Live vs. Video on Demand within a VPN Detection

Andrey Pristinsky[1] Da Gong[2] Mariam Qader[3] Tianran Qiu[4] Zishun Jin[5]
[1]Halıcıoğlu Data Science Institute,
University of California, San Diego

I. INTRODUCTION

Due to the variety, affordability and convenience of online video streaming, there are more subscribers than ever to video streaming platforms. Moreover, the decreased operation of non-essential businesses and increase in the number of people working from home in this past year has further compounded this effect. More people are streaming live lectures, sports, news, and video calls via the internet at home today than we have ever seen before. In March 2020, Youtube saw a 2.5x increase in the amount of time people spent streaming live video [1]. Twitch more than doubled their hours of content in three months after the start of the pandemic [1]. There is a huge boom in the video content world, and it does not seem to be slowing down anytime soon. Internet Service Providers, such as Viasat, are tasked with optimizing internet connections and tailoring their allocation of resources to fit each unique customer's needs. With this increase in internet activity, it would be especially beneficial for Viasat to understand what issues arise when customers stream various forms of video. When a user has difficulties with their internet connections, ISP's want to be able to understand their activity to give potential reasons to why the problem occurred and a quick solution.

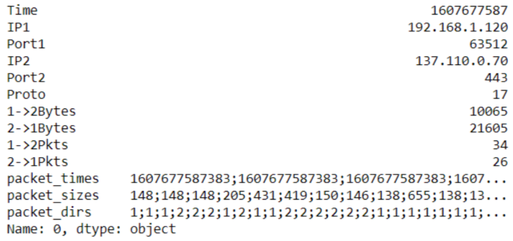
Although we are able to identify the genre of an activity when a user is not using a VPN, the challenge arises when a user chooses to surf the web through a VPN. When it comes to VPN use cases we can't identify a user's unique activity when they experience issues, thus making us unable to successfully troubleshoot those problems. Different forms of video streaming require different network resources for optimal experience, thus understanding how video is delivered could be especially handy. For example, if a customer watches a lot of live video they may prefer a connection with lower latency. A live stream requires low latency in order for the streamer and audience to communicate in real time without a significant lag. As latency increases, the delay in time between the audience receiving the video from the streamer (lag) increases. However, for VoD high latency is acceptable since an influx of packets can be stored or paused for longer and take a backburner on the network without a user noticing a change in video quality. This is where a tool that could identify various internet activities, specifically live or video on demand (VoD) streaming, within a VPN tunnel would be extremely useful for an Internet Service Provider.

Previously, there have been effective ways to distinguish video streaming from general internet activity, yet distinguishing between different types of video is a little more challenging.[1] User's experience both live video and VoD in the same way over the internet; users can play videos while the video platforms send the proceeding content making the video stream smooth with minor buffers. However, live video has a few components that VoD does not. Some live videos have an interactive component where the audience can communicate with the streamer in real time. Live streams also do not have quality controls, where users can set the quality of the video to a certain level. Our goal is to distinguish how providers send live video vs. pre-uploaded videos (VoD) to their users. Other works have successfully achieved classifying the two types of video by looking at the payloads of packets, however their methods do not work with encrypted VPN data since we are unable to see the raw data within packets [2].

Although we cannot see the contents of the packets transported across a network, patterns in the way they are sent can still help us identify if a user is streaming a live or pre-uploaded video. Our task is to detect key differences between the way information is sent across a network for live video streaming and VoD. To achieve this, we have generated an extensive data set consisting of network data for both types of video content. To create the data, we used a tool that connects to our own interfaces and captures consistent real time internet traffic from our personal devices. We have chosen to generate network data from platforms that offer both live and VoD content, such as Youtube and Twitch, as well as data from platforms such as Netflix, Facebook Live, Radio.com, Amazon Prime, Hulu, and Zoom (live video calls) and more. Through an extensive dataset drawing from multiple providers we were able to create a robust model that can identify when a user is streaming a VoD or a live video. This model is meant to be used in conjunction with another pipeline that can first verify that video streaming is occurring within a VPN tunnel. Using our findings, we can further classify what type of video a user is streaming, to help gain a better understanding of user activity to ultimately enhance user experience.

