Extensible Machine Learning for Encrypted Network Traffic Application Labeling via Uncertainty Quantification

Steven Jorgensen, John Holodnak, Jensen Dempsey, Karla de Souza, Ananditha Raghunath, Vernon Rivet, Noah DeMoes, Andrés Alejos, and Allan Wollaber

***Abstract*—With the increasing prevalence of encrypted network traffic, cybersecurity analysts have been turning to machine learning (ML) techniques to elucidate the traffic on their networks. However, ML models can become stale as new traffic emerges that is outside of the distribution of the training set. In order to reliably adapt in this dynamic environment, ML models must additionally provide contextualized uncertainty quantification to their predictions, which has received little attention in the cybersecurity domain. Uncertainty quantification is necessary both to signal when the model is uncertain about which class to choose in its label assignment and when the traffic is not likely to belong to any pretrained classes. We present a new public dataset of network traffic that includes labeled virtual-private-network-encrypted network traffic generated by ten applications and corresponding to five application categories. We also present an ML framework that is designed to rapidly train with modest data requirements and provide both calibrated predictive probabilities and an interpretable "out-of-distribution" (OOD) score to flag novel traffic samples. We describe calibrating OOD scores using p-values of the relative Mahalanobis distance. We demonstrate that our framework achieves an F1-score of 0.98 on our dataset and that it can extend to an enterprise network by testing the model: 1) on data from similar applications; 2) on dissimilar application traffic from an existing category; and 3) on application traffic from a new category. The model correctly flags uncertain traffic and, upon retraining, accurately incorporates the new data.**

***Impact Statement*—Labeling application-generated virtual private network (VPN)-encrypted network traffic is challenging given limited public datasets, the difficulty of signal extraction using only packet size and timing information, and the need to account for uncertainty in the machine learning (ML) pipeline to screen out predictions for novel applications or low-confidence predictions. We provide a new public labeled packet capture dataset with VPN-encrypted network traffic for researchers to train their own models, filling a needed gap for this problem domain. We propose data features and a neural network architecture that accurately trains with modest data requirements and provides predictions alongside confidence scores that allow analysts to know when a model is unsure about a given prediction. By integrating data, feature generation, and confidence-aware ML, we have laid a foundation that will allow future researchers to create robust analytics for this challenging domain at the intersection of cyber security and ML.**

***Index Terms*—Cybersecurity, discrete wavelet transform, encrypted traffic, machine learning (ML), network traffic classification, uncertainty quantification, virtual private networks (VPNs).**

Manuscript received 14 June 2022; revised 15 September 2022 and 24 October 2022; accepted 15 January 2023. Date of publication 14 February 2023; date of current version 8 January 2024. This work was supported by U.S. Cyber Command under Air Force Contract FA8702-15-D-0001. This paper was recommended for publication by Associate Editor Supratik Mukhopadhyay upon evaluation of the reviewers' comments. *(Corresponding author: Steven Jorgensen.)*

The authors are with the MIT Lincoln Laboratory, Lexington, MA 02421 USA (e-mail: steven.jorgensen@ll.mit.edu; john.holodnak@ll.mit.edu; jensen.dempsey@ll.mit.edu; karla.deSouza@ll.mit.edu; ananditha.raghunath@ll.mit.edu; vernon.rivet@ll.mit.edu; noah.demoes@ll.mit.edu; andres.alejos@ll.mit.edu; allan.wollaber@ll.mit.edu).

Digital Object Identifier 10.1109/TAI.2023.3244168

I. Introduction

THE proportion and popularity of encrypted network traffic has quickly risen over the past several years, with over 95% of traffic across Google and 97 of the top 100 websites defaulting to encryption [1]. This is a boon for online privacy; however, there is a corresponding increase in encryption for malware delivery [2], [3]. Beyond that, MITRE ATT&CK enumerates several techniques in which encryption has been used to obfuscate command and control or to masquerade one application as another. Although signature-based techniques, such as website fingerprinting [4], [5] and Palo Alto's App-ID, can flag particular packet sequences that indicate a known website request or protocol handshake, the general problem of inferring an application when signatures and heuristics fail remains a challenging problem. In particular, virtual private network (VPN) providers have also lowered the barriers for users not only to encrypt the ports, protocols, and IP addresses of their connections, but also to obfuscate the IP address of the VPN server itself [6] and/or pad all packets to be the same size before encryption [7], which can defeat signature detection capabilities. In light of these difficulties, network security analysts have begun turning to machine learning (ML) approaches that leverage latent signals in the packet timings, sizes, and their encrypted payloads in order to predict the applications that generated the encrypted traffic.

Until recently, the existing work in ML for encrypted traffic classification has focused primarily on feature construction and model development to optimize raw performance metrics such as overall accuracy. For example, traditional ML models (like naïve Bayes and decision trees) have been applied to simple statistical features [8], [9], and neural networks based on one- or two-dimensional convolutions have been applied to sequences of raw byte values, interarrival times, and wavelet coefficients [10], [11], [12], [13], [14], [15], [16]. Many of these techniques yield very good performance on publicly available data, the most popular being the ISCXVPN2016 dataset for studying VPN encryption [9]. Much more recently, end-to-end and transformer-based architectures have begun showing impressive performance [17], [18], [19], [20] on a variety of Internet of Things (IoT) and mobile traffic datasets outside of a VPN setting [21], [22], [23].

In this article, we build upon some previously developed features (interarrival statistics and wavelet signal processing) with an eye toward enabling rapid learning with limited data. In addition, we focus on developing an ML model with accurate uncertainty quantification that is able to 1) produce accurate probabilistic predictions for the classes on which the model is trained and 2) detect examples dissimilar to application categories that the model was trained to predict. While there has been some limited mention of uncertainty quantification in cybersecurity applications of ML in general [24], [25], [26], [27], [28], the problem has not received the same attention in cybersecurity as it has in other domains, such as computer vision and health care. We consider this unfortunate and believe that in a dynamic environment such as a computer network, it is critically important to quantify model uncertainty. A recent review highlighted the need to reduce errors (false positives) when moving from a closed training set to real-world data for encrypted network traffic analysis [29], and Nascita et al.'s [28] emphasis on trustworthiness and explainability in deep learning in traffic classification is a step in the right direction.

In our work, we develop a classifier for encrypted network traffic based on prototypical networks, which leverage distances to class prototypes (means) in a learned embedding space to compute class probabilities and, thus, predictions [30]. We use the recently developed relative Mahalanobis distance between test examples and class prototypes to detect out-of-distribution (OOD) examples [31]. Our model uses as input a mixture of simple features derived from flow statistics as well as wavelet transform coefficients to produce class predictions for segments of bidirectional network flows (although more advanced transformer architectures such as in [19] should also be compatible with a prototypical network layer). To enable regular predictions in VPN settings in which 1) the embedded five-tuples for connections are encrypted and 2) the beginnings and endings of connections are obfuscated, we split all network flows into discretely sampled time segments and make one prediction for each time segment in each flow.

We apply our model to a new dataset (which we have made publicly available) of encrypted network traffic collected on a testbed from ten applications that cover five broad, but not comprehensive categories of traffic. This dataset supplements those already available in the literature, which either cover only very specific types of network traffic such as the UC Davis dataset of QUIC traffic [14] or the Orange dataset of HTTPS traffic [16] or contain artifacts such as unencrypted packets that could affect the validity of models trained on them.[1] When trained with an 80/20 train/test split, our model has a micro F1-score of 0.98. Furthermore, we investigate our model's performance when training with only small fractions of the dataset, achieving (for instance) 95% accuracy on Voice over Internet Protocol (VoIP) traffic in a five-class problem using just under 14 min of data capture. On the UTMobileNetTraffic2021 dataset, our model has 80% accuracy, comparable to Heng et al.'s [22] best results.

We stress-test the model by attempting to transfer its learning from the testbed to an enterprise network over several categories. We found that in some instances (a streaming and VoIP application), the model cleanly transfers onto another network with high accuracy and low uncertainty, and in others (two file transfer applications), the model successfully reports low in-distribution confidence and can be quickly retrained to incorporate the new applications for identification in the known category.

To test our model's ability to identify OOD examples, we perform inference on PCAP containing an unseen traffic category (Zoom) from an enterprise network and demonstrate that the model reliably assigns the new category a high OOD score, with over 77% predicted as significantly OOD. Upon retraining with about 2 h of labeled examples of Zoom, which takes under 5 min, the fraction of significantly OOD test examples drops to 26%.

Finally, we test our model's robustness by investigating the case where all packets are padded to have identical sizes, as occurs in the Noise Protocol or in certain implementations of traffic flow confidentiality [7], [32]. We see that the model (which utilizes both packet timing and size information) is still able to obtain very good performance (micro F1-score of 0.97) indicating that packet size information is not necessary for accurate classification. In addition, this observation calls into question the degree of privacy offered by such obfuscation strategies.

To summarize, the primary contributions of this article are as follows:

1) a new public dataset of application-labeled, VPN-encrypted, and non-VPN-encrypted network traffic;
2) an ML architecture for encrypted traffic designed to quickly train with limited labeled data and provide calibrated uncertainty-contextualized predictions for two tasks: a) predictive uncertainty, when the model is indecisive about which *known* label a sample belongs to; and b) model uncertainty, when the model encounters OOD data potentially from an *unknown* label, enabling the model to be systematically extended to new data;
3) a demonstration of sufficiently high accuracy and F1-score for application data that are grouped into discrete time windows, instead of by collecting statistics or data for an entire connection or relying upon fingerprints or signatures, enabling sequential application predictions under a VPN.

The rest of this article is organized as follows. Section II presents related work. Section III discusses our PCAP dataset. Section IV presents data and learning techniques. Section V presents computational experiments. Section VI gives discussion. Finally, Section VII concludes this article.

[1] See Section II-E for specific examples of artifacts we identified in the often-used ISCXVPN2016 dataset, introduced in [9], which we hope to supplement.

II. Related Work

In this section, we describe the existing work in the literature related to traffic classification in general, encrypted traffic classification, and website fingerprinting. Owing to our focus on uncertainty in encrypted traffic classification, we briefly review relevant work on uncertainty in deep learning from the ML community. Finally, we discuss existing encrypted traffic datasets and our motivation for collecting our own.

