Detecting VoIP Data Streams: Approaches Using Hidden Representation Learning

Maya Kapoor[1], Michael Napolitano[1], Jonathan Quance[1], Thomas Moyer[2], Siddharth Krishnan[2]

[1] Defense and Intelligence Sector, Parsons Corporation
[2] University of North Carolina at Charlotte
maya.kapoor@parsons.com, michael.napolitano@parsons.com, nic.quance@parsons.com,
skrishnan@uncc.edu, tmoyer2@uncc.edu

Abstract

The use of voice-over-IP technology has rapidly expanded over the past several years, and has thus become a significant portion of traffic in the real, complex network environment. Deep packet inspection and middlebox technologies need to analyze call flows in order to perform network management, load-balancing, content monitoring, forensic analysis, and intelligence gathering. Because the session setup and management data can be sent on different ports or out of sync with VoIP call data over the Real-time Transport Protocol (RTP) with low latency, inspection software may miss calls or parts of calls. To solve this problem, we engineered two different deep learning models based on hidden representation learning. MAPLE, a matrix-based encoder which transforms packets into an image representation, uses convolutional neural networks to determine RTP packets from data flow. DATE is a density-analysis based tensor encoder which transforms packet data into a three-dimensional point cloud representation. We then perform density-based clustering over the point clouds as latent representations of the data, and classify packets as RTP or non-RTP based on their statistical clustering features. In this research, we show that these tools may allow a data collection and analysis pipeline to begin detecting and buffering RTP streams for later session association, solving the initial drop problem. MAPLE achieves over ninety-nine percent accuracy in RTP/non-RTP detection. The results of our experiments show that both models can not only classify RTP versus non-RTP packet streams, but could extend to other network traffic classification problems in real deployments of network analysis pipelines.

Introduction

Internet telephony, or Voice-over-IP (VoIP), allows for every form of video and audio communication from one-on-one calls to live streams on the web. It began as an alternative to landline-based telephone and has continued to replace these devices in individuals' homes, enterprises, and government facilities. In 2021, it was projected that 3 billion people use VoIP technology as a regular part of their daily lives. In the era of the COVID-19 pandemic, online/hybrid learning systems and the work from home movement encouraged the use and market of online streaming platforms which use underlying VoIP technology to transport network traffic data. In 2021-2022, Zoom reported over 300 million users (TeamStage 2022) Webex reported over 600 million, and Microsoft Teams recorded 275 million participants (Curry 2022).

The widespread adaptation of VoIP technology has highlighted several cybersecurity vulnerabilities. It is simple to spoof IP addresses or URIs with VoIP technology and produce robocalls used in phishing/spam or in denial of service attacks (Edwards, Gonzales, and Sullivan 2020). Malware can also be embedded into VoIP packet data, and is thus vulnerable to being the transport for backdoors, worms, trojans, and viruses (Wu et al. 2021; Nagaraja and Shah 2019). Cybersecurity specialists designing intrusion detection systems need to be able to intercept and process VoIP streams in order to detect these kinds of intrusions (Choti et al. 2021). In the forensic environment, VoIP call analysis and reconstruction can provide critical evidence of criminal activity. Policy or content rights violations or copyright infringement may also be detected through the analysis of the reconstructed calls (Kmet, Matousek, and Martin 2014; Sha, Manesh, and El-atty 2016). This process and search capability is also useful to intelligence operations to find mission critical data from the enormous amount of traffic flowing in mission networks (Kao, Wang, and Tsai 2020).

A majority of mainstream VoIP protocols use the real-time transport protocol (RTP) as their transport layer for encapsulation. The application layer protocol, for example the Session Initiation Protocol (SIP), Media Gateway Control Protocol (MGCP), or H.248, will establish the connection between endpoints and carry important session information such as codec encoding and metadata for the call. The application then relies on RTP to manage the dataflow between the connected systems. In the *complex, real network environment*, the implementation of SIP and RTP data flow is often sent de-coupled across physical signals and ports which presents a challenge for cybersecurity specialists using middlebox technologies for packet inspection and call reconstruction. Because the RTP data is encoded, it is necessary to know the signaling information in order to retrieve the codec information for proper decoding. Furthermore, the metadata associated with signaling protocols such as the SIP URI identifier provides necessary enrichment and context for the actual call. Lastly, RTP data may arrive at the middlebox before the signaling information which contains the port numbers associated with the RTP stream for that partic-

ular call; thus, the middlebox must retrospectively identify data packets which belong to particular signaling information headers.

Deep packet inspection, or DPI, is the process of inspecting packet payload contents in order to derive metadata or classify the packet. Industry is currently dominated by filtering and signature-matching solutions for deep packet inspection. Detecting the RTP stream itself this way without port numbers from other types of traffic can be a difficult problem because the RTP signature is weak. Furthermore, the rise of encryption (SRTP) makes text-based signatures ineffective. The problem of accurately classifying network traffic has expanded beyond the scope of capability of text-based solutions. Instead, we propose using higher dimensionality of network traffic in order to advance RTP detection capability.