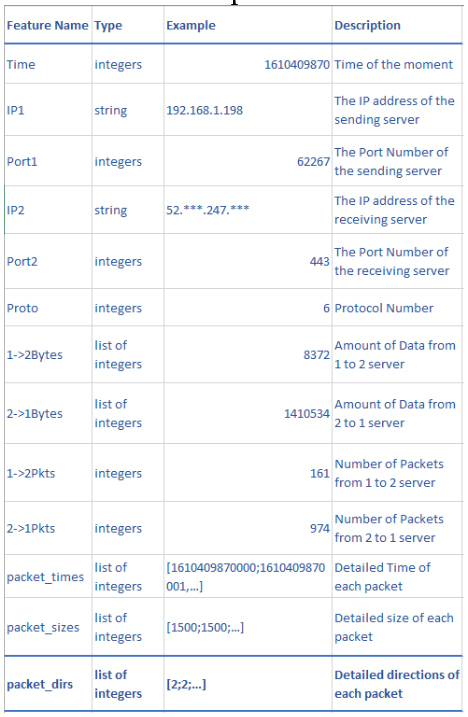
II. DATASET

The dataset of our project is generated using csv files from our google drive folder. Each csv file represents a five-minutes long recorded network traffic using network-stats tool provided by the Viasat.[2] Below is one of example of a row generated using this tool.



```
Time                                                    1607677587
IP1                                                  192.168.1.120
Port1                                                        63512
IP2                                                   137.110.0.70
Port2                                                          443
Proto                                                           17
1->2Bytes                                                    10065
2->1Bytes                                                    21605
1->2Pkts                                                        34
2->1Pkts                                                        26
packet_times     1607677587383;1607677587383;1607677587383;1607...
packet_sizes     148;148;148;205;431;419;150;146;138;655;138;13...
packet_dirs      1;1;1;2;2;2;1;2;1;1;2;2;2;2;2;2;1;1;1;1;1;1;1;...
Name: 0, dtype: object
```

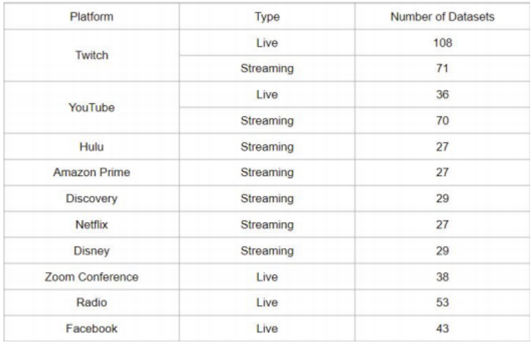
Below is the description of each column of a file.



Feature Name	Type	Example	Description
Time	integers	1610409870	Time of the moment
IP1	string	192.168.1.198	The IP address of the sending server
Port1	integers	62267	The Port Number of the sending server
IP2	string	52.***.247.***	The IP address of the receiving server
Port2	integers	443	The Port Number of the receiving server
Proto	integers	6	Protocol Number
1->2Bytes	list of integers	8372	Amount of Data from 1 to 2 server
2->1Bytes	list of integers	1410534	Amount of Data from 2 to 1 server
1->2Pkts	integers	161	Number of Packets from 1 to 2 server
2->1Pkts	integers	974	Number of Packets from 2 to 1 server
packet_times	list of integers	[1610409870000;1610409870001,...]	Detailed Time of each packet
packet_sizes	list of integers	[1500;1500;...]	Detailed size of each packet
packet_dirs	**list of integers**	**[2;2;...]**	**Detailed directions of each packet**

For the whole dataset, we have collected 556 five minute data chunks in total. The dataset is balanced with 278 VoD data and 278 Live Streaming data.

Below is the breakdown of sources for all the data (streaming data is videos on demand).