A. Early Traffic Classification

In the early days of the Internet, censors used port-based traffic classification to restrict Internet content [33]. Port-based classification models could classify network traffic using the port numbers encoded in the packet headers because the Internet Assigned Numbers Authority maps services to unique port numbers [34]. In response, clever Internet users and application developers adapted to port-based censorship by dynamically changing port numbers, hiding behind port numbers already assigned to well-known, trusted applications, or using unregistered port numbers [35], [36], [37]. Deep packet inspection (DPI) then emerged as a technique that inspects the payloads of packets and classifies them accordingly [33], [37], [38], [39], [40]. DPI is often used to check for malicious code, obtain situational awareness on a network, and eavesdrop on connections. For example, the operators of the Great Firewall of China have used DPI to inform active probing to identify and censor Tor [33]. However, with the rise of easy-to-use encryption schemes, such as HTTPS and VPNs, DPI has lost its effectiveness. Although possible, traffic decryption is not a viable countermeasure, because decryption not only compromises user privacy but is also computationally expensive and difficult to implement [41]. VPNs in particular can effectively obscure payloads, ports, and IP addresses, through encryption, leaving only packet timing and size information as potential signal sources.

B. Encrypted Traffic Classification

To overcome the limitations of port- and payload-based classification models, researchers have focused on traffic classification models that use features from observable encrypted traffic metadata [29], [41].

Some examples of features constructed from observable metadata include features based on simple statistics derived from flows [9], [36], [37], [41], [42], [43], [44], [45], [46], [47], [48], [49], [50] and wavelet-based features [10], [51]. While traffic classification methods that use flow-statistic-based features yield respectable results for isolated application classification in contained environments, it is uncertain whether they are robust enough to be utilized in real-world settings. Flow-statistic-based features can vary widely between network topologies and are not able to characterize the highly variable and bursty nature of Internet traffic [52], [53], [54], [55]. On the other hand, wavelet-based features can capture the inherent nonlinearities of Internet traffic, such as jumps and edges, by providing representations of the observable metadata from the bidirectional connection that are localized in both time and frequency [56]. Shi et al. [51] are the first to use wavelet-based features for traffic classification and show that support vector machines trained with wavelet features outperform those trained with flow-statistic-based features. Later, Liang et al. [10] used convolutional neural networks trained on wavelet-based features for traffic classification. We build on this approach by dividing the connections into chunks of time rather than classifying the aggregate connection all at once and combining flow-statistic-based features and wavelet features in each time window. Roy et al. [57] also use a time-based approach via a neural ordinary differential equation on packet interarrival times, and they also avoid payload inspection, which is similar in spirit to our approach. Shapira and Shavitt [58] use a clever time/size technique to generate images ("FlowPic") from application traffic, which then enables standard image classification techniques. Recently, end-to-end and transformer-based architectures have bypassed the need for feature generation, shifting the burden into the neural network, which typically leverages on the order of a million parameters and achieve state-of-the-art accuracies [17], [18], [19], [20]. In our work, we achieve high accuracy using fewer than 30 000 parameters, which enables rapid training (minutes).

C. Website Fingerprinting

Website fingerprinting is a process of recognizing specific web traffic (e.g., to particular websites) based on unique patterns in the traffic [59]. This approach leverages the fact that websites tend to have unique packet sequence patterns (handshakes) that allow for identification. Traffic features can include information such as unique packet lengths, sequence lengths, packet ordering, and packet interarrival timings [60]. These features are then fed to ML algorithms that classify traffic based on the observed patterns. However, the identifying handshake must be learned beforehand and observed in a traffic sample to enable detection. Website fingerprinting techniques also tend to be sensitive to changes in network patterns such as packet padding and fluctuations in packet round trip time, and it is challenging to know when the models must be updated [61].

We envision our approach would work in tandem with website fingerprinting and signature-based approaches, allowing for contextual uncertainty-quantified predictions to be made before or after a handshake has been identified. An example of this is the case where, within some packet capture, website fingerprinting identifies a handshake to a malicious website. Our approach could then also be applied to the connection to determine if any file transfer or new command and control channels occurred after the signature was detected.

D. Uncertainty in Deep Learning

Quantifying the uncertainty of model predictions is an active area of research in the ML community. Considerable research has been focused on improving the ability of classification models to output calibrated probabilistic predictions and to detect OOD examples at test time.

To be clear, a model's predictive probabilities are calibrated if their confidence (highest predicted probability) matches their accuracy. In other words, a model should correctly label 90%

TABLE I
DISTANCE-BASED APPROACHES TO OOD DETECTION

Approach	Comments
Mahalanobis distance to class prototypes [31], [62], [63], [64], [65]	Ming et al. [65] modify the loss to separate class prototypes using angular distance [31] normalizes the distance to the class prototypes using the distance to the overall data distribution
Euclidean distance to fixed point in feature space [66]	Fixed point is the embedding of random noise
Euclidean distance to nearest neighbors [67]	Avoids making parametric assumptions (multivariate normality) on the embeddings
Cosine similarity to weight vectors in last layer [68]	Modifies loss to use softmax of similarities

of the examples that it categorizes with 90% confidence. Guo et al. [69] showed that while shallow neural networks tend to be reasonably well calibrated, deep neural networks are not. Various procedures for improving calibration have been proposed, including temperature calibration [69], deep ensembles [70], and Bayesian model averaging [71]. In the context of traffic classification, Nascita et al. [28] evaluate the calibration of several classifiers in terms of expected calibration error (ECE) (binwise average difference between confidence and accuracy) and classwise ECE, and their emphasis on trustworthiness and explainability makes their article nearest in spirit to this work, although it does address OOD detection.

We now briefly review some of the major approaches to OOD detection in the ML literature. The degree to which examples are OOD is usually quantified using either the distribution of predicted probabilities for examples or some notion of distance to the distribution of training data. A detailed review of OOD detection approaches is given in [72].

For example, the magnitude of the largest predicted probability was used in [73] as a score, encoding the assumption that if a data example is OOD, no class will be assigned a high probability. Other work [74] improved this approach by applying input preprocessing and softmax temperature scaling to improve separation between the largest predicted probabilities of in-distribution and OOD examples. In another approach, Malinin and Gales [75] model the output probabilities as a distribution and place a Dirichlet prior over them. During training, they attempt to enforce the assumption that uncertain examples produce a flat distribution of predictive probabilities by training on OOD examples. This idea is improved in [76], which removes the necessity for using OOD examples in training.

On the other hand, Lee et al. [62] fit class-conditional Gaussian distributions in the embedding space of a model with softmax output probabilities and compute Mahalanobis distances to the Gaussian distributions at test time. A very similar approach is taken in [63], but in the context of meta-learning and prototypical networks. In this case, the authors use the distance to the closest class prototype as an OOD score. Maciejewski et al. [64] examine the effect of covariance types and the dimension of the embedding space on OOD detection performance. To encourage separation between classes, Ming et al. [65] attempt to minimize the angular distance between a class prototype and examples of that class and maximize the angular distance different class prototypes. They then use the Mahalanobis distance to the closest class to uncover OOD examples. Other interesting approaches include basing OOD detection on the distance to a fixed point in feature space, representing the embedding of random noise [66] and computing the cosine similarity between the embedding and the last layer weight vectors [68]. Finally, to avoid making distributional assumptions, Sun et al. [67] compute the distance to the kth nearest neighbor.

Our approach is also based on the distance of test examples to the training data. It is most similar to that in [63], but instead makes use of the relative Mahalanobis distance from [31]. We summarize the distance-based approaches, as they are most similar to ours, in Table I.

E. Publicly Available VPN-Encrypted Traffic Datasets

The development of reproducible research in VPN-encrypted traffic classification is hindered by the shortage of publicly available VPN-encrypted traffic data. In 2016, the Canadian Institute for Cybersecurity published the ISCXVPN2016 dataset of VPN and non-VPN traffic from 20 applications and seven traffic categories [9]. As the first publicly available dataset of VPN traffic, the ISCXVPN2016 dataset has been used by many researchers to train and test ML models. Other datasets with encrypted traffic have since been released as well, such as the UC Davis dataset of QUIC traffic [14], the Orange dataset of HTTPS traffic [16], as well as multiple IoT and mobile datasets [21], [22], [23], although none of these distinguishes between VPN and non-VPN encrypted data.

Despite its widespread use, we found that the ISCXVPN2016 dataset contains discrepancies in the VPN captures that compromise the integrity of encrypted network traffic classification methods evaluated on it (these are in addition to Aceto et al.'s [20] observation that 65% of the biflows are due to BlueStacks and should be filtered out). Upon closer inspection of the PCAP files from the VPN captures, we find packets with unencrypted payloads. For example, the payload of the 17th packet in the PCAP file for the ICQ Chat VPN capture in `vpn_icq_chart1b.pcap` contains body HTML text saying "how are you today," which can be easily viewed in WireShark. We also find PCAP files for VPN-labeled captures to contain multiple connections, although we expect the PCAP files for a VPN session to contain a single connection between the VPN client and the VPN server. As an example, the vast majority of packets in the `vpn_netflix_A.pcap` file are over port 80 to a Netflix IP address, the majority of packets in `vpn_hangouts_audio1.pcap` resolve

TABLE II
APPLICATIONS AND TRAFFIC CATEGORIES USED TO GENERATE THE DATASET

Traffic Category	**Applications**
Streaming	Vimeo, Netflix, YouTube
VoIP	Zoiper
Chat	Skype
Command & Control	SSH, RDP
File Transfer	SFTP, RSYNC, SCP

to a Google IP address, and the majority of packets in `vpn_facebook_audio2.pcap` resolve to a Facebook IP address (with a minority fraction also going to Google). These findings imply that either the network was tapped before the packets went through the VPN client or the packets were not encrypted.

Unencrypted packet payloads will unfairly strengthen traffic classification methods that leverage packet payloads, such as DPI or the deep packet approach in [15]. In addition, the presence of multiple connections per VPN capture will unfairly strengthen methods that leverage connection information aggregated at the flow level.

