitive nature of hash computations. In profiling, the most impactful procedure in the model is MinHash, which is likely optimizable through an FPGA [35]. The measure of what is acceptable throughput depends on the network environment; however, in terms of scalability it is valuable that our model does not seem to increase exponentially or vary in performance wildly depending on the data. Rather, it is data independent. Our model also trains quickly, meaning that models could be swapped or trained online in the field if needed. Another potential avenue for future research is batch optimization and parallel processing. It would be beneficial and possible to run some of the training computations in parallel. Furthermore, the distance measures could be taken in parallel for different models on separate threads and then the results aggregated.

R6. As indicated in Table VII, for small signature sets Hyperscan initially outperformed our model; however, when scaled to larger signature sets, our model clearly is much more capable of handling the extra load. The comparative line graphs in Figure 4 illustrate the exponential versus sublinear growth patterns of each measure of performance, clearly demonstrating our systems' scalability over the state of the art. Our model takes just over 7 minutes to construct the LSH Forest with 1 million signatures, while the Hyperscan implementation takes over 2 hours. Even our testing time is much smaller than the Hyperscan test time at scale. For memory usage, our model also performs with reduced space complexity as the scale increases. At 1 million signatures, our model used just over 4GB of heap space, while Hyperscan crashed our setup due to exceeding 8GB at the same scale. If the problem set or data structure size is small, Hyperscan could be an optimal choice; however, our model is more generalizable to larger data sets and adaptable to a variety of problems. It would not be unreasonable to scale to several million signatures in a real surveillance or DPI system which ALPINE and PALM would be better suited for than the current capability of regular expression matching.

IX. Conclusion

We present ALPINE and PALM, techniques for using packet header information and payloads as informative feature sets for locality-sensitive hashes. This approach is a forward-thinking alternative to regular expression payload matching, and may be used for protocol autodetection with highly accurate results. It also requires less computation and storage space than a regular expression/automata scanning approach does. The problem of network traffic classification is incredibly complex, with many levels of granularity and unique features of each protocol and traffic type. Because of this diversity, it is likely that there is no true perfect solution for all classification problems. Instead, the optimal solutions are ones which consider multiple features and are resilient to non-standard implementations. ALPINE and PALM show the potential for locality-sensitive measurements to do just that and identify a large variety of protocols as a fine-tuned system that has the potential to scale in both system complexity and storage capacity for measuring real-world networking applications for the next generation of network technology.

References

[1] D.-Y. Kao, "Using the actionable intelligence approach for the dpi of cybercrime insider investigation," in *2020 22nd International Conference on Advanced Communication Technology (ICACT)*, 2020, pp. 1218–1224.

[2] M. Kapoor, G. Fuchs, and J. Quance, "Rexactor: Automatic regular expression signature generation for stateless packet inspection," in *2021 IEEE 20th International Symposium on Network Computing and Applications (NCA)*, 2021, pp. 1–9.

[3] H. Schulzrinne, S. L. Casner, R. Frederick, and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications," RFC 3550, Jul. 2003. [Online]. Available: https://www.rfc-editor.org/info/rfc3550

[4] J. Hypolite, J. Sonchack, S. Hershkop, N. Dautenhahn, A. DeHon, and J. M. Smith, *DeepMatch: Practical Deep Packet Inspection in the Data Plane Using Network Processors*. New York, NY, USA: Association for Computing Machinery, 2020, p. 336–350. [Online]. Available: https://doi.org/10.1145/3386367.3431290

[5] Google, "Google transparency report," 2022. [Online]. Available: https://transparencyreport.google.com/https/overview?hl=en

[6] IANA, "Assigned protocol numbers," February 2021. [Online]. Available: https://www.iana.org/assignments/protocol-numbers/protocol-numbers.xhtml#protocol-numbers-1

[7] M. Smith and R. Hunt, "Network security using nat and napt," in *Proceedings 10th IEEE International Conference on Networks (ICON 2002). Towards Network Superiority (Cat. No.02EX588)*, 2002.

[8] C. VinothGeorge and V. Ewards, "Efficient regular expression signature generation for network traffic classification," 2013.

[9] Z. Fu, Z. Liu, and J. Li, "Efficient parallelization of regular expression matching for deep inspection," in *2017 26th International Conference on Computer Communication and Networks (ICCCN)*, 2017, pp. 1–9.