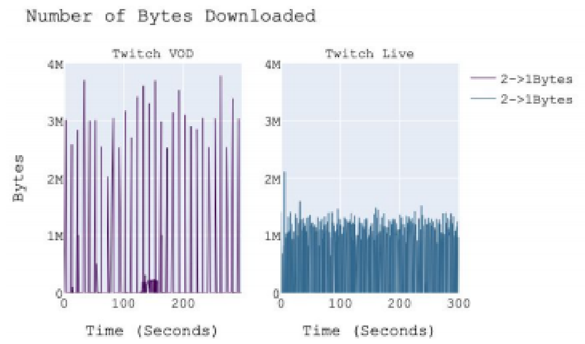


Platform	Type	Number of Datasets
Twitch	Live	108
	Streaming	71
YouTube	Live	36
	Streaming	70
Hulu	Streaming	27
Amazon Prime	Streaming	27
Discovery	Streaming	29
Netflix	Streaming	27
Disney	Streaming	29
Zoom Conference	Live	38
Radio	Live	53
Facebook	Live	43

III. METHODS

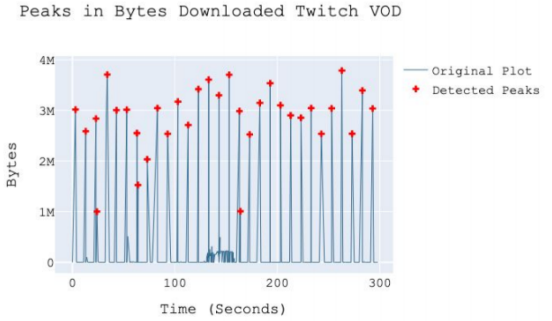
The internet data that we have collected consists of the number of packets and bytes uploaded and downloaded across a connection. A connection consists of the source and destination IP addresses and ports. Using this data, we can look at the flow of packets and bytes sent back and forth over time between the user and destination.With this information, and lots of exploratory data analysis we were able to find some key identifiers that can help us distinguish what type of video is being played.

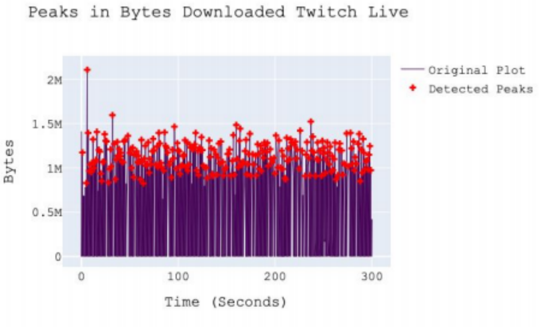
Similar to other common approaches to analyze internet network data, we have chosen to look for statistical differences between the flow of packets across a network for live video streaming and VoD. The graphs below look at the number of bytes downloaded across a network over time for five minute chunks of both Twitch live and Twitch uploaded videos.



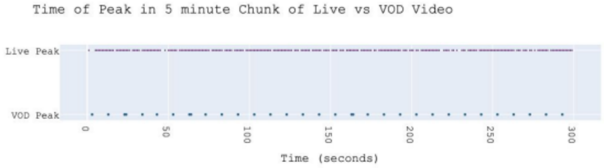
First, it is important to note that both graphs follow patterns we typically see for video streaming within a VPN, thus we can verify the data we are looking at is useful and correct. Video streaming data can be identified by patterns of consistent data being transferred across a network like we see in the graphs above. General internet traffic has a more sporadic and unpredictable looking plot. When looking at the graphs above a few differences are immediately apparent. First, we can see that the live video has a denser plot with bytes being downloaded in more frequently. On the other hand, the VoD has more time between each spike but the magnitude of packets coming in at a time is larger. To quantify this key difference, we can take the ratio of time packets that are being sent to the time packets that are not being sent (packet size is 0).[3] This will tell us how much time during the viewing of the video no packets were being sent from the destination to the user, which is larger for VoD compared to live video viewing. This is because since a VoD is pre-uploaded, providers can send larger chunks of video content to their users, whereas for live video streams providers send content to user's as they receive it in real time.