In light of these concerns, we recommend that researchers seeking to utilize the ISCXVPN2016 dataset ensure that their traffic classification methods do not presume that VPN-labeled packet payloads are VPN-encrypted. In addition, if their methods for analyzing VPN-encrypted traffic rely on connection information aggregated at the flow level, we recommend that they postprocess the packet headers to ensure that all packets appear under the same five-tuples, as they would inside a VPN connection. Alternatively, one could also replay the VPN-labeled files through a VPN.

Finally, there is also a challenge with respect to consistent assignment of the traffic categories to the collected PCAP files. As an example, the file `skype_video1a.pcap` could reasonably be categorized as VoIP or Web Browsing, but probably would have fit better with other video-teleconferencing (VTC) PCAPs in its own VTC category.

In response to the limitations of the ISCXVPN2016 dataset, we introduce a supplemental labeled dataset of VPN and non-VPN network traffic for public use and to test our extensible ML framework.

III. Encrypted Traffic Dataset

In this section, we describe the characteristics of our dataset[2] and outline the process used to produce the dataset. Our dataset is a collection of 36.1 GB of labeled network traffic that contains both VPN-encrypted and unencrypted flows from the five traffic categories shown in Table II, which is inspired by the categories in [9] with a few omissions (e.g., P2P and Web Browsing) and a new category (Command & Control). See Table VI in the Appendix for the file mappings used to assign class labels.

[2]The dataset is available for download at: https://www.ll.mit.edu/r-d/datasets/vpnnonvpn-network-application-traffic-dataset-vnat

TABLE III
TOTAL PACKET SIZES, CONNECTIONS, AND TIME PER TRAFFIC CATEGORY IN THE DATASET, AS WELL AS THE PERCENTAGE OF EACH CATEGORY THAT IS VPN ENCRYPTED

Traffic Category	**Size (MB)**	**Connections**	**Time (h)**	**VPN %**
File Transfer	32,601	16,430	14.8	43
Streaming	2626	1,764	4.3	41
VoIP	103	617	2.7	65
Command & Control	622	13,599	127.4	49
Chat	185	1,301	123.0	63
Total	36,137	33,711	272.3	43

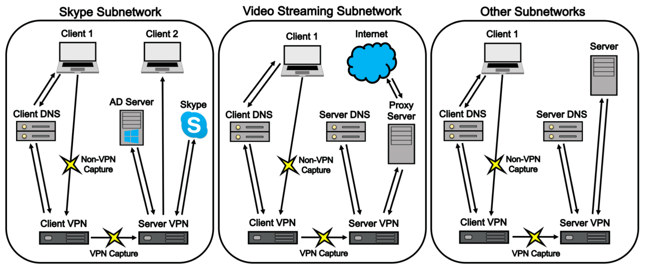

Fig. 1. Diagram of the virtual subnetworks used to capture network traffic. The yellow stars denote where the data captures occur. The non-VPN traffic is captured between the client and the client VPN, while the VPN traffic is captured between the client VPN and the VPN server.

A. Dataset Characteristics

Our dataset consists of 36.1 GB, 33 711 connections, and approximately 272 h of packet capture from five traffic categories, as listed in Table III. Although the File Transfer category constitutes 90% of the total size of the dataset, it only contributes to roughly 40% and 5% of the total connections and time, respectively. In contrast, the Command & Control category only makes up about 2% of the dataset in size while comprising 40% of the connections and almost 47% of the total time. The Streaming category contributes approximately 7%, 5%, and 2% of the total size, connections, and time, respectively. The Chat category contributes less than 1% of the total size, 4% of the total connections, and 45% of the total time. The smallest traffic category, VoIP, makes up around 1% of the total size and time and less than 2% of the total connections. Although each protocol is encrypted by default, we also report the percentage of each category that is VPN-encrypted, which varies from 41% to 65% by size. In addition to the complete dataset in PCAP form, researchers can also download a pandas dataframe in HDF5 format that contains the connection (biflow) five tuples, packet time stamps, directions, and corresponding label (filename containing application and VPN status).

B. Network Setup

To produce the dataset, we begin by creating virtual subnetworks for each traffic category, as shown in Fig. 1. Each subnetwork contains a client, a client DNS server, a VPN client, and a VPN server. The Skype subnetwork contains an additional client to allow for bidirectional chat. The video streaming and web browsing subnetworks are connected to the Internet to enable access to Firefox, Chrome, YouTube, Netflix, and Vimeo. As shown in Fig. 1, VPN traffic is captured between the VPN client and the VPN server. Separately, non-VPN traffic is captured between the VPN client and the application layer.

C. Data Generation

Once the network setup is complete, we generate the dataset for each traffic category individually to ensure data purity. All traffic is captured using tcpdump and stored in the PCAP format. Filename-to-category mappings are provided in Table VI in the Appendix.

The Streaming category includes Vimeo, Netflix, and YouTube traffic generated by team members watching videos on each platform. The VoIP category is composed of VoIP audio generated by team members having conversations with one another using Zoiper. The Chat category consists of Skype Chat traffic generated with bots that replay actual chats from publicly available logs from Gitter.im chat rooms [77].

The Command & Control category includes SSH and RDP traffic. RDP traffic is generated by team members actively performing tasks, such as editing Word documents, over an RDP connection. To emulate SSH traffic, we created a script to randomly sample 2–20 commands with replacement from a predetermined list of 17 commands. For each command, it samples a random wait time between 1 and 60 seconds. Once it is finished executing each command and wait time, it waits 60 seconds before repeating the entire process from the beginning.

The File Transfer category contains SFTP, RSYNC, and SCP traffic. To emulate users performing file transfers, we created scripts that randomly choose a file out of a list of 15 files of sizes ranging from 1 kB to 1 GB. Then, they randomly decide whether to send the chosen file to or from a remote server. After executing the file transfer, the scripts wait 60 seconds before selecting a new file.

IV. Methods

In this section, we discuss the features that we used as model input, the specifics of our model architecture, and how we define OOD scores.

A. Data Preprocessing

We group packets into connections based on the five tuples consisting of the source IP address, destination IP address, source port, destination port, and protocol using the dpkt[3] Python module. We split the connections into 40.96-second intervals[4] and discard any intervals containing fewer than 20 packets. For each 40.96-second interval, we obtain the packet time stamps, payload sizes, and directions discretized into 0.01-second bins. After this preprocessing, the data collection contains 1675 Command & Control examples, 10 498 Chat, 851 File Transfer, 1827 Streaming, and 243 VoIP examples.

[3]https://github.com/kbandla/dpkt

[4]A discussion of our selection of 40.96-second intervals and 0.01-second bins is provided in the Appendix.

TABLE IV
Flow Statistic Features Calculated Over Each Time Window

Feature	Description
total_forward	Number of forward packets
total_backward	Number of backward packets
bytes_per_sec	Total bytes per second
total_bytes_forward	Total forward bytes
total_bytes_backward	Total backward bytes
FIAT (mean, min, max, std)	Forward inter-arrival time
BIAT (mean, min, max, std)	Backward inter-arrival time
FlowIAT (mean, min, max, std)	Flow inter-arrival time
Active (mean, min, max, std)	Total seconds flow is active
Idle (mean, min, max, std)	Total seconds flow is idle

TABLE V
Wavelet-Based Features Calculated Over Each Time Window

Feature	Description
RelEng_Forward	Forward relative wavelet energy
RelEng_Backward	Backward relative wavelet energy
Entropy_Forward	Forward Shannon entropy
Entropy_Backward	Backward Shannon entropy
MeanDetail_Forward	Absolute mean of forward detail coefficients
MeanDetail_Backward	Absolute mean of backward detail coefficients
StdDetail_Forward	Standard deviation of forward detail coefficients
StdDetail_Backward	Standard deviation of backward detail coefficients

B. Feature Construction

For each 40.96-second interval, we compute aggregate statistics (such as total, average, maximum, etc.) for flow features such as interarrival times and time spent as active or idle. We compute statistics both for the flow's forward and backward direction. We note here that in our current implementation, "forward" and "backward" are arbitrary; the first observed packet in the connection is designated as "forward" for the remainder of the connection. The full set of aggregate statistics features are listed in Table IV and are very similar to those in [9].

We also generate wavelet-based features from the 40.96-second intervals using the discrete wavelet transform, as done by Shi et al. [51]. We select the Haar basis as the mother wavelet because it produces the maximum-energy-to-Shannon-entropy ratio. Before calculating the wavelet-based features, we extract the detail coefficients from frequency bands 0–12 of the wavelet-transformed connection sizes in the forward and backward directions using the PyWavelets Python package [78]. Then, we use the detail coefficients of each band to calculate the wavelet-based features from the connection sizes in the forward and backward directions. The wavelet-based features include the relative wavelet energy, Shannon entropy, and absolute means and standard deviations of the detail coefficients, as shown in Table V. The absolute means and standard deviations of the detail coefficients are log-transformed to induce symmetry because the original distributions of these features are heavily right-skewed. See the Appendix for details on how to calculate the wavelet-based features.

In total, this data pipeline collapses the (up to) gigabytes of network traffic that could flow across 40.96 seconds into two vectors of length 4096 that are then reduced to 129 features per time window. Some of these features are also correlated and could be winnowed (data not shown) to further sparsen the signal.

C. Classification Model

Prototypical networks are neural networks that compute class probabilities using the distances to a set of the so-called prototypical examples [30], rather than by applying a softmax activation function to the output layer. The neural network that forms the base of the prototypical network is usually referred to as an embedding function. We denote the embedding function as $f_\theta : \mathbb{R}^D \to \mathbb{R}^E$, where D is the dimension of the feature space, E is the dimension of the embedding space, and θ are weights to be learned. We assume access to a set $X = \{x_i\}$ of examples with corresponding labels $y_i \in \{1, 2, \ldots, K\}$. We will use the notation X_{train}, X_{cal}, and X_{test} to refer to partitions of the dataset. The prototype for each class k, $1 \le k \le K$, which we denote as c_k, is simply the mean of the embedding of a few (S) examples of the class (known as the support examples), that is, $c_k = \frac{1}{S}\sum_{s=1}^{S} f_\theta(x_s^k)$. Here, the superscript k indicates the examples come from class k.

For a given observation, the class probabilities are calculated by applying the softmax activation function to the distances between the class prototypes and the embedding of the observation. Specifically, the probability of class k for input example x_i is

$$P(y_i = k|x_i) = \frac{\exp(-d_k(f_\theta(x_i), c_k))}{\sum_{k'} \exp(-d_{k'}(f_\theta(x_i), c_{k'}))} \tag{1}$$

where each d_k is a distance function on the embedding space.