[10] F. Yu, Z. Chen, Y. Diao, T. V. Lakshman, and R. H. Katz, "Fast and memory-efficient regular expression matching for deep packet inspection," in *Proceedings of the 2006 ACM/IEEE Symposium on Architecture for Networking and Communications Systems*, ser. ANCS '06. New York, NY, USA: Association for Computing Machinery, 2006, p. 93–102. [Online]. Available: https://doi.org/10.1145/1185347.1185360

[11] X. Wang, Y. Hong, H. Chang, K. Park, G. Langdale, J. Hu, and H. Zhu, "Hyperscan: A fast multi-pattern regex matcher for modern CPUs," in *16th USENIX Symposium on Networked Systems Design and Implementation (NSDI 19)*. Boston, MA: USENIX Association, Feb. 2019, pp. 631–648. [Online]. Available: https://www.usenix.org/conference/nsdi19/presentation/wang-xiang

[12] K.-S. Shim, S.-H. Yoon, S.-K. Lee, and M. Kim, "Sigbox: Automatic signature generation method for fine-grained traffic identification," *J. Inf. Sci. Eng.*, vol. 33, pp. 537–569, 2017.

[13] M. Ye, K. Xu, J. Wu, and H. Po, "Autosig-automatically generating signatures for applications," in *2009 Ninth IEEE International Conference on Computer and Information Technology*, vol. 2, 2009, pp. 104–109.

[14] B.-C. Park, Y. J. Won, M.-S. Kim, and J. W. Hong, "Towards automated application signature generation for traffic identification," in *NOMS 2008 - 2008 IEEE Network Operations and Management Symposium*, 2008, pp. 160–167.

[15] O. Salman, I. H. Eljhajj, A. Kayssi, and A. Chehab, "A review on machine learning-based approaches for internet traffic classification," *Ann. Telecommun.*, vol. 75, no. 11-12, pp. 673–710, December 2020.

[16] J. Cao, Z. Fang, G. Qu, H. Sun, and D. Zhang, "An accurate traffic classification model based on support vector machines," *Netw.*, vol. 27, no. 1, p. n/a, January 2017. [Online]. Available: https://doi.org/10.1002/nem.1962

[17] H.-K. Lim, J.-B. Kim, J.-S. Heo, K. Kim, Y.-G. Hong, and Y.-H. Han, "Packet-based network traffic classification using deep learning," in *2019 International Conference on Artificial Intelligence in Information and Communication (ICAIIC)*, 2019, pp. 046–051.

[18] Z. Li, R. Yuan, and X. Guan, "Accurate classification of the internet traffic based on the svm method," in *2007 IEEE International Conference on Communications*, 2007, pp. 1373–1378.

[19] M. Song, J. Ran, and S. Li, "Encrypted traffic classification based on text convolution neural networks," in *2019 IEEE 7th International Conference on Computer Science and Network Technology (ICCSNT)*, 2019, pp. 432–436.

[20] G. Draper-Gil., A. H. Lashkari., M. S. I. Mamun., and A. A. Ghorbani., "Characterization of encrypted and vpn traffic using time-related features," in *Proceedings of the 2nd International Conference on Information Systems Security and Privacy - ICISSP,*, INSTICC. SciTePress, 2016, pp. 407–414.

[21] A. H. Lashkari, G. Draper-Gil, M. S. I. Mamun, and A. A. Ghorbani, "Characterization of tor traffic using time based features," in *ICISSP*, 2017.

[22] O. Jafari, P. Maurya, P. Nagarkar, K. M. Islam, and C. Crushev, "A survey on locality sensitive hashing algorithms and their applications," 2021.

[23] P. Indyk and R. Motwani, "Approximate nearest neighbors: Towards removing the curse of dimensionality," in *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing*, ser. STOC '98. New York, NY, USA: Association for Computing Machinery, 1998, p. 604–613. [Online]. Available: https://doi.org/10.1145/276698.276876

[24] A. Z. Broder, "Identifying and filtering near-duplicate documents," in *Proceedings of the 11th Annual Symposium on Combinatorial Pattern Matching*, ser. COM '00. Berlin, Heidelberg: Springer-Verlag, 2000, p. 1–10.

[25] M. Gabryel, K. Grzanek, and Y. Hayashi, "Browser fingerprint coding methods increasing the effectiveness of user identification in the web traffic," *Journal of Artificial Intelligence and Soft Computing Research*, vol. 10, no. 4, pp. 243–253, 2020.