Visually, we can note that there are a tremendous amount of spikes in the live video data compared to the VoD. To quantify this difference, we can simply look at the count of spikes in bytes being downloaded for a five minute chunk of video. The graphs below highlight where each spike is, and we classify a 'spike' and peak in the data using the mean number of bytes downloaded as a lower bound. Using this lower bound can help us filter out any noise that can be present when collecting network data at lower magnitudes.[4]



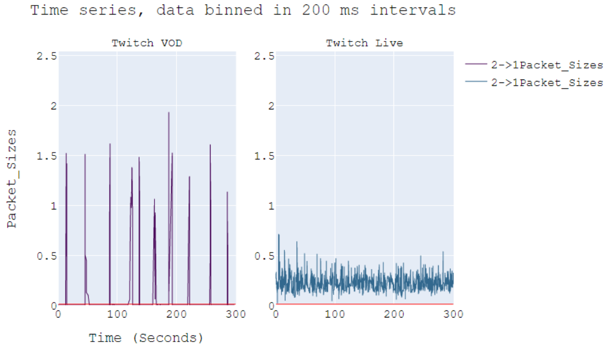


It is clear that Live video has a much larger number of spikes than the VOD. In this specific example live video has over five times as many peaks present. Another way to quantify this density in peaks is to look at how far apart spikes are from each other. Below we can see a graph that plots what time each spike occurs for VOD and Live Video in a span of 300 seconds.



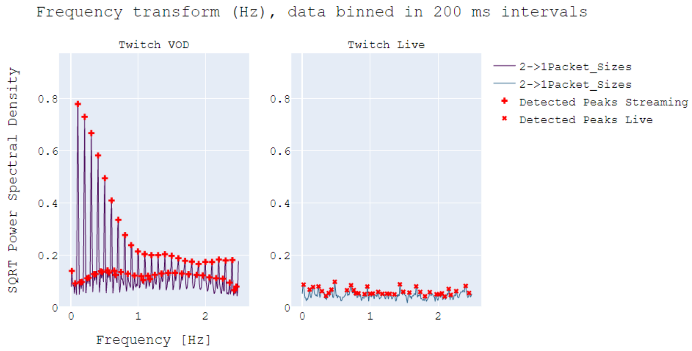
As we can see, the peaks for live video are much closer together than those for VoD. Each peak is about 10 seconds apart for VoD, whereas peaks for live videos are very frequent. It is important to note that there are many micro spikes that can't always be observed when looking at the main plot of bytes downloaded over a network (figure 1), which could taint the valid packet rate. For example, a noisy VoD data file may have many small packet transactions, resulting in the ratio of valid packets transferred to be as high as the live video streaming. Looking at the time intervals between peaks helps eliminate this possible error, since we are filtering out the smaller spikes.[5]

With live vs VOD being visually different when looking at the Byte Counts, we decided to further our analysis by exploring the extended columns within our datasets. These extended columns such as Packet Sizes were provided to us through our data collection method, that could potentially give more in depth information about the data we are working with. By binning Packet Sizes in 200 millisecond intervals, we were able to discover a new angle in which to analyze the datasets in the time domain. The graph below shows Twitch VoD vs Twitch Live, with Packet Sizes being significantly larger for VoD than for Live.



With further analysis, we could also see that the VoD data has larger spikes of Packet Sizes and large increments of nearly no Packet Sizes being transferred while the Live data always has Packet Sizes being transferred. The red line in the graphs above are at a 0.01 threshold, and from this we can infer that a possible feature is in the works just by looking at how many Packet Sizes are below that threshold.[6]

After binning in 200 millisecond intervals, we wanted to look at the data we had in the frequency domain to see if we can find any further information present for Packet Sizes. By using Welch's Method[9], we were able to transform the data to the frequency domain. The graphs below show the difference between VoD and Live, with peaks being significantly larger in VoD in comparison to the Live data.



The differences between these large peaks and troughs led us to believe that the height between them is a viable feature that we could use in our model.[6] As well, the frequency domain showed us that peaks are occurring at very specific intervals between both VoD and Live. Every increment of 0.1Hz and 0.2Hz, the peaks for VoD were following a consistent downward trend while Live peaks spiked and fell in a very consistent linear trend. These Hz values showed enough of a difference between VOD and Live that we decided to find the minimum 0.1Hz [8] and 0.2Hz [9] values present in each dataset and normalize them to create two new features, with streaming on average having larger values than live.

With all of our analysis, we were able to find key features that would benefit our model. Typically, VOD has more leisure time and live streaming has less. Live streaming requires video providers to consistently send data to their users as they are sending it in real time: this is a key difference in the way live streaming vs. VOD is delivered to viewers.