We propose two methods of embedding higher dimensionality features of network traffic into processable input for deep learning models. The deployable system MAPLE, a **ma**trix-based **p**ay**l**oad **e**ncoder, transforms network packets into two-dimensional grayscale images. Doing this, the system is capable of encoding hidden latent representations from the structure of the byte payload. The MAPLE model uses these embeddings as input to a two-dimensional convolutional neural network (CNN) which then classifies the packet as RTP or non-RTP. We also propose DATE, a cluster **d**ensity **a**nalysis-based **t**ensor **e**ncoder model which expands network packets into three-dimensional point clouds, which we then extract features of regarding cluster density in order to find commonalities and classify based on data shape. Using MAPLE and DATE, specialists are capable of detecting RTP streams without signaling information from other types of data in the real field.

MAPLE provides the following contributions to the state-of-the-art:

- An algorithm for encoding packet payloads to image-based embeddings for latent representation,
- an empirical evaluation and comparison of CNN models on RTP data,

DATE provides the following additional contributions:

- an algorithm for generating three-dimensional point clouds from packet data, creating spatial latent representations as a novel method of packet analysis,
- a novel application of density-based cluster analysis on packet payloads,

Both tools as part of a DPI framework contribute the following:

- deployable classification models for identifying RTP streams from unknown data for real system application,
- a descriptive end-to-end pipeline for packet analysis using both header features as well as density-based payload analysis.

Convolutional Neural Networks

Convolutional neural networks are designed to process pixelated data and discover visual patterns in the latent space. CNNs organize their data into three dimensions, the spatial dimensionality of the input (height and width) and the depth.

The CNN model is comprised of three types of layers following the input layer. A *convolutional layer* uses a kernel filter in order to compute feature maps. The feature value of location (i, j) in the kth feature map of the ith layer $z^l_{i,j,k'}$ is calculated by:

$$z^l_{i,j,k} = {w^l_k}^T x^l_{i,j} + b^l_k \tag{1}$$

where w^l_k and b^l_k are the weight vector and bias term of the kth filter at the lth layer. The activation function, usually sigmoid, tanh, or rectified lineur unit (ReLU), may then be calculated as:

$$a^l_{i,j,k} = a(z^l_{i,j,k}) \tag{2}$$

A *pooling* layer performs down-sampling along the spatial dimensionality of the given input in order to reduce dimensions. For each feature map $a^l_{i,j,k}$, pooling can be calculated as follows:

$$y^l_{i,j,k} = pool(a^l_{m,n,k}), \forall (m, n) \in R_{ij} \tag{3}$$

where R_{ij} represents the local neighborhood around point (i, j). Common pooling functions include max pooling and average pooling (Gu et al. 2018). Finally, *fully-connected* layer takes input from the previous layer and connects them to neurons of the output layer in order to perform high-level reasoning or learn some global semantic information (O'Shea and Nash 2015).

LeNet

LeNet (LeCun et al. 1998) is a traditional CNN architecture for image recognition. It consists of two sets of convolutional, activation, and pooling layers. There is a final set of fully-connected, activation, and fully-connected layers with a softmax normalization and classification at the output layer.

ResNet

Deep neural networks suffer from accuracy degradation due to how difficult it can become to map intermediary or residual layer outputs to inputs. A residual convolutional neural network (ResNet) is built up of blocks of stacked layers formed from a series of convolutions and non-linear activation functions. Following the authors who first introduced deep residual networks (He et al. 2015), we define a block as:

$$y = F(x, \{W_i\}) + x \tag{4}$$

Where x and y are the input and output vectors, respectively, and $F(x, \{W_i\})$ is the residual mapping with a set of weights W_i. If $H(x)$ represents the ideal output mapping that corresponds to the ground truth, residual networks hypothesize that $F(x) + x = H(x)$. The residual function is pushed to zero such that the equation becomes an identity function $H(x) = x$. This identity mapping, or shortcut layer, is what reduces the degradation problem and minimizes training error as the network becomes deeper (He et al. 2015).

Point Clouds

Point clouds are a set of Cartesian coordinates (x, y, z) which represent some points in a three-dimensional space. In computer vision, point clouds may be used as a method for mapping high-dimensional shapes into lower-dimensioned space. We emulate the work of Quach et al (Quach, Valenzise, and Dufaux 2020) and treat the point clouds as static representations analagous to manifolds in three dimensional space. This captures a geometrical representation of packet data which we use for cluster analysis.

Density-based Clustering

Density-Based Spatial Clustering of Applications with Noise (DBSCAN) uses three different point classifications (see Figure 1): *Core points* represent centers of clusters and points are labeled as such based on the number of neighbors they have compared to their neighbors. *Border points* represent the edges and possess the least number of neighbors but are still within the range of a core point. Noise or *Outlier points* are neither borders nor cores and do not belong to a cluster. The DBSCAN algorithm visits each point and classifies them as such in order to paint a picture of where clusters of data are (Schubert et al. 2017).

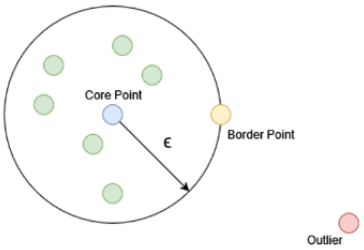

Figure 1: Illustration of types of points in DBSCAN.

Related Work

Costeux et al (Costeux, Guyard, and Bustos 2006) provide early work on the fast detection of Skype and other RTP-based telephony traffic. They focus their work on detecting applications, but specify several fields from the RTP and encapsulation header which can be used to filter out non-RTP traffic. We employ this technique in the end-to-end scenario discussed in this paper.