In our application, we use an embedding model with four fully connected hidden layers each containing 64 neurons with rectified linear unit activations. We perform 25% dropout on the weights connecting the second and third and the third and fourth layers.

We learn our model's parameters using the episodic training paradigm [30]. In episodic training, for each of N_{episode} episodes, we sample N_{query} query examples and S_{train} support examples. The prototypes are computed from the support examples, and we compute the cross-entropy loss over the query examples. We summarize our training procedure in Algorithm 1.

Throughout this article, we set N_{episode} to 20 000, N_{query} to 512, and S_{train} to 5. As in [30], the choice of squared Euclidean distance during training roughly corresponds to spherical Gaussian densities in $\mathbb{R}^E$; this motivates our use of the Mahalanobis distance for OOD scoring.

D. OOD Scores

We now describe how we obtain OOD uncertainty measures from the prototypical network. Our OOD score is based on the Mahalanobis distance. The Mahalanobis distance is used to measure the distance between a point and a distribution. Given a point x and a distribution with mean μ and covariance matrix Σ, the Mahalanobis distance between x and the distribution is

$$M(x;\, \mu, \Sigma) = \sqrt{(x-\mu)^T \Sigma^\dagger (x-\mu)} \tag{2}$$

Algorithm 1: Train.

```
Require: $X_{\text{train}}, y_{\text{train}}, N_{\text{episodes}}, N_{\text{query}}, S_{\text{train}}, f_\theta$
  for $1 \le b \le N_{\text{episodes}}$ do
    Sample support $\{x_s^k\}_{s=1}^{S_{\text{train}}}$ for each class $k$ and query
      examples $\{x_i\}_{i=1}^{N_{\text{query}}}$ from $X_{\text{train}}$
    Compute embeddings $f_\theta(x_i)$ and $f_\theta(x_s^k)$
    Compute prototypes $c_k = \frac{1}{S_{\text{train}}}\sum_{i=s}^{S_{\text{train}}} f_\theta(x_s^k)$.
    Calculate class probabilities as in (1) where each $d_k$ is
      the Euclidean distance
    Compute the gradient of the loss over the query
      examples; update parameters $\theta$
  end for
Ensure: $f_\theta$
```

where $\Sigma^\dagger$ denotes the pseudoinverse of Σ. Using the pseudoinverse rather than the matrix inverse handles the case in which the covariance matrix is singular. For a dataset consisting of K classes, the relative Mahalanobis distance to class k is defined to be the Mahalanobis distance to class k minus the Mahalanobis distance to the overall data distribution [31]. That is

$$M_{\text{rel}}^k(x;\, \mu_k, \mu_0, \Sigma, \Sigma_0) = M(x;\, \mu_k, \Sigma) - M(x;\, \mu_0, \Sigma_0) \tag{3}$$

where μ_k is the mean of class k, μ_0 is the mean of the dataset as a whole, Σ is the covariance matrix of each class k, and Σ_0 is the covariance matrix of the dataset as a whole. To be clear, $\mu_k = \frac{1}{K}\sum_{s=1}^{S} f_\theta(x_s^k)$ and $\mu_0 = \frac{1}{KS}\sum_{k=1}^{K}\sum_{s=1}^{S} f_\theta(x_s^k)$, while

$$\Sigma = \frac{1}{KS}\sum_{k=1}^{K}\sum_{s=1}^{S}\left(f_\theta\left(x_s^k\right) - \mu_k\right)\left(f_\theta\left(x_s^k\right) - \mu_k\right)^T \tag{4}$$

and

$$\Sigma_0 = \frac{1}{KS}\sum_{k=1}^{K}\sum_{s=1}^{S}\left(f_\theta\left(x_s^k\right) - \mu_0\right)\left(f_\theta\left(x_s^k\right) - \mu_0\right)^T. \tag{5}$$

Ren et al. [31] argue that subtracting out $M(x;\, \mu_0, \Sigma_0)$ removes the outlier score contribution along dimensions that are not discriminative between in-distribution and OOD examples. This is especially important for scenarios in which a large number of dimensions are not discriminative.

After training is complete, we use a support set of maximum size $S_{\text{cal}} = 100$ sampled from the training data (or all the training data if there are fewer than 100 examples) to fit the means and full covariance matrices used in the relative Mahalanobis distance. While the relative Mahalanobis distance allows us to rank the examples in the test set according to how outlying they are, it does not immediately suggest a threshold. To convert the raw distance into an interpretable score, we fit a univariate kernel density estimate (KDE) to the relative Mahalanobis distances (for each class) computed on a set of held-out in distribution examples X_{cal}. The density estimate will be used at test time to define an outlier score between 0 and 1 by integrating under the density to determine the proportion of calibration examples that are more outlying than the test point. We refer to this process as calibration and summarize our procedure in Algorithm 2.

Algorithm 2: Calibrate.

```
Require: $X_{\text{train}}$, $y_{\text{train}}$, $X_{\text{cal}}$, $y_{\text{cal}}$, $S_{\text{cal}}$, $f_\theta$
  Sample support $\{x_s^k\}_{s=1}^{S_{\text{cal}}}$ from the training data for each
   class $k$
  Compute $\mu_k$, for each class $k$, $\mu_0$, $\Sigma$, and $\Sigma_0$ as defined
   in Section IV-D with $S = S_{\text{cal}}$
  for $1 \leq k \leq K$ do
    Compute $\Sigma_k = \frac{1}{S_{\text{cal}}} \sum_{s=1}^{S_{\text{cal}}} (f_\theta(x_s^k) - \mu_k)(f_\theta(x_s^k)$
     $-\mu_k)^T$ (used only in testing phase)
    Compute relative Mahalanobis distance for calibration
     examples $M_{\text{rel}}^k(f_\theta(x_i^k))$
    Fit Gaussian kernel density estimate $G_k$ to the set of
     relative Mahalanobis distances $M_{\text{rel}}^k(f_\theta(x_i^k))$
  end for
Ensure: $G_k$, $\mu_k$, $\mu_0$, $\Sigma$, $\Sigma_0$, $\Sigma_k$
```

Algorithm 3: Test.

```
Require: $G_k$, $\mu_k$, $\mu_0$, $\Sigma$, $\Sigma_0$, $\Sigma_k$ $x_{\text{test}}$, $f_\theta$
  Compute predicted class $k^*$ as in (1) where
   $d_k = M(x; \mu_k, \text{diag}(\Sigma_k))$
  Compute $r = M_{\text{rel}}^{k^*}(f_\theta(x_{\text{test}}))$
  Compute $p$-value $p = \int_r^\infty G_{k^*}(r')dr'$
Ensure: $1 - p$
```

At test time, to compute a p-value for the null hypothesis that a test sample is in-distribution, we calculate the area under the density corresponding to the predicted class for distances larger than the relative Mahalanobis distance of the test point. To maintain the convention that large scores indicate OOD examples, we take one minus the p-value as our OOD score. We summarize this procedure in Algorithm 3. The computational overhead of the OOD approach is small. Computing the covariance matrices on the calibration data takes $O(E^2)$, where E is the dimension of the embedding space. Computing the relative Mahalanobis distances is also $O(E^2)$. Finally, computing each p-value from the KDE involves $|X_{\text{cal}}|$ evaluations of the normal cumulative distribution function.

V. Experiments and Results

In order to validate our framework, we perform a series of increasingly challenging tests for our ML framework.

A. Model Performance on the Dataset

First, we train the model on the provided dataset using an increasing stratified fraction of the dataset and compute the F1-scores (harmonic average of precision and recall) on a held-out test set (20% of the data). We also compute the model's ECE as a way to measure the quality of its probabilistic forecasts. The ECE measures the discrepancy between a model's predicted probability and the observed accuracy of those predictions. For more details, see [69]. For each training dataset size considered, we repeat the experiment ten times. The results for F1-score and ECE are shown in Figs. 2 and 3. Clearly, the model's performance improves as we increase the fraction of data provided, both for F1-score and ECE. It is interesting to note that we achieve quite good performance with only a limited number of training examples. To call out a specific example, using 10%[5] of the training data corresponds to 118 examples of Command & Control, 945 examples of Chat, 66 examples of File Transfer, 143 examples of Streaming, and just 19 examples of VoIP. Even with only this very small training set, we are able to obtain a micro F1-score of nearly 0.96. In addition, the calibration error using 10% of the data for training is about 0.04, which indicates that the model's confidence is fairly consistent with its accuracy. Marginal gains in calibration (lower ECE values) can be achieved via neural network ensembles as in [25] (data not shown); we speculate that the relatively shallow network, dropout, and prototypical network architecture helped to enable the low ECE values. Fig. 4 depicts the average confusion matrix for each of the ten training runs using an 80/20 train/test split, indicating that the model performs well in every category, the worst confusion being 3% of File Transfer examples being incorrectly predicted as VoIP.

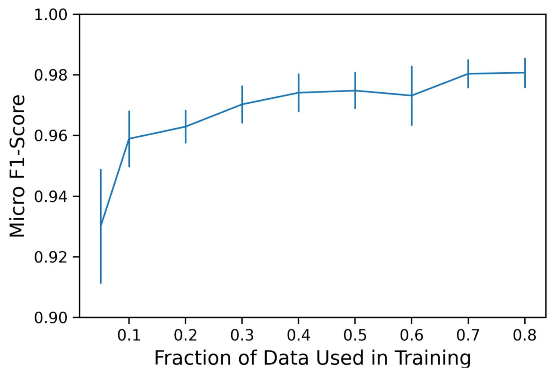

Fig. 2. Micro F1-score of the model against the fraction of data used to train the model. We plot the average F1-score over ten trials. The error bars indicate one standard deviation.