IV. MODEL

Using the features we have extracted, we then fed them through various supervised machine learning models to predict if live video or VoD streaming is occurring within a VPN tunnel. We trained each model with 80% of our data (446 datasets), and tested it with 20% (112 datasets). The data used for training/testing was decided randomly, to ensure a representative sample of platforms and classes (live vs VoD). The models we looked at are as follows:

A. SVM(Support-Vector Machine)

Support-vector machines is a supervised learning model with associated learning algorithms that analyze data for classification and regression analysis. It will construct a hyperplane or set of hyperplanes in a high- or infinite-dimensional space for further classification. For this specific project, we have passed scaled features[11] value into the SVM classifier, this model has an accuracy of around 76%.

B. Linear SVM(Support-Vector Machine)

Similar to SVM classifier above but use linear as kernel type instead. It means that lines will be used to classify each feature to two classes in each dimension. This model has an accuracy of 88%.

C. Logistic Regression

Logistic Regression is one of the most commonly used machine learning algorithms for classification. It is a reliable predictor for the two classes classification problem we have. After passing all our features into it, the model has an 89% accuracy.

D. KNeighbors

K-NN is a type of classification that relies on distance for classification. For classification, a useful technique is used to assign weights to the contributions of the neighbors, so that the nearer neighbors contribute more to the average than the more distant ones. For this project, we have used a K-NN classifier which takes 3 nearest neighbors for classification. This model has an accuracy of 96%.

E. Random Forest

Random forests is an ensemble learning method for classification that operates by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean/average prediction (regression) of the individual trees. Random decision forests correct for decision trees' habit of overfitting to their training set. Normally, it will outperform the normal decision tree model which is the main reason we used it for our project. Following is the importance of each feature returned by the random forest model. The accuracy of this model is 97%.

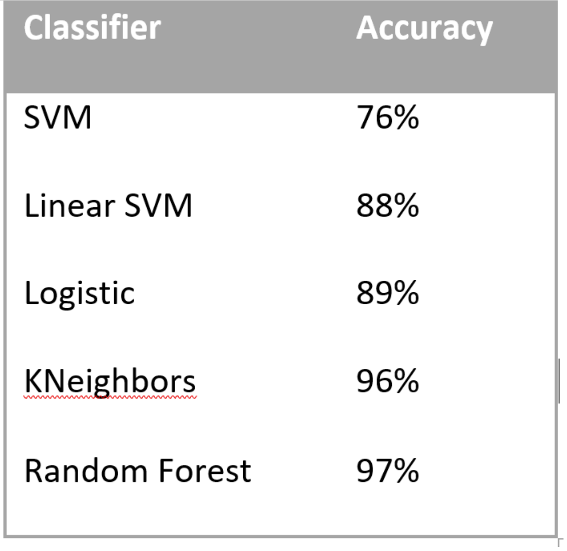
V. Results

A. *Trained Binary Classifier*

For the training process, we have randomly split 20 percent of our dataset for the validation set. For each of the classifiers, we used 80 percent of the data files to train our model and used the rest for model testing purposes.

B. Model Selection

Below is the accuracy of each classifier on the validation set.

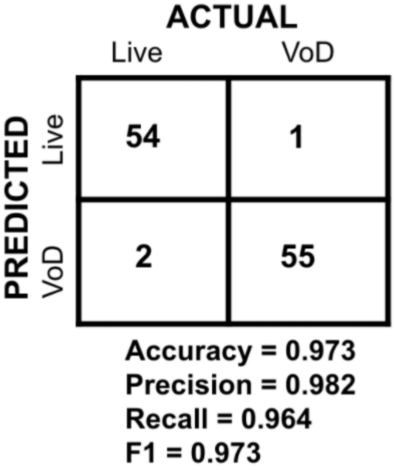


Classifier	Accuracy
SVM	76%
Linear SVM	88%
Logistic	89%
KNeighbors	96%
Random Forest	97%

We can observe from above that Random Forest Classifier has the best accuracy. By this, our model is trained using Random Forest Classifier.