Kmet et al (Kmet, Matousek, and Martin 2014) build on the previous work by incorporating additional header features for per-packet selection and add a flow-based solution to reduce the over-selection problem. They were able to minimize mis-classification to nearly zero by buffering up to 10 packets of the RTP stream. The synchronization source number from the header is used along with timestamps to check for proper increment and stream correlation. In this set of experiments we focus on per-packet identification only.

CNNs have been used to classify network traffic in multiple problems for both per-packet and per-flow scenarios. Lim et al (Lim et al. 2019) also transform packets into grayscale images and label them by application (RDP, Skype, BitTorrent, and others). They use both a CNN and ResNet architecture. When using payloads of 1024B and a ResNet architecture, they achieved a 0.97 F1 score for accurately classifying the packets as one of eight types of applications.

Deep Packet (Lotfollahi et al. 2020) is a well-cited work in deep learning for traffic classification. They use a stacked auto-encoder and CNN to perform traffic classification into traffic types and application types as labeled in the VPN/non-VPN UNB ISCX dataset (Gil et al. 2016). Their solution achieves 94% accuracy in the traffic classification task per traffic type, and 97% accuracy in the application classification task. Incorporating the same dataset as well as additional, more recent flows, we are able to achieve higher accuracy in RTP detection with a less computationally complex framework in MAPLE.

DATE is a novel contribution to the field of packet processing as there is little published, previous work in generating 3D point cloud representations of packets. Raw packet data from LiDAR (light detection and radiation) systems have been shaped losslessly into 2D matrices and point clouds (Tu et al. 2019). This work introduces the problem of spatial correlation in packet data, namely that packets in their raw state are not usually uniform or in a state which may be processed using the spatial or geometric strategies employed by image and computer vision algorithms. Thus, data compression or pre-processing must be performed to create the necessary uniformity.

Some early research (Patwari, Hero III, and Pacholski 2005) describes expanding NetFlow data generated by routers into 2D manifolds. We reference this as an intial introduction of spatial expansion of traffic data in order to perform dimensionality reduction, which is a key theoretical linkage to our work.

MAPLE Methodology

It is expected that even in highly entropic or variable packet data, there will be some level of standardization (i.e. method names or status codes) or some commonalities such as user names, device details, semantic contexts, and other conversation or implementation-specific indicators. In order to delve deeper into these hidden layers and capture more latent space representations in packets, we propose in MAPLE to transform packets into grayscale images, hypothesizing that RTP packets will appear similar to one another. We base this assumption on previous work (Lim et al. 2019) which has shown similarities in grayscale images rendered from packet data in order to classify traffic by application and protocol type which are detectable by CNNs. Figure 2 asserts the validity of this assumption by showing comparative images rendered from packets in our combined dataset.

To shape RTP packet payloads into an image, we extract the payloads and convert them from byte encoding to a normalized decimal integer $i \in [0, 255]$ (Jo et al. 2020). For image size, we chose a 28×28 representation, for a total of 784 input features. In cases where the payload is shorter than 784 bytes, we apply padding for input normalization, or otherwise truncate the payload data.

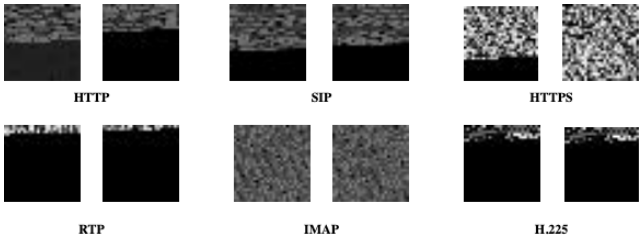

Figure 2: Grayscale images generated from packets in our combined dataset used in the following experiments.

Detection Model

Model 1: LeNet

We selected LeNet as a standard model in order to test the efficacy of convolutional neural networks to the RTP detection problem. LeNet has a low complexity, so has higher potential for practical use in real-time, line-rate packet inspection systems than deeper models which require more time. Figure 3 shows the set up of the model which we use as a baseline. We designed two separate models where one employed 6 and 16 kernels per convolutional layer (A) and another that used a 16 and 32 kernels (B). Each model contained two fully-connected, dense layers of size 1024 with a binary softmax classifier at the output layer.

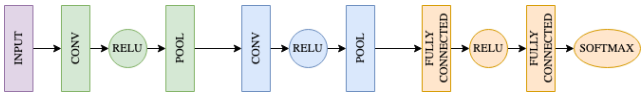

Figure 3: The architecture of our LeNet models.

Model 2: ResNet

For deployment purposes in real network environments, it is imperative that artificial intelligence solutions are capable of scaling to line-rate while still being able to perform the classification task to the required level of accuracy. While the definition of line-rate varies per network environment, the identity mappings of ResNet do not introduce additional complexity (He et al. 2015). Thus, we introduce residual mapping as a potential solution to adding additional layers of representation and thus improve classification through more hidden features, while also minimizing overhead.

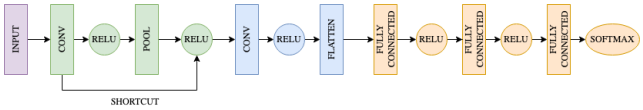

Figure 4: The architecture of our ResNet models.