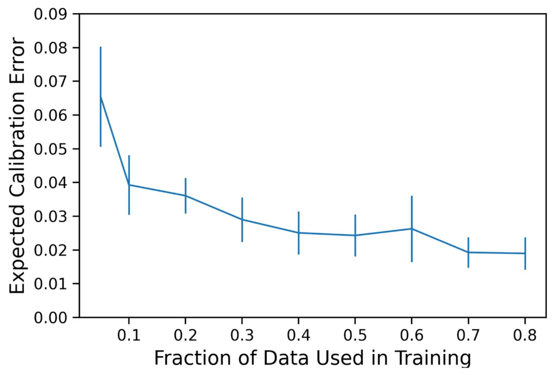

Fig. 3. ECE of the model against the fraction of data used to train the model. We plot the average F1-score over ten trials. The error bars indicate one standard deviation.

[5]Technically, only 8% of the data is used for training, as 20% of the 10% is reserved internally by our implementation for generating the p-value distributions.

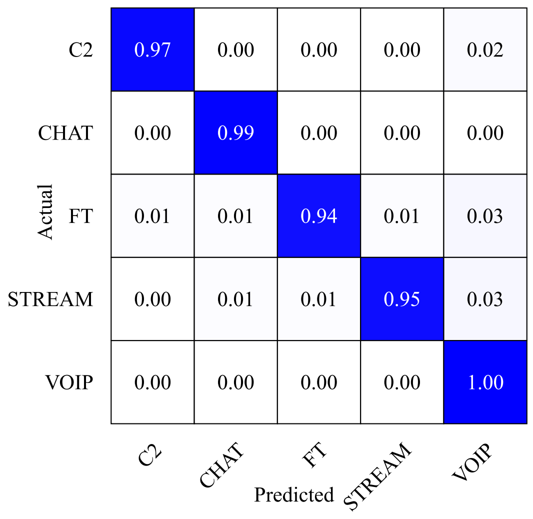

Fig. 4. Average confusion matrix for ten runs using a randomized 80/20 train/test split (FT indicates File Transfer).

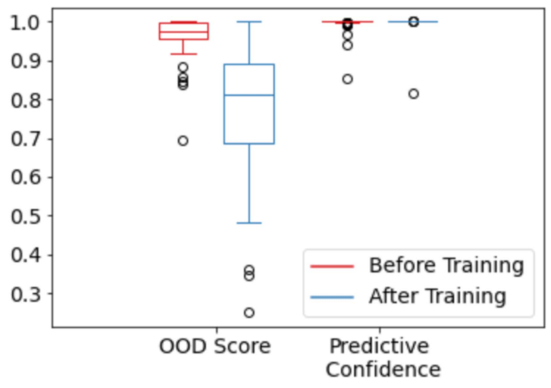

Fig. 5. Model uncertainties for SMB from an enterprise network before (using a pretrained model from our dataset) and after retraining (adding 91 labeled examples).

B. Model Performance on UTMobileNet2021

In order to evaluate our approach on a greater variety of labels, we evaluated our model on the UTMobileNet2021 encrypted traffic dataset, which contains data from 17 unique traffic applications. Using the above settings, we found that our approach retained 80% accuracy across all applications using an 80/20 train/test split, which is on par with their best reported results [22]. We also tested the effect of adding multiple labels to nonoverlapping applications from our own dataset, and we found that for 5, 10, 15, and 20 application labels, our approach had overall accuracies of 93%, 87%, 80%, and 80%, respectively, indicating that our per-application performance degrades a bit but remains competitive. For more details, see Table VII in the Appendix.

C. Model Performance in a New Environment

Having established the "usual" ML metrics, we then turn to a more challenging problem of validation outside of the training dataset. To do these experiments, we used WireShark to capture network traffic on a live enterprise network and then tested our model on this new traffic. We consider three increasingly challenging scenarios. First, can the model recognize traffic from an existing application in an existing category? Second, can the model recognize new applications in an existing category? Finally, can the model detect data that is OOD (comes from a new category)?

To answer the first question, we collected 40 additional instances of YouTube (an application in the Streaming category). We trained the model on 80% of our training dataset and then tested on the YouTube data. We repeated the experiment ten times. The results were quite good. For the YouTube traffic, the model predicted the correct class 40 out of 40 times in nine of the trials and 39 out of 40 times in the other trial, only predicting one of the examples to have an OOD score above 0.95 on average.[6] Thus, even though the YouTube traffic was collected on a different network than the one that the training data came from, the model was still able to recognize its class (Streaming). This is an encouraging result, although we do not rely on this transferability holding true for all applications in the training set, as will be demonstrated next.

To answer the second question, we collected data from three applications that do not appear in the training data (Avaya, Samba (SMB), and the Code42 automated file backup program). In our labeling schema, we consider Avaya to be a VoIP application and SMB and Code42 to be File Transfer. We collected 123 examples of Avaya, 141 examples of SMB, and 202 examples of Code42. As before, we trained on 80% of our training dataset (for ten trials) and tested on the new applications. Avaya was easily recognized by the model as VoIP traffic, correctly labeling all 123 examples in each trial, although the model was less certain about them, assigning about half of them OOD scores above 0.95. SMB and Code42, however, were not initially recognized as File Transfer. Encouragingly, the model tended to give these examples a high OOD score. Specifically, across the ten trials, an average of 78% of the SMB traffic received an OOD score above 0.95, and an average of about 63% of the file backup program received an OOD score above 0.95.

To see whether the model could learn to classify SMB and Code42 as File Transfer, given a few training examples, we investigate retraining the model. For the case of SMB, we add 91 examples to our training set and test on the remaining 50. After retraining, the model is able to correctly classify an average of 92% of the test samples. In addition, the OOD scores are much lower. After retraining, only an average of 10% of examples had OOD scores above 0.95 (left side of Fig. 5). Fig. 5 also provides the predictive confidences before and after retraining with examples of SMB, which refers to the probability associated with the class prediction (the highest probability class among known classes). The distributions exhibit high confidences both before and after retraining. This indicates that it is important

[6]We have selected OOD scores above 0.95 as a threshold of interest, as it corresponds to a p-value of 0.05 or lower for rejecting the null hypothesis that the test samples are in-distribution.

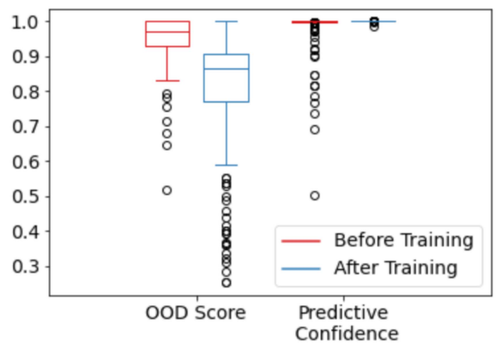

Fig. 6. Model uncertainties for Code42 from an enterprise network before (using a pretrained model from our dataset) and after retraining (adding 42 labeled examples).

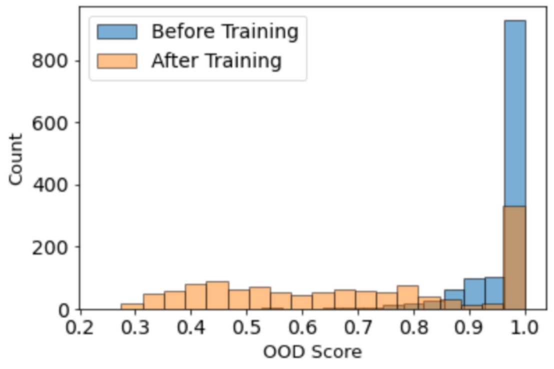

Fig. 7. Distribution of Zoom OOD scores on a held-out validation set of 128 examples (sum of ten runs) using the pretrained model from our dataset ("Before Training") and after retraining with 195 labeled examples ("After Training").

to not rely solely on the predictive confidences for uncertainty quantification, as they are essentially meaningless when applied to OOD test examples.

In the case of Code42, we add 42 examples to our training set and test on the remaining 160. The model now correctly classifies an average of 98% of the test samples. As with SMB, the OOD scores are now much lower as well, with only an average of 5% with scores above 0.95. In Fig. 6, we show OOD score and predictive confidence before and after training with examples of Code42.

To answer the final question, we collected 323 examples of Zoom, which is a new application of a new category one could label as VTC. Since VTC is a new class, we expect that the model will give high OOD scores to the Zoom examples. This ended up being the case, as an average of 78% of examples had an OOD score above 0.95. We then retrained our model with 195 Zoom examples added to the training set (with the new VTC label). After retraining, the fraction of OOD scores above 0.95 is reduced to 26%. Fig. 7 depicts the distribution of OOD scores before and after training on a held-out validation set of 128 Zoom examples; the post-training results are more uniform, as expected. Across ten trials, the model correctly labeled an average of 73% of the validation examples as Zoom, generally confusing the errors as VoIP. We note that this accuracy is lower than the previous examples; however, about 97% of the incorrectly labeled examples still have OOD scores above 0.95,

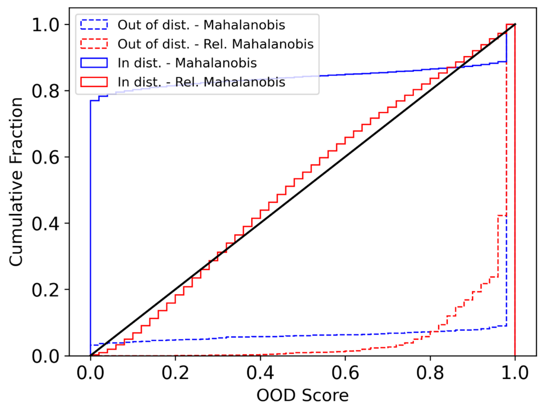

Fig. 8. Empirical cumulative distribution function of OOD scores on a held-out set of 116 Zoom examples (OOD data) and a held-out set of in-distribution data. OOD scores are produced using the squared Mahalanobis distance and χ^2 distribution as well as the relative Mahalanobis distance and a KDE fit on calibration examples.

indicating that these errors can easily be screened out due to their model uncertainty. It perhaps indicates that Zoom traffic behaves differently depending upon its mode of usage (screen sharing, video, or audio-only); it may be possible to cluster self-similar OOD examples in the embedding space to assist analysts with new labeling.

D. *Model Performance on Padded Packets*

In order to address any concerns about the model's resilience to encryption protocols that pad packets to be of uniform size, we repeated the initial testing on the collected dataset but masked the sizes of all packets to be 1500 bytes before feature generation. Upon retraining ten times with 80/20 train/test splits, the model retained an average F1-score of 0.971 ± 0.008 on the dataset. This is a marginal drop from the raw unpadded data at 0.982 ± 0.004. These results indicate that our model is not much affected by the effective removal of packet size information. In addition, this result presents initial evidence that packet padding alone may not offer substantial privacy benefits, at least at the level of granularity of traffic categories.