VI. **Discussion**

The most powerful model we tried is Random Forest, and this is the model we chose to output for our project. With a high accuracy score of 97%, the model generalizes very well compared to others. The confusion matrix for the model can be seen below:



		ACTUAL	
		Live	VoD
PREDICTED	Live	54	1
	VoD	2	55

Accuracy = 0.973
Precision = 0.982
Recall = 0.964
F1 = 0.973

Out of over 100 test data files, only 3 files were inaccurately classified with the wrong class label. The model has high accuracy, precision, recall and F1 scores, proving it to be robust and correct. It is also important to note that although this is the best model, the other models were able to achieve high accuracy scores as well. This shows that the features we chose to extract are very telling of how live video and VoD is delivered to users. We can also see that there are very dramatic differences in the way these two types of video content are sent, helping us achieve creating a strong model that is robust and generalizable. With this information, ISP's can use our model in conjunction with others to get a better understanding of user streaming patterns and ultimately improve overall experience.

VII. **Further Work**

As the project is finished, we are thinking about what further steps we can take to perfect our project, which includes improving the project's comprehensiveness, effectiveness and ease of use. In the future, we will possibly cover more user scenarios to make the project more comprehensive, such as still image video versus action videos, high resolution video versus, low resolution video. Moreover, we can expand our project beyond video streaming, by exploring use cases like gaming, music streaming, video streaming etc. These added cases can hugely increase the usability and flexibility of our algorithm to reach out for a bigger market and to benefit users of different types.

As we think about these further steps, it is important to keep data ethics and privacy in mind. In the wake of data breaches and privacy concerns, VPNs are more popular today than ever before due to privacy concerns of users [4]. Although it is justifiable for ISP's to understand user traffic, by being able to identify game and music streams, to optimize connections and tailor services to their customer's needs, there is a limit. [5] The lengths we take to unravel the decryption procedure by VPN's could start to make user's uncomfortable, and must always be considered before we look at user data.

Regarding the effectiveness, we would want to cut down the size of the data chunks needed for our training and prediction. The smaller chunks of traffic data will simplify the data collection and training process while improving the user experience of the model. On one hand, users and us can spend less hours collecting data; on the other hand, our analysis can be more precise because smaller chunks of traffic data can represent more variations within the traffic. In addition, we will automate the pipeline for data collection, analysis, model selection and prediction to improve the efficiency. In the end, in order for general users to use our algorithm easily and for clearer presentation, we will embed our algorithm to a website or software so that it could be explicitly used by more people.

Traditional internet traffic investigations such as studying packet payloads are still very common amongst many companies, which is very ineffective. Due to the increasing amount of personal internet data and the popularization of the VPN, these traditional methods are hindered. Even if the methods work, the efficiency is worse compared to the ML algorithm and the user's privacy is less protected. Therefore, we support these companies to keep up with the trend to update their back end services and incorporate ML into their algorithms. This will eventually save them time and energy, hence bringing more efficiency to their services as well as higher user satisfaction.

In the end, the possible use case for our projects would be focused on companies like ISPs and VPN providers. Firstly, these companies have the ability to collect user data due to the nature of their services. Secondly, they have the pressing need to investigate user's data, at the same time respect users' privacy, in order to provide better internet services and ultimately enhance user experience.

References

[1] Binder, Matt. "The livestreaming boom isn't slowing down anytime soon." https://mashable.com/article/future-of-livestreaming/

[2] Boxcast Team. "This Is Why Your Live Stream Lags: Intro To Live Streaming Latency" https://www.boxcast.com/blog/live-stream-video-latency

[3] R. Nossenson and S. Polacheck. "On-Line Flows Classification of Video Streaming Applications." https://ieeexplore.ieee.org/document/7371733

[4] Schaub, Sebastian. "Ethics and VPN: The Industry Needs to Aim Higher." TechRadar. https://www.techradar.com/news/ethics-and-vpn-the-industry-needs-to-aim-higher.

[5] Regano, Leonardo, Martino Trevisan, Alessio Viticchie, and Ali Safari Khatouni. "Ethical Issues of ISPs in the Modern Web." https://www.researchgate.net/publication/315514560_Ethical_issues_of_ISPs_in_the_modern_web

[6] Polacheck, Shuval. (2013). "Online Classification of VoD and Live Video Streaming Applications." https://www.idc.ac.il/en/schools/cs/research/documents/online\%20classification\%20of\%20vod\%202013.pdf.