We developed four ResNet-based models for the MAPLE system's experiments. The first model (C) consists of three convolutional layers with 32, 32, and 64 kernels of dimension 3×3, and three dense layers with output sizes of 64, 32, and 2 respectively. The second model (D) follows a similar architecture except the values are halved. Convolutional layers had 16, 16, and 32 kernels per layer and the three dense layers had output sizes of 32, 16, and 2. Model (E) corresponds to the same number of kernels and output sizes as (C), except it uses a kernel dimension of 7×7. For all models, the adam optimizer was used as well as ReLU activation between all layers except the final, which used softmax for normalization and classification. Categorical cross-entropy was used as the loss function for training. We also employed a dropout layer in order to reduce model overfitting.

Type	Model	# Kernels	Kernel Size	FC Dim
LeNet	A	3×3	6, 16	1024, 2
	B	3×3	16, 32	1024, 2
ResNet	C	3×3	32, 32, 64	64, 32, 2
	D	3×3	16, 16, 32	32, 16, 2
	E	7×7	32, 32, 64	64, 32, 2

Table 1: Configurations of each model used in the experiments.

DATE Methodology

DATE describes the density-based clustering analysis which is a novel contribution of this work; we also present DATE in the context of a larger packet processing system for real AI applications.

Point Cloud Creation

Payload data is extracted from the RTP packets and truncated to 1024 bytes. In cases where the payload is shorter than 1024, we add padding to normalize the input size. We convert the byte values v to decimal values $v' \in [0, 255]$. A sliding window technique is then used to create three-dimensional points in our feature space. We map each byte to a value in the (x, y, z) coordinate. The coordinates are mapped to a three-dimensional space to form point clouds like Figure 5. We expect that packets with similar data will thus result in similar point clouds.

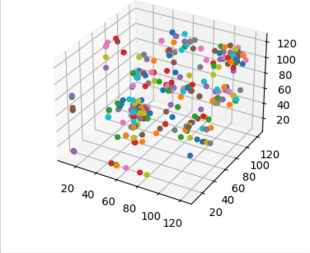

Figure 5: 3D point cloud made out of a Session Initiation Protocol (SIP) packet.

Clustering Analysis

After the point clouds are created, we process them using the DBSCAN algorithm in order to determine natural clusters

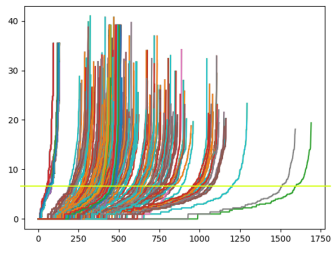

Figure 6: K-nearest neighbors graph generated for packets in the dataset using a `min_samples` = 6.

that exist in RTP packets versus non-RTP packets. The minimum number of samples must at least be the number of dimensions, and as a heuristic, it is generally twice the dimensions. All tests were performed with the DBSCAN configuration of ϵ = 7 and `min_samples` = 7. Having a baseline of 7 points to make a cluster within a 3D dimensional plane would be forgiving enough to create more clusters in a lower populated point cloud. An ϵ value of 7 should also allow for a greater number of clusters to be recognized (Rahmah and Sitanggang 2016).

Tuning DBSCAN has the potential to uncover great performance when it comes to creating recognizable clouds within the 3D space. As stated earlier, two inputs to DBSCAN can be tuned: ϵ and `min_samples`. It is important to have these values set reasonably to see favorable results. As a rule of thumb, the minimum number of samples needed to classify a cluster should be set to at least the number of dimensions. If it is set below this threshold, then singular points or two points that are close together would constitute a cluster, which would not be an accurate way of determining where data is clumped (Mullin 2020). Similarly, ϵ needs to be set based on general rules. If the value of ϵ chosen is too small then more clusters will be created, and more data points will be taken as noise. However, if set too large, then a number of smaller clusters will be more likely to merge into one large cluster. One way that a value of ϵ can be estimated is by examining the input data and finding the average distance between each point and its k nearest neighbors. Because each packet is highly variable, choosing an exact value for ϵ is a difficult problem. An estimation of ϵ must be taken from a conglomeration of the packet's k-nearest neighbors data where k is the value of `min_samples` chosen. As shown in the figure above, the point of each line with the greatest curvature is considered the ideal value for ϵ. A line has been illustrated in Figure 6 where an estimated average is made for every sample. The variation in an ideal ϵ value could be a major factor in why some packets are recognizable in their cloud state and others are not (Rahmah and Sitanggang 2016).

We extract the following features as the feature vector for classification from the DBSCAN results (i) clusterCount = number of clusters; (ii) averageClusterSize = average cluster size; (iii) standardDeviation = standard deviation; (iv) noisePercent = percent of cloud containing noise; (v) totalSize = number of points in the cloud.

Packet Classification

For deep learning, we feed the extracted cluster features forward into a two-layer multilayer perceptron unit (MLP) with a final softmax layer for classification. We use binary cross entropy as the loss function.