E. *Baseline OOD Performance*

Finally, we compare our OOD approach to a baseline approach. In our baseline, rather than fitting a KDE on a calibration set to compute an outlier score, we note that the squared Mahalanobis distance follows a χ^2 distribution with 64 (the dimension of our embedding space) degrees of freedom. We use this distribution to compute p-values for new observations. As before, our OOD score is simply one minus the p-value. This assumes, of course, that the embeddings are actually normally distributed according to the model in Section IV. In Fig. 8, we evaluate our model on a held-out test set of in distribution data as well as 116 examples of Zoom traffic (which is OOD).

We first examine the in-distribution data. Notice that the OOD scores based on the squared Mahalanobis distance and χ^2 distribution tend to be either close to 0 or close to 1. This stands

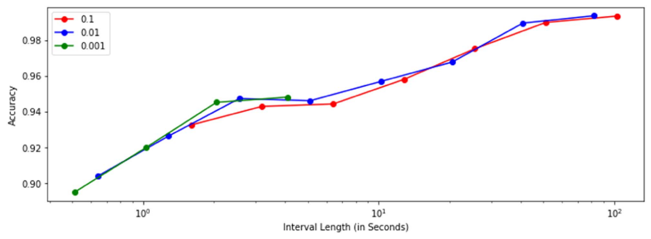

Fig. 9. Study in the sensitivity of the overall ML accuracy with respect to different choices of window length T and three time-bin sizes Δt, 0.1, 0.01, and 0.001 second. Note the semilog x-scale.

in contrast to the scores produced by the relative Mahalanobis distance and the KDE, which are roughly uniform. Thus, if we set the OOD detection threshold to α, then we expect to filter out $(1 - \alpha)$% of the in-distribution data, when using the approach proposed in this article.

For OOD data, both the approaches tend to assign highs scores, for the most part. The squared-Mahalanobis-distance-based score again tends to be either very close to 1 or very close to 0, while the relative-Mahalanobis-based score varies more smoothly.

VI. Discussion

The results of the previous section appear very encouraging, particularly since: 1) the framework performs effectively, achieving high accuracy without requiring rigorous feature optimizations and on only timing and size information; 2) the dataset and model extended in part to another network for YouTube and VoIP; and 3) the model "degraded gracefully" when confronted with novel application data and was quick to retrain on limited data. We envision this framework to be used in a tool that would allow network analysts to systematically build insight for applications of interest in conjunction with other analysis techniques (malware signatures, website fingerprinting, etc.) to supplement where they fail.

Shorter window lengths could hypothetically provide more label opportunities over the entire observation period, but arbitrarily short window lengths would reduce accuracy (additional analysis is provided in Fig. 9). Longer window lengths within a VPN would also be more likely to merge multiple application types into the same sample, and our approach only predicts one application per time window. We also note that this framework does not leverage continuity between time windows as of yet, and certain connections can tend to be shorter (loading a web page) or longer (streaming movies).

Our approach can also be attacked and degraded by manipulating the traffic, e.g., by adding "cover packets" to flood connections or mask timings, re-encoding traffic through another protocol [79] or by leveraging adversarial ML [80], [81]. However, our approach still retains an advantage in that it forces an adversary to work harder to obfuscate the nature of their connections. We also note that our approach could be used to "red-team" censorship evasion development by incorporating uncertainty in addition to the standard accuracy metrics.

We finally comment that although our dataset does not comprehensively capture the diversity of Internet applications, it was enough to "generate intuition" for a generic ML framework to predict time-windowed encrypted application traffic on a different network, and we have demonstrated the importance of treating uncertainty as a first-class feature to enable a systematic extension of the dataset onto other networks or application categories of interest.

VII. Conclusion

In this article, we presented a labeled dataset of VPN-encrypted and unencrypted network traffic to facilitate ML applications seeking to leverage packet size and timing information to predict the traffic's most likely application categories. We discussed potential issues and possible workarounds for a popular alternative dataset, which motivated the organization of our dataset. We showed an ML framework leveraging wavelet-based and statistical features on discrete sampled time windows in a prototypical network architecture that achieved high F1-scores (0.98) on the labeled dataset. Furthermore, we showed that the architecture could be trained on a small random fraction of the dataset, achieving an accuracy of 0.96 on VOIP using fewer than 14 min of data. In addition, we demonstrated low calibration errors for predictive confidence (3%). We also tested the model on application traffic (YouTube and a new VoIP application) from an enterprise network and found it extended accurately.

We used a technique for constructing an OOD score based on p-values of the relative Mahalanobis distance on in-distribution data to reliably indicate model degradation in a series of increasingly challenging experiments. In all the cases, the model retrained quickly with relatively few examples to learn to incorporate the new applications into a known category or assign them to a new one. We additionally demonstrated that the framework retains very high performance (F1-score of 0.97) even when all packet sizes are uniform, such as occurs in encryption protocols that ensure all packets are padded to the same size before encryption.

Future work should incorporate the identification of concurrent applications inside a VPN (multilabel predictions), as our approach currently only predicts one class per time interval. Another interesting avenue of research would be to look for structure in OOD examples, perhaps via clustering, and suggest possible new classes to the user. Finally, future work should consider adversarial methods that attack the framework, such as flooding connections with dummy packets, re-encoding traffic through other protocols, or adversarial ML techniques that target both the accuracy and the uncertainty scores.

Appendix
Filename-to-Category Mappings

We use the mappings shown in Table VI to assign class labels to each PCAP file. If any of the words in the "File Keyword" column occurs in the column name, then we assign the label in the left column.

TABLE VI
FILENAME-TO-CATEGORY LABELING SCHEME

Traffic Category	**File Keyword**
STREAMING	vimeo, netflix, youtube
CHAT	chat
C2	ssh, rdp
FILE_TRANSFER	sftp, rsync, scp
VOIP	voip

WAVELET-BASED FEATURES

This section provides details on how to calculate the wavelet-based features extracted from the wavelet-transformed connection sizes in the forward and backward directions for frequency bands 0–12. The wavelet-based features include the relative wavelet energy, Shannon entropy, and absolute means and standard deviations of the detail coefficients.

The continuous wavelet transform coefficients are defined as

$$d(a,b) = \frac{1}{\sqrt{|a|}} \int_0^T x(t)\overline{\psi}\left(\frac{t-b}{a}\right) dt \tag{6}$$

where $x(t)$ is the time-dependent signal over the observation period $[0, T]$ (for instance, bytes transferred in a flow during a sample interval), $\overline{\psi}(\frac{t-b}{a})$ is the wavelet function, a is the scale factor, and b is the translation factor. This article employs the discrete wavelet transformation, which restricts a and b to discrete values that balance temporal and frequency resolution while reducing computational burden. We employ the shift-invariant (stationary) wavelet transformation to ensure that the resulting features are insensitive to signal translations, otherwise known as the "algorithm a-trous" [82], [83].

The time-dependent signal, $x(t)$, required for the wavelet transform is represented by a discrete vector over an observation period from $[0, T]$ that is discretized into N smaller time bins of width Δt. Within each Δt time interval, all of the packet statistics are aggregated. For instance, let t_j and y_j be the packet time stamp and size for packet j in a series of J flow-aggregated packets with $t_0 = 0$, respectively. Then, a vector version of $x(t)$, appropriate for the wavelet transform, is defined by $x_n(t)$, where $T = N\Delta t$, $n \in [0, N-1]$, and

$$x_n(t) = \sum_{j=0}^{J} y_j \chi_j(t)\,;\ \chi_j(t) = \begin{cases} 1, & n\Delta t \le t_j < (n+1)\Delta t \\ 0, & \text{else} \end{cases}.$$

We enforce that the length of the vector x_n be an integer power of 2 ($N = 2^K$ for some K) by judiciously selecting Δt and T or by reflecting the signal in our pipeline. Applying (6) for fixed values of $a \in 2^k$ and $b \in x_n$ gives a rectangular coefficient matrix of the form $d_{n,k}$, where $n \in [0, N-1]$ indexes the coefficients in time and $k \in [0, K]$ indexes the coefficients in frequency band.

Relative Wavelet Energy

For each frequency band k, the relative wavelet energy ρ_k is given by

$$\rho_k = \frac{E_k}{E_{\text{total}}}, \quad \text{for } k = 0, 1, \ldots, K \tag{7}$$

TABLE VII
UTMOBILENET2021 MULTIPLE CLASSES' STUDY

Number of Classes	Applications	Accuracy
5-class	hangout, youtube, skype, twitter, hulu	93%
10-class	hangout, instagram, google-maps, dropbox, gmail, spotify, twitter, hulu, facebook, reddit	87%
15-class	skype, facebook, pandora, spotify, twitter, gmail, reddit, messenger, hulu, netflix, hangout, instagram, youtube, google-maps, pinterest	80%
20-class	dropbox, facebook, gmail, google-drive, google-maps, hangout, hulu, instagram, messenger, netflix, pandora, pinterest, reddit, skype, spotify, twitter, youtube, **zoiper5, rdp, ssh**	80%

Bold classes indicate applications supplemented from our dataset.

where E_k is the energy of frequency band k

$$E_k = \sum_n |d_{n,k}|^2, \quad \text{for } n = 1, 2, \ldots, N \tag{8}$$

and E_{total} is the total energy of the signal after wavelet decomposition

$$E_{\text{total}} = \sum_k E_k, \quad \text{for } k = 0, 1, \ldots, K\,. \tag{9}$$

Wavelet Shannon Entropy

For each frequency band k, the wavelet Shannon entropy S_k is given by

$$S_k = -\sum_n p_n \log(p_n), \quad \text{for } n = 1, 2, \ldots, N \tag{10}$$

where p_n is the fraction of energy of the nth detail coefficient

$$p_n = \frac{d_{n,k}^2}{E_k}\,. \tag{11}$$

Detail Coefficient Statistics

For each frequency band k, the absolute mean of the detail coefficients μ_k is given by

$$\mu_k = \frac{1}{N}\sum_n |d_{n,k}|, \quad \text{for } n = 1, 2, \ldots, N \tag{12}$$

and the standard deviation of the detail coefficients σ_n is given by

$$\sigma_k = \sqrt{\frac{1}{N}\sum_n (\mu_k - d_{n,k})^2}, \quad \text{for } n = 1, 2, \ldots, N\,. \tag{13}$$

Fig. 9 depicts the results of a range of experiments using three time bins (Δt of 1, 10, and 100 ms) and variable time windows, T. The general trend is that the longer the sample window, T, the more accurate the model. However, this has the tradeoff that longer windows require longer vectors, particularly for small time bins (note that we capped the number of wavelet bands at 13 to limit feature-computation requirements). The second-to-rightmost blue datapoint represents $(\Delta t, T)$=$(0.01, 40.96)$, the selected value for the analysis in this article, as this neared the maximum accuracy and used an acceptable vector length for computing the wavelet features.