[26] U. Bayer, P. M. Comparetti, C. Hlauschek, C. Kruegel, and E. Kirda, "Scalable, behavior-based malware clustering." in *NDSS*, vol. 9. Citeseer, 2009, pp. 8–11.

[27] M. Bawa, T. Condie, and P. Ganesan, "Lsh forest: self-tuning indexes for similarity search," in *In WWW*, 2005.

[28] P. Lee, L. V. Lakshmanan, and J. X. Yu, "On top-k structural similarity search," in *2012 IEEE 28th International Conference on Data Engineering*, 2012, pp. 774–785.

[29] H. Yang, Q. He, Z. Liu, and Q. Zhang, "On top-k structural similarity search," *Security and Communication Networks*, vol. 2021, 2021.

[30] B. Subba and P. Gupta, "A tfidfvectorizer and singular value decomposition based host intrusion detection system framework for detecting anomalous system processes," *Computers & Security*, vol. 100, p. 102084, 2021.

[31] T. Blue and H. Shahriar, "Distributed denial of service attack detection," in *National Cyber Summit*. Springer, 2020, pp. 240–241.

[32] R. Rawat, V. Mahor, S. Chirgaiya, R. N. Shaw, and A. Ghosh, "Analysis of darknet traffic for criminal activities detection using tf-idf and light gradient boosted machine learning algorithm," in *Innovations in Electrical and Electronic Engineering*, S. Mekhilef, M. Favorskaya, R. K. Pandey, and R. N. Shaw, Eds. Singapore: Springer Singapore, 2021, pp. 671–681.

[33] A. Dalvi, I. Siddavatam, A. Jain, S. Moradiya, F. Kazi, and S. G. Bhirud, "Element: Text extraction for the dark web," in *Advanced Computing and Intelligent Technologies*, M. Bianchini, V. Piuri, S. Das, and R. N. Shaw, Eds. Singapore: Springer Singapore, 2022, pp. 537–551.

[34] Z. Tang, Q. Wang, W. Li, H. Bao, F. Liu, and W. Wang, ""hslf: Http header sequence based lsh fingerprints for application traffic classification"," in *Paszynski M., Kranzlmüller D., Krzhizhanovskaya V.V., Dongarra J.J., Sloot P.M.A. (eds) Computational Science – ICCS 2021*, 2021.

[35] W. Jiang and M. Gokhale, "Real-time classification of multimedia traffic using fpga," in *2010 International Conference on Field Programmable Logic and Applications*, 2010, pp. 56–63.

[36] Y. Leskovec, A. Rajaraman, and J. Ullman, *Mining of Massive Datasets*. Standford University, 2022. [Online]. Available: http://www.mmds.org/

[37] B. Sangster, T. J. O'Connor, T. Cook, R. Fanelli, E. Dean, W. J. Adams, C. Morrell, and G. Conti, "Toward instrumenting network warfare competitions to generate labeled datasets," in *Proceedings of the 2nd Conference on Cyber Security Experimentation and Test*, ser. CSET'09. USA: USENIX Association, 2009, p. 9.

[38] C. Guarnieri, "Skynet: a tor-powered botnet straight from reddit," 2012. [Online]. Available: https://contagiodump.blogspot.com/2012/12/dec-2012-skynet-tor-botnet-trojantbot.html

[39] G. D. Gil, A. H. Lashkari, M. Mamun, and A. A. Ghorbani, "VPN/non-VPN Dataset (ISCXVPN2016)," 2016. [Online]. Available: https://www.unb.ca/cic/datasets/vpn.html

[40] ——, "Tor/non-Tor Dataset (ISCXTor2016)," 2016. [Online]. Available: https://www.unb.ca/cic/datasets/tor.html

[41] J. V. V. Silva, N. R. Oliveira, D. S. V. Medeiros, M. A. Lopez, and D. M. F. Mattos, "A statistical analysis of instrinsic bias of network security datasets for training machine learning mechanisms," February 2022. [Online]. Available: https://doi.org/10.1007/s12243-021-00904-5

[42] K. Tumer and J. Ghosh, "Error correlation and error reduction in ensemble classifiers," *Connection science*, vol. 8, no. 3-4, pp. 385–404, 1996.