Experiments and Results

We ran several experiments to test each model configuration for binary RTP/non-RTP detection as well as multiclass protocol identification. Instead of using a single dataset or network environment, we used several available public datasets. Our sources include the CDX 2009 captures (Sangster et al. 2009), the Skynet Tor dataset (Guarnieri 2012), the Canadian Institute for Cybersecurity's ISCX VPN/non-VPN and Tor/non-Tor datasets (Gil et al. 2016), 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 (Silva et al. 2022). 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. Real network environments, particularly at scale, are more diverse. To increase scope, we merged PCAPs from these experiments and created our own repository of PCAPs containing 26 different protocols (see Table 2 for breakdown). This dataset is publicly available [1].

In the binary confusion matrix, true positive (TP) indicates correct classification of data as RTP. True negative (TN) is correct classification of data as non-RTP. false negative (FN) implies incorrect identification of traffic as non-RTP, and false positive (FP) is the incorrect classification of data as RTP. We use traditional measurements of precision, recall, and F1-score for model evaluation.

Experiment 1: RTP Detection

For the binary RTP/non-RTP classification for both models, we re-labeled all non-RTP and non-SRTP traffic as *non-RTP*, and merged SRTP and RTP traffic into an *RTP* label. For testing the different configurations of the MAPLE model, traffic was first processed and transformed into a matrix image, and then used as input for each of the models. We performed random under-sampling to avoid bias in the dataset, and then split the data into 60%/40% for training and testing. We ran tests with training for 3 epochs for each CNN model. As we plan to deploy this technology in a real RTP detection and deep packet inspection system, we also measured classification throughput in order to determine how much traffic the MAPLE system would be able to process given the detection model configuration. Results are provided in Table 3 and 4.

Generally, the ResNet model performed exceedingly well at the RTP detection task, exceeding ninety-nine percent accuracy across all the configurations. Decreasing the number of kernels may have slightly impacted accuracy, but improved overall throughput significantly and lessens memory

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

Protocol	Number of Packets
Bittorrent	20648
DHCP	1444
DNS	10563
FTP	10015
FTP_DATA	4000
GPRS	9981
GQUIC	1740
H.225	1300
HTTP	21298
IMAP	3318
LDAP	1354
MGCP	1568
NBNS	1216
NTP	1940
POP3	1675
PPTP	1288
RTCP	1626
RTP	15552
SIP	1112
SMB	3554
SMTP	5981
SSDP	8504
SSH	3039
Telnet	1888
TLS	4000
XMPP	1553

Table 2: Breakdown of dataset per protocol.

Model	P	R	F1
A			
Non-RTP	0.96	0.99	0.98
RTP	0.99	0.96	0.98
B			
Non-RTP	0.97	0.99	0.98
RTP	0.99	0.97	0.98
C			
Non-RTP	1.00	0.99	1.00
RTP	0.99	1.00	1.00
D			
Non-RTP	1.00	1.00	1.00
RTP	1.00	1.00	1.00
E			
Non-RTP	1.00	1.00	1.00
RTP	1.00	1.00	1.00

Table 3: Classification results of RTP vs non-RTP detection for all MAPLE models.

footprint. The LeNet models performed marginally worse in terms of accuracy, but the smaller of the two models had the highest throughput of any of the tested configurations. In a real network environment, it may be worth consideration to sacrifice some accuracy in order to be able to monitor more traffic or more effectively load-balance the received input depending on hardware capability. Thus, the ideal model configuration is often environment-dependent and inspired us to incorporate configuration capability in MAPLE so that the deployed system may best suit the environment. Another configuration choice which may have significant impact on

Model	Accuracy	Mb/s
A	0.975348	12.5
B	0.978258	9.28
C	0.996934	7.07
D	0.995908	10.41
E	0.996847	5.3

Table 4: Throughput results of RTP vs non-RTP detection for all MAPLE models.

model accuracy is the number of epochs trained. Real systems place an emphasis on minimizing down time of a system, but in actual deployment many models may be trained offline and then embedded. In systems where training time is not a factor of performance, we can thus increase the number of epochs with little real impact. In order to test the efficacy of such a design choice, we re-ran MAPLE models A and C with 10 epochs to see if performance improved. Results in Tables 5 and 6 show that ResNet appears to learn faster than LeNet, as it gained very little improvement with additional training time. Interestingly, with additional training time LeNet approaches similar accuracy to ResNet but still maintains higher throughput.

Model	P	R	F1
A			
Non-RTP	0.99	0.99	0.99
RTP	0.99	0.99	0.99
C			
Non-RTP	1.00	1.00	1.00
RTP	1.00	1.00	1.00

Table 5: Classification results of RTP vs non-RTP detection for MAPLE models A and C with additional training time.

Model	Accuracy	Mb/s
A	0.990499	14.99
C	0.998172	7.38

Table 6: Throughput results of RTP vs non-RTP detection for MAPLE models A and C with additional training time.

Class	P	R	F1
1 Epoch			
Non-RTP	0.85	0.84	0.84
RTP	0.83	0.85	0.84
10 Epochs			
Non-RTP	0.94	0.81	0.87
RTP	0.83	0.95	0.89
20 Epochs			
Non-RTP	0.96	0.79	0.87
RTP	0.82	0.96	0.89

Table 7: Classification results of RTP vs non-RTP detection for the DATE model.