UTMobileDataNet2021 Study

Table VII provides the random classes used to study the effects of adding additional classes on the accuracy of our model. We note that we did not attempt to optimize which applications were chosen for the few-application study, and that the applications selected are random subsample of the UTMobileNet2021 dataset [22], with the exception of the 20-class case, which uses three classes from our dataset (zoiper5, rdp, and ssh). We also note that Heng et al.'s paper [22] appears to indicate that there are only 16 classes in this dataset, but this excludes their Skype data, which accounts for the "extra" label.

Acknowledgment

Any opinions, findings, conclusions, or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of U.S. Cyber Command.

Distribution Statement: Approved for public release. Distribution is unlimited.

References

[1] HTTPS Encryption on the Web, Nov. 2021. Accessed: Nov. 15, 2021. [Online]. Available: https://transparencyreport.google.com/https/overview?hl=en

[2] E. Ahlm, J. D'Hoinne, L. Orans, and A. Hils, "Predicts 2017: Network and gateway security," *Gartner Res., Stamford, CT, USA*, Tech. Rep., 2016. [Online]. Available: https://www.gartner.com/en/documents/3542117/predicts-2017-network-and-gateway-security

[3] C. Warfield and J. Hall, "WatchGuard threat lab reports 91.5% of malware arrived over encrypted connections in Q2 2021," Sep. 2021. Accessed: Nov. 15, 2021. [Online]. Available: https://www.watchguard.com/wgrd-news/press-releases/watchguard-threat-lab-reports-915-malware-arrived-over-encrypted

[4] V. Rimmer, D. Preuveneers, M. Juarez, T. Van Goethem, and W. Joosen, "Automated website fingerprinting through deep learning," in *Proc. Netw. Distrib. Syst. Secur. Symp.*, 2017, pp. 1–15.

[5] N. P. Hoang, A. A. Niaki, P. Gill, and M. Polychronakis, "Domain name encryption is not enough: Privacy leakage via IP-based website fingerprinting," 2021, *arXiv:2102.08332*.

[6] Obfuscated servers, Nov. 2021. Accessed: Nov. 15, 2021. [Online]. Available: https://nordvpn.com/features/obfuscated-servers/

[7] T. Perrin, "The noise protocol framework," Jul. 2018. Accessed: Nov. 15, 2021. [Online]. Available: http://www.noiseprotocol.org/noise.html

[8] R. Alshammari and A. N. Zincir-Heywood, "Machine learning based encrypted traffic classification: Identifying SSH and Skype," in *Proc. IEEE Symp. Comput. Intell. Secur. Defense Appl.*, 2009, pp. 1–8.

[9] 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 *Proc. 2nd Int. Conf. Inf. Syst. Secur. Privacy*, 2016, pp. 407–414.

[10] Y. Liang, Y. Xie, X. Fei, X. Tan, and H. Ma, "Content recognition of network traffic using wavelet transform and CNN," in *Proc. Int. Conf. Mach. Learn. Cyber Secur.*, 2019, pp. 224–238.

[11] M. Abbasi, A. Shahraki, and A. Taherkordi, "Deep learning for network traffic monitoring and analysis (NTMA): A survey," *Comput. Commun.*, vol. 170, pp. 19–41, 2021.

[12] W. Wang, M. Zhu, J. Wang, X. Zeng, and Z. Yang, "End-to-end encrypted traffic classification with one-dimensional convolution neural networks," in *Proc. IEEE Int. Conf. Intell. Secur. Informat.*, 2017, pp. 43–48.

[13] W. Wang, M. Zhu, X. Zeng, X. Ye, and Y. Sheng, "Malware traffic classification using convolutional neural network for representation learning," in *Proc. Int. Conf. Inf. Netw.*, 2017, pp. 712–717.

[14] S. Rezaei and X. Liu, "How to achieve high classification accuracy with just a few labels: A semisupervised approach using sampled packets," in *Proc. 19th Ind. Conf. Adv. Data Mining—Appl. Theor. Aspects*, 2019, pp. 28–42.

[15] M. Lotfollahi, M. J. Siavoshani, R. S. H. Zade, and M. Saberian, "Deep packet: A novel approach for encrypted traffic classification using deep learning," *Soft Comput.*, vol. 24, no. 3, pp. 1999–2012, 2020.

[16] I. Akbari et al., "A look behind the curtain: Traffic classification in an increasingly encrypted web," in *Proc. ACM SIGMETRICS/Int. Conf. Meas. Model. Comput. Syst.*, 2021, pp. 23–24.

[17] C. Liu, L. He, G. Xiong, Z. Cao, and Z. Li, "FS-Net: A flow sequence network for encrypted traffic classification," in *Proc. IEEE Conf. Comput. Commun.*, 2019, pp. 1171–1179.

[18] G. Aceto, D. Ciuonzo, A. Montieri, and A. Pescap, "MIMETIC: Mobile encrypted traffic classification using multimodal deep learning," *Comput. Netw.*, vol. 165, 2019, Art. no. 106944.

[19] Y. Wang, X. Yun, Y. Zhang, C. Zhao, and X. Liu, "A multi-scale feature attention approach to network traffic classification and its model explanation," *IEEE Trans. Netw. Service Manage.*, vol. 19, no. 2, pp. 875–889, Jun. 2022.

[20] G. Aceto, D. Ciuonzo, A. Montieri, and A. Pescap, "DISTILLER: Encrypted traffic classification via multimodal multitask deep learning," *J. Netw. Comput. Appl.*, vols. 183/184, 2021, Art. no. 102985.

[21] G. Aceto, D. Ciuonzo, A. Montieri, V. Persico, and A. Pescap, "MIRAGE: Mobile-app traffic capture and ground-truth creation," in *Proc. 4th Int. Conf. Comput., Commun. Secur.*, 2019, pp. 1–8.

[22] Y. Heng, V. Chandrasekhar, and J. G. Andrews, "UTMobileNetTraffic2021: A labeled public network traffic dataset," *IEEE Netw. Lett.*, vol. 3, no. 3, pp. 156–160, Sep. 2021.

[23] F. Zaki, A. Gani, H. Tahaei, S. Furnell, and N. B. Anuar, "Grano-GT: A granular ground truth collection tool for encrypted browser-based internet traffic," *Comput. Netw.*, vol. 184, 2021, Art. no. 107617.

[24] K. Highnam, K. Arulkumaran, Z. Hanif, and N. R. Jennings, "BETH dataset: Real cybersecurity data for anomaly detection research," in *Proc. Workshop Uncertainty Robustness Deep Learn.*, 2021, p. 8.

[25] K. E. Brown, F. A. Bhuiyan, and D. A. Talbert, "Uncertainty quantification in multimodal ensembles of deep learners," in *Proc. 33rd Int. Flairs Conf.*, 2020, pp. 422–427.

[26] D. Engel et al., "UQ methods for HPDA and cybersecurity models, data, and use cases," Pacific Northwest Nat. Lab., Richland, WA, USA, Tech. Rep. PNNL-24901, 2015.

[27] M. C. Darling and D. J. Stracuzzi, "Toward uncertainty quantification for supervised classification," *Sandia Nat. Lab.*, Albuquerque, NM, USA, Tech. Rep. SAND2018-0032, 2018.

[28] A. Nascita, A. Montieri, G. Aceto, D. Ciuonzo, V. Persico, and A. Pescap, "XAI meets mobile traffic classification: Understanding and improving multimodal deep learning architectures," *IEEE Trans. Netw. Service Manage.*, vol. 18, no. 4, pp. 4225–4246, Dec. 2021.

[29] E. Papadogiannaki and S. Ioannidis, "A survey on encrypted network traffic analysis applications, techniques, and countermeasures," *ACM Comput. Surv.*, vol. 54, no. 6, pp. 1–35, 2021.

[30] J. Snell, K. Swersky, and R. Zemel, "Prototypical networks for few-shot learning," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2017, pp. 4077–4087.

[31] J. Ren, S. Fort, J. Liu, A. Guha Roy, S. Padhy, and B. Lakshminarayanan, "A simple fix to Mahalanobis distance for improving near-OOD detection," in *Proc. Uncertainty Deep Learn. Workshop/Int. Conf. Mach. Learn.*, 2021, pp. 1–8.

[32] S. Kent, "IP encapsulating security payload (ESP)," *Internet Requests Comments*, RFC Editor, Tech. Rep. 4303, 2005. [Online]. Available: https://www.rfc-editor.org/info/rfc4303

[33] M. C. Tschantz, S. Afroz, and V. Paxson, "SoK: Towards grounding censorship circumvention in empiricism," in *Proc. IEEE Symp. Secur. Privacy*, 2016, pp. 914–933.

[34] N. A. Khater and R. E. Overill, "Network traffic classification techniques and challenges," in *Proc. 10th Int. Conf. Digit. Inf. Manage.*, 2015, pp. 43–48.

[35] A. C. Callado et al., "A survey on internet traffic identification," *IEEE Commun. Surv. Tuts.*, vol. 11, no. 3, pp. 37–52, 3rd Quarter 2009.

[36] M. Roughan, S. Sen, O. Spatscheck, and N. Duffield, "Class-of-service mapping for QoS: A statistical signature-based approach to IP traffic classification," in *Proc. 4th ACM SIGCOMM Conf. Internet Meas.*, 2004, pp. 135–148.

[37] A. W. Moore and K. Papagiannaki, "Toward the accurate identification of network applications," in *Proc. Int. Workshop Passive Act. Netw Meas.*, 2005, pp. 41–54.

[38] L. Deri, M. Martinelli, T. Bujlow, and A. Cardigliano, "nDPI: Open-source high-speed deep packet inspection," in *Proc. Int. Wireless Commun. Mobile Comput. Conf.*, 2014, pp. 617–622.

[39] S. Sen, O. Spatscheck, and D. Wang, "Accurate, scalable in-network identification of P2P traffic using application signatures," in *Proc. 13th Int. Conf. World Wide Web*, 2004, pp. 512–521.

[40] T. Choi et al., "Content-aware internet application traffic measurement and analysis," in *Proc. IEEE/IFIP Netw. Oper. Manage. Symp.*, 2004, pp. 511–524.

[41] B. Anderson and D. McGrew, "Identifying encrypted malware traffic with contextual flow data," in *Proc. ACM Workshop Artif. Intell. Secur.*, 2016, pp. 35–46.

[42] A. McGregor, M. Hall, P. Lorier, and J. Brunskill, "Flow clustering using machine learning techniques," in *Proc. Int. Workshop Passive Act. Netw. Meas.*, 2004, pp. 205–214.

[43] S. Zander, T. Nguyen, and G. Armitage, "Automated traffic classification and application identification using machine learning," in *Proc. IEEE Conf. Local Comput. Netw. 30th Anniversary*, 2005, pp. 250–257.

[44] W. Li, M. Canini, A. W. Moore, and R. Bolla, "Efficient application identification and the temporal and spatial stability of classification schema," *Comput. Netw.*, vol. 53, no. 6, pp. 790–809, 2009.

[45] R. Yuan, Z. Li, X. Guan, and L. Xu, "An SVM-based machine learning method for accurate internet traffic classification," *Inf. Syst. Front.*, vol. 12, no. 2, pp. 149–156, 2010.

[46] M. Soysal and E. G. Schmidt, "Machine learning algorithms for accurate flow-based network traffic classification: Evaluation and comparison," *Perform. Eval.*, vol. 67, no. 6, pp. 451–467, 2010.

[47] H. Zhang, G. Lu, M. T. Qassrawi, Y. Zhang, and X. Yu, "Feature selection for optimizing traffic classification," *Comput. Commun.*, vol. 35, no. 12, pp. 1457–1471, 2012.

[48] A. Fahad, Z. Tari, I. Khalil, A. Almalawi, and A. Y. Zomaya, "An optimal and stable feature selection approach for traffic classification based on multi-criterion fusion," *Future Gener. Comput. Syst.*, vol. 36, pp. 156–169, 2014.

[49] B. Yamansavascilar, M. A. Guvensan, A. G. Yavuz, and M. E. Karsligil, "Application identification via network traffic classification," in *Proc. Int. Conf. Comput., Netw. Commun.*, 2017, pp. 843–848.

[50] A. H. Lashkari, G. Draper-Gil, M. S. I. Mamun, and A. A. Ghorbani, "Characterization of Tor traffic using time based features," in *Proc. Int. Conf. Inf. Syst. Secur. Privacy*, 2017, pp. 253–262.

[51] H. Shi, H. Li, D. Zhang, C. Cheng, and W. Wu, "Efficient and robust feature extraction and selection for traffic classification," *Comput. Netw.*, vol. 119, pp. 1–16, 2017.

[52] A. C. Gilbert, W. Willinger, and A. Feldmann, "Scaling analysis of conservative cascades, with applications to network traffic," *IEEE Trans. Inf. Theory*, vol. 45, no. 3, pp. 971–991, Apr. 1999.

[53] S. Molnár et al., "Source characterization in broadband networks," Tech. Univ. Budapest, Budapest, Hungary, Interim Report COST-257, 1999.

[54] R. H. Riedi, M. S. Crouse, V. J. Ribeiro, and R. G. Baraniuk, "A multifractal wavelet model with application to network traffic," *IEEE Trans. Inf. Theory*, vol. 45, no. 3, pp. 992–1018, Apr. 1999.

[55] R. H. Riedi and W. Willinger, "Toward an improved understanding of network traffic dynamics," in *Self-Similar Network Traffic and Performance Evaluation*. Hoboken, NJ, USA: Wiley, 2000, pp. 507–530.

[56] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. New York, NY, USA: Springer, 2009.

[57] S. Roy, T. Shapira, and Y. Shavitt, "Fast and lean encrypted Internet traffic classification," *Comput. Commun.*, vol. 186, pp. 166–173, 2022.

[58] T. Shapira and Y. Shavitt, "FlowPic: A generic representation for encrypted traffic classification and applications identification," *IEEE Trans. Netw. Service Manage.*, vol. 18, no. 2, pp. 1218–1232, Jun. 2021.

[59] S. Bhat, D. Lu, A. Kwon, and S. Devadas, "Var-CNN: A data-efficient website fingerprinting attack based on deep learning," *Proc. Privacy Enhancing Technol.*, vol. 2019, no. 4, pp. 292–310, 2019.

[60] M. S. Rahman, P. Sirinam, N. Mathews, K. G. Gangadhara, and M. Wright, "Tik-tok: The utility of packet timing in website fingerprinting attacks," in *Proc. Privacy Enhancing Technol. Symp.*, 2020, pp. 5–24.

[61] M. Juarez, S. Afroz, G. Acar, C. Diaz, and R. Greenstadt, "A critical evaluation of website fingerprinting attacks," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2014, pp. 263–274.

[62] K. Lee, K. Lee, H. Lee, and J. Shin, "A simple unified framework for detecting out-of-distribution samples and adversarial attacks," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2018, vol. 31, pp. 7167–7177.

[63] K.-C. Wang, P. Vicol, E. Triantafillou, C.-C. Liu, and R. Zemel, "Out-of-distribution detection in few-shot classification," 2020. [Online]. Available: https://openreview.net/forum?id=BygNAa4YPH

[64] H. Maciejewski, T. Walkowiak, and K. Szyc, "Out-of-distribution detection in high-dimensional data using mahalanobis distance—Critical analysis," in *Computational Science*, D. Groen, C. de Mulatier, M. Paszynski, V. V. Krzhizhanovskaya, J. J. Dongarra, and P. M.A. Sloot, Eds. Cham, Switzerland: Springer, 2022, pp. 262–275.

[65] Y. Ming, Y. Sun, O. Dia, and Y. Li, "CIDER: Exploiting hyperspherical embeddings for out-of-distribution detection," 2022, *arXiv:2203.04450*.

[66] H. Huang, Z. Li, L. Wang, S. Chen, B. Dong, and X. Zhou, "Feature space singularity for out-of-distribution detection," in *Proc. Workshop Artif. Intell. Saf.*, 2021, pp. 1–9.

[67] Y. Sun, Y. Ming, X. Zhu, and Y. Li, "Out-of-distribution detection with deep nearest neighbors," in *Proc. Int. Conf. Mach. Learn.*, 2022, pp. 20827–20840.

[68] E. Techapanurak, M. Suganuma, and T. Okatani, "Hyperparameter-free out-of-distribution detection using cosine similarity," in *Proc. Asian Conf. Comput. Vis.*, 2020, pp. 53–69.

[69] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger, "On calibration of modern neural networks," in *Proc. 34th Int. Conf. Mach. Learn.*, 2017, pp. 1321–1330.

[70] B. Lakshminarayanan, A. Pritzel, and C. Blundell, "Simple and scalable predictive uncertainty estimation using deep ensembles," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2017, pp. 6405–6416.

[71] A. G. Wilson and P. Izmailov, "Bayesian deep learning and a probabilistic perspective of generalization," in *Proc. 34th Int. Conf. Neural Inf. Process. Syst.*, 2020, pp. 4697–4708.

[72] J. Yang, K. Zhou, Y. Li, and Z. Liu, "Generalized out-of-distribution detection: A survey," 2021, *arXiv:2110.11334*.

[73] D. Hendrycks and K. Gimpel, "A baseline for detecting misclassified and out-of-distribution examples in neural networks," in *Proc. 5th Int. Conf. Learn. Represent.*, 2017, pp. 1–12.

[74] S. Liang, Y. Li, and R. Srikant, "Enhancing the reliability of out-of-distribution image detection in neural networks," in *Proc. 6th Int. Conf. Learn. Represent.*, 2018, pp. 1–15.

[75] A. Malinin and M. Gales, "Predictive uncertainty estimation via prior networks," in *Proc. 32nd Int. Conf. Neural Inf. Process. Syst.*, 2018, pp. 7047–7058.

[76] T. Tsiligkaridis, "Information aware max-norm Dirichlet networks for predictive uncertainty estimation," *Neural Netw.*, vol. 135, pp. 105–114, 2021.

[77] freeCodeCamp.org, "freeCodeCamp/gitter-history," 2016. [Online]. Available: https://github.com/freeCodeCamp/gitter-history

[78] G. R. Lee, R. Gommers, F. Waselewski, K. Wohlfahrt, and A. O'Leary, "PyWavelets: A python package for wavelet analysis," *J. Open Source Softw.*, vol. 4, no. 36, 2019, Art. no. 1237.

[79] P. K. Sharma, D. Gosain, and S. Chakravarty, "Camoufler: Accessing the censored web by utilizing instant messaging channels," in *Proc. ACM Asia Conf. Comput. Commun. Secur.*, 2021, pp. 147–161.

[80] M. Nasr, A. Bahramali, and A. Houmansadr, "Defeating DNN-based traffic analysis systems in real-time with blind adversarial perturbations," in *Proc. 30th USENIX Secur. Symp.*, 2021, pp. 2705–2722.

[81] K. Bock, G. Hughey, X. Qiang, and D. Levin, "Geneva: Evolving censorship evasion strategies," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2019, pp. 2199–2214.

[82] M. Holschneider, R. Kronland-Martinet, J. Morlet, and P. Tchamitchian, "A real-time algorithm for signal analysis with the help of the wavelet transform," in *Wavelets*. New York, NY, USA: Springer, 1990, pp. 286–297.

[83] D. Percival and A. Waldern, *Wavelet Methods for Time Series Analysis*. Cambridge, U.K.: Cambridge Univ. Press, 2000.