We use the same data and experimental setup in order to test the performance of DATE. First, packets are processed

into three-dimensional point clouds, and run through DBSCAN in order to glean statistical features. The features are used as input into the multi-layer perceptron for classification. The results are provided in Table 7. In order to benchmark the system for real deployment, we recorded the time classification took. While this is heavily system dependent and showed some variance across parameter tuning or model configuration, the average classification time for a single packet by DATE was 0.15823 seconds. All tests were performed on a single CPU of a 1.6 GHz dual-core Intel i5 processor with 16 GB DDR3 RAM. In future work, we propose implementing multi-threading and multi-core processing as potential optimizations. We observed that point cloud generation was a pain point for the system in terms of cycles, and propose offloading such repetitive calculations to a specialized hardware such as FPGA (Song and Lockwood 2005) when available in the deployed system.

Experiment 2: H.225 Detection

A similar detection problem to the RTP situation exists for H.225 signaling protocol, which handles the registration and call setup for certain VoIP architectures using the H.323 protocol suite. Like RTP, the H.225 data may arrive to an endpoint first before the H.323 data, and also suffers from a weak signature. Thus, we additionally tested DATE's ability to detect H.225 traffic from non-H.225 traffic. Our setup of the dataset labels and the DBSCAN configuration was similar to the RTP test (results in Table 8).

Class	P	R	F1
1 Epoch			
Non-H.225	0.98	0.83	0.90
H.225	0.85	0.99	0.92
10 Epochs			
Non-H.225	0.91	0.87	0.89
H.225	0.88	0.92	0.90
20 Epochs			
Non-H.225	0.98	0.83	0.90
H.225	0.85	0.99	0.92

Table 8: Classification results of H.225 vs non-H.225 detection for the DATE model.

Experiment 3: Header-based Filtering

Middleware boxes which perform packet capture on signals are commercially available. Processing the entire traffic stream with deep learning techniques will overwhelm NICs in a real system and cause buffer overflow as data is waiting to be processed. For this reason, we add an initial culling filter on the traffic which may be optimally performed in data plane processing. The following checks are performed:

- Ethertype: 0x800 or 0x86dd (IPv4/IPv6)
- IP Next Protocol Field: 17 (UDP)
- UDP Header Length: $\geq$12
- UDP Source/Dest Ports: $>$1023
- RTP Version Field: 2
- RTP Payload Type: $t \in \{$0-34, 96-127, 192, 193, 200-204$\}$
- RTP Layer Length: $12 + 4{\cdot}CC_{RTP} + P_{RTP} \cdot PLen_{RTP} + X_{RTP} \cdot XLen_{RTP} \cdot X \geq Len_{UDP} - 8$

Prefiltering is effective for load-balancing for systems at scale (Baboescu and Varghese 2001). Another filter we implement is locality-sensitive hashing in order to approximate the Jaccard similarity of encapsulation header features among packets. From the IP layer of packets, we extract the *type of service* field, *total length* field, *IP flags*, and *next protocol* field. From the transport layer, we add the *source port* and *destination port*. We treat all these values as tokens in the set S and approximate the Jaccard similarity of packets by calculating their MinHash (Broder 2000). Our implementation uses LSHForest (Bawa, Condie, and Ganesan 2005) for similarity search. Overall, Table 9 shows the system was 100% accurate in classifying RTP traffic, and over 99% successful in accurately classifying H.225 packets. DATE as part of a multi-embedding improved overall classification accuracy for H.225, as well.

Class	P	R	F1
DATE & LSH			
Non-H.225	1.00	0.99	1.00
H.225	0.99	1.00	1.00
Non-RTP	1.00	1.00	1.00
RTP	1.00	1.00	1.00
LSH			
Non-H.225	1.00	0.97	0.99
H.225	0.97	1.00	0.99
Non-RTP	1.00	1.00	1.00
RTP	1.00	1.00	1.00

Table 9: Classification results for the end-to-end pipeline for each of the configurable payload embedding models.

Comparison of Models

We propose both MAPLE and DATE as potential methods for generating hidden, latent-space representations of traffic as their capabilities extend to different problem areas. MAPLE works well on input data such as a payload which may be distinct one data sample from another, but can be divided and matricized into units for comparison (i.e. turned into a grayscale image of uniform dimension). On the other hand, DATE has the potential to expand to include other non-network features and represent more heterogenous data. For example, input sensor data from IoT devices or light detection and ranging (LiDAR) equipment have used DBSCAN for data processing and normalization (Wang et al. 2019); it could be combined with cloud/network data inputs as an additional embedding model for classification problems. While for the RTP problem MAPLE provides high accuracy with more efficiency than DATE, there is a trade-off within the embedding space as DATE may be able to represent data of higher complexity in other problems for future work.

End-to-End Deployment

As real deployable systems, MAPLE and DATE are suited well to fit into any middlebox technology or processing flow. In order to demonstrate its capability, we provide an example end-to-end deployment using available technologies along with the models for VoIP call reconstruction and analysis for forensic, intelligence, law enforcement, and cybersecurity applications.

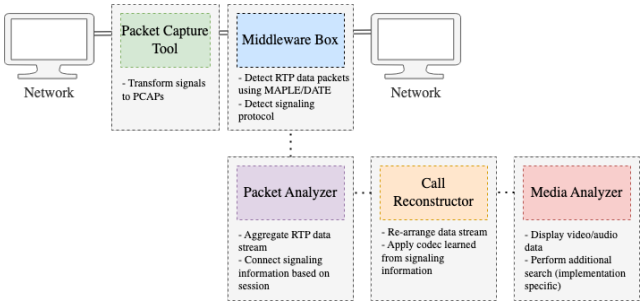

Figure 7: Our proposed data processing pipeline for analyzing VoIP calls.

Packet Capture Tool

In order for our middle box technology to process the incoming VoIP streams, a packet capture tool is necessary to process signals. A detailed capture system is outside the scope of this work, but commercial packet sniffers are available. As an open source example, Wireshark may be used to capture traffic and write it to a file storage system as packet captures (PCAP). In this scenario, we assume that there are more traffic types than VoIP in the network, for example web traffic like HTTP/HTTPS and DNS, or email data like POP3 and SMTP. We also assume that the analysts who are controlling our end-to-end system are interested in more data types than just VoIP, and are thus configured to capture more than just that.

Middleware Box

The middleware box is the core component which performs RTP detection. In the scenario, the middleware box may perform additional filtering and searching on header features as we previously mentioned in Experiment 3. Once traffic has been reasonably thinned, the data selected will be further classified as RTP or non-RTP by the MAPLE or DATE model. We assume in this scenario that the model has been pre-trained on RTP data. The system encodes packets into a matrix image representation as described in the methods section. Packet images are then processed through the CNN or density-based analysis algorithm and classified as RTP or non-RTP.

Packet Analyzer

Once data is captured, it must be directed from multiple signals to the same packet analysis software. This is done so that signaling information can be matched to the RTP packets. In the scenario, the packet analyzer recognizes the SIP handshake process and establishes the clients and their communication channels. From the signaling information, the corresponding dynamic ports being used for RTP transport can be parsed and paired with incoming RTP traffic for correlation (Sha, Manesh, and El-atty 2016).

Call Reconstructor

Once the session information is connected with the RTP flow, the codec information found in the RTP payload type field can be associated with the call for further decoding. This also requires the re-ordering of RTP packets as they may be sent out-of-order over UDP. RTP packets have a 16-bit sequence number which may be used for re-ordering. Sha et al (Sha, Manesh, and El-atty 2016) describe the call reconstruction process in further detail.

Media Analyzer

Once the call is fully reconstructed, the codec may be applied for video and/or audio playback using any media player which supports the format. Further software analysis of the VoIP data will be implementation specific. Example applications would include a signature-matching component to detect embedded malware (Rehman, Hazarika, and Chetia 2011). A software could be implemented to search VoIP calls for content of interest for policy checking, intelligence operations, or forensic analysis (Kao, Wang, and Tsai 2020). Call quality could also be monitored for network administration, or statistics on call data flow recorded for load balancing and network health monitoring (Assem et al. 2013).

Conclusion

We present MAPLE and DATE, deployable representation learning-based solutions to RTP detection in a middlebox software pipeline. For deep packet inspection, these systems are able to create latent representations of payload data which uniquely identify traffic of different types at a higher dimension than is considered by current signature-matching or filter-based solutions used in industry. Our implementations for this initial deployment solve the RTP detection problem with high accuracy using a minimal framework ideal for line-rate. We plan to expand this emerging technology to deployed VoIP processing solutions as an applied AI in the real network environment.

References

Assem, H.; Malone, D.; Dunne, J.; and O'Sullivan, P. 2013. Monitoring VoIP call quality using improved simplified E-model. 927–931. ISBN 978-1-4673-5287-1.

Baboescu, F.; and Varghese, G. 2001. Scalable packet classification. *ACM SIGCOMM Computer Communication Review*, 31(4): 199–210.

Bawa, M.; Condie, T.; and Ganesan, P. 2005. Lsh forest: self-tuning indexes for similarity search. In *In WWW*.

Broder, A. Z. 2000. Identifying and Filtering Near-Duplicate Documents. In Giancarlo, R.; and Sankoff, D., eds., *Combinatorial Pattern Matching*, 1–10. Berlin, Heidelberg: Springer Berlin Heidelberg. ISBN 978-3-540-45123-5.

Choti, C.; Hnoohom, N.; Tritilanunt, S.; and Yuenyong, S. 2021. Prediction of Intrusion Detection in Voice over Internet Protocol System using Machine Learning. In *The 2021 9th International Conference on Computer and Communications Management*, 149–155.

Costeux, J.-L.; Guyard, F.; and Bustos, A.-M. 2006. QRP08-5: Detection and Comparison of RTP and Skype Traffic and Performance. In *IEEE Globecom 2006*, 1–5.

Curry, D. 2022. Microsoft Teams Revenue and Usage Statistics (2022). https://www.businessofapps.com/data/microsoft-teams-statistics/. Accessed: 2023-01-07.

Edwards, G. W.; Gonzales, M. J.; and Sullivan, M. A. 2020. Robocalling: STIRRED AND SHAKEN! - An Investigation of Calling Displays on Trust and Answer Rates. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, CHI '20, 1–12. New York, NY, USA: Association for Computing Machinery. ISBN 9781450367080.

Gil, G. D.; Lashkari, A. H.; Mamun, M.; and Ghorbani, A. A. 2016. VPN/non-VPN Dataset (ISCXVPN2016); Tor/non-Tor Dataset (ISCXTor2016).

Gu, J.; Wang, Z.; Kuen, J.; Ma, L.; Shahroudy, A.; Shuai, B.; Liu, T.; Wang, X.; Wang, G.; Cai, J.; et al. 2018. Recent advances in convolutional neural networks. *Pattern recognition*, 77: 354–377.

Guarnieri, C. 2012. Skynet: a tor-powered botnet straight from Reddit. https://contagiodump.blogspot.com/2012/12/dec-2012-skynet-tor-botnet-trojantbot.html. Accessed: 2023-01-07.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2015. Deep Residual Learning for Image Recognition. *CoRR*, abs/1512.03385.

Jo, W.; Kim, S.; Lee, C.; and Shon, T. 2020. Packet Preprocessing in CNN-based network intrusion detection system. *Electronics*, 9(7): 1151.

Kao, D.-Y.; Wang, T.-C.; and Tsai, F.-C. 2020. Forensic Artifacts of Network Traffic on WeChat Calls. In *2020 22nd International Conference on Advanced Communication Technology (ICACT)*, 262–267. IEEE.

Kmet, M.; Matousek, P.; and Martin, O. R. 2014. FAST RTP DETECTION AND CODECS CLASSIFICATION IN INTERNET TRAFFIC.

LeCun, Y.; Bottou, L.; Bengio, Y.; and Haffner, P. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11): 2278–2324.

Lim, H.-K.; Kim, J.-B.; Heo, J.-S.; Kim, K.; Hong, Y.-G.; and Han, Y.-H. 2019. Packet-based Network Traffic Classification Using Deep Learning. In *2019 International Conference on Artificial Intelligence in Information and Communication (ICAIIC)*, 046–051.

Lotfollahi, M.; Jafari Siavoshani, M.; Shirali Hossein Zade, R.; and Saberian, M. 2020. Deep packet: A novel approach for encrypted traffic classification using deep learning. *Soft Computing*, 24(3): 1999–2012.

Mullin, T. 2020. DBSCAN Parameter Estimation Using Python. https://medium.com/@tarammullin/dbscan-parameter-estimation-ff8330e3a3bd. Accessed: 2023-01-07.

Nagaraja, S.; and Shah, R. 2019. VoipLoc: VoIP call provenance using acoustic side-channels. *IEEE Security and Privacy 2020*.

O'Shea, K.; and Nash, R. 2015. An Introduction to Convolutional Neural Networks. *ArXiv*, abs/1511.08458.

Patwari, N.; Hero III, A. O.; and Pacholski, A. 2005. Manifold learning visualization of network traffic data. In *Proceedings of the 2005 ACM SIGCOMM workshop on Mining network data*, 191–196.

Quach, M.; Valenzise, G.; and Dufaux, F. 2020. Folding-Based Compression Of Point Cloud Attributes. In *2020 IEEE International Conference on Image Processing (ICIP)*, 3309–3313.

Rahmah, N.; and Sitanggang, I. S. 2016. Determination of optimal epsilon (eps) value on dbscan algorithm to clustering data on peatland hotspots in sumatra. In *IOP conference series: earth and environmental science*, volume 31, 012012. IOP Publishing.

Rehman, R.; Hazarika, G.; and Chetia, G. 2011. Malware threats and mitigation strategies: a survey. *Journal of Theoretical and Applied Information Technology*, 29(2): 69–73.

Sangster, B.; O'Connor, T. J.; Cook, T.; Fanelli, R.; Dean, E.; Adams, W. J.; Morrell, C.; and Conti, G. 2009. Toward Instrumenting Network Warfare Competitions to Generate Labeled Datasets. In *Proceedings of the 2nd Conference on Cyber Security Experimentation and Test*, CSET'09, 9. USA: USENIX Association.

Schubert, E.; Sander, J.; Ester, M.; Kriegel, H. P.; and Xu, X. 2017. DBSCAN revisited, revisited: why and how you should (still) use DBSCAN. *ACM Transactions on Database Systems (TODS)*, 42(3): 1–21.

Sha, M.; Manesh, T.; and El-atty, S. M. A. 2016. VoIP Forensic Analyzer. *International Journal of Advanced Computer Science and Applications*, 7.

Silva, J. V. V.; Oliveira, N. R.; Medeiros, D. S. V.; Lopez, M. A.; and Mattos, D. M. F. 2022. A statistical analysis of instrinsic bias of network security datasets for training machine learning mechanisms.

Song, H.; and Lockwood, J. W. 2005. Efficient packet classification for network intrusion detection using FPGA. In *FPGA '05*.

TeamStage. 2022. Zoom Statistics 2022: How Video Conferencing Changed Our Lives. https://teamstage.io/zoom-statistics/. Accessed: 2023-01-07.

Tu, C.; Takeuchi, E.; Carballo, A.; and Takeda, K. 2019. Point Cloud Compression for 3D LiDAR Sensor using Recurrent Neural Network with Residual Blocks. In *2019 International Conference on Robotics and Automation (ICRA)*, 3274–3280.

Wang, C.; Ji, M.; Wang, J.; Wen, W.; Li, T.; and Sun, Y. 2019. An Improved DBSCAN Method for LiDAR Data Segmentation with Automatic Eps Estimation. *Sensors*, 19(1).

Wu, Z. Z.; Guo, J.; Zhang, C.; and Li, C. 2021. Steganography and Steganalysis in Voice over IP: A Review. *Sensors (Basel, Switzerland)*, 21.