[7] Bagui, Sikha & Fang, Xingang & Kalaimannan, Ezhil & Bagui, Subhash & Sheehan, Joseph. (2017). Comparison of machine-learning algorithms for classification of VPN network traffic flow using time-related features. Journal of Cyber Security Technology. 1. 1-19. 10.1080/23742917.2017.1321891.

[8] Guo, Lulu & Wu, Qianqiong & Liu, Shengli & Duan, Ming & Li, Huijie & Sun, Jianwen. (2020). Deep learning-based real-time VPN encrypted traffic identification methods. Journal of Real-Time Image Processing.17.1-12.10.1007/s11554-019-00930-6.

VIII. Appendix

A. Related Works

1) We have found some studies similar to our project. The first one comes from Shuval Polacheck's "Online Classification of VoD and Live Video Streaming Applications"[7]. In his research, he found out that there are two ways to classify VoD and Live Streaming. The first way is to classify use downlink average packets size and the second way is to compare average time difference between similar packets since video streaming data usually has larger information offset and live streaming data usually has smaller information offset. Similar to his research, we also tried to classify these two data types using features both from time domain and frequency domain. Also, our model outperformed his model due to the fact that we have extracted more features into the training process. The second research similar to us is the "Comparison of machine-learning algorithms for classification of VPN network traffic flow using time-related features"[8]. In this paper, the authors state that useful time-related features are in the picture below.

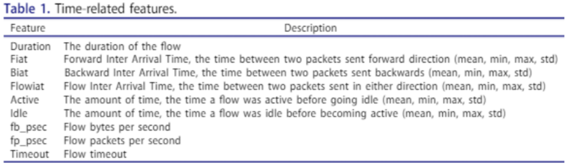


Table 1. Time-related features.

Feature	Description
Duration	The duration of the flow
Fiat	Forward Inter Arrival Time, the time between two packets sent forward direction (mean, min, max, std)
Biat	Backward Inter Arrival Time, the time between two packets sent backwards (mean, min, max, std)
Flowiat	Flow Inter Arrival Time, the time between two packets sent in either direction (mean, min, max, std)
Active	The amount of time, the time a flow was active before going idle (mean, min, max, std)
Idle	The amount of time, the time a flow was idle before becoming active (mean, min, max, std)
fb_psec	Flow bytes per second
fp_psec	Flow packets per second
Timeout	Flow timeout

We have extracted some features in the time domain similar to their findings like the feature "Packet Zeros" which returns the number of empty packets which relates to the time a flow is going idle. For the third research paper, we have "Deep learning-based real-time VPN encrypted traffic identification methods"[9]. The authors of this paper suggest there are two deep-learning based models which can be used for research encrypted data. The first one is convolutional Auto-Encoding (CAE) and the second one is Convolutional Neural Network (CNN). However, our random forest classifier works perfectly with around 97 percent accuracy. By this, we haven't applied deep learning methods in our project for potential waste of resources.

B. Network Stats

2) See for documentation on how data was generated: https://github.com/Viasat/network-stats

C. Features

3) Valid Packet Rate - this feature finds the ratio of time that there is a valid packet being sent within the 5 minute chunk of video. This is created by grouping the data by the time column, to count the number of valid packets (packet size > 0) that are downloaded in the 2->1Bytes column in the five minute chunk of video. Then,this is calculated by dividing the number of valid packets with the total time.

4) Number of Peaks - This feature finds the number of peaks that are greater than the mean of the 2 ->1 Bytes columns. It is calculated by the find_peaks method from the scipy library.

5) Interval gaps - This feature looks at the total length of intervals between the peaks of the spikes in the 2 ->1 Bytes column. The peak of spikes is defined when the value is greater than the mean of the 2 ->1 Bytes column. And then, by subtracting the time difference between peaks, the length of intervals between peaks is calculated. Then, by summing the lengths of intervals, the total interval gaps are found.

6) Packet Zeros - This feature finds the percentage of packets that are zero (below the 0.01 threshold), with the % being the value returned per dataset. It is calculated by finding the total number of packets below the 0.01 threshold in the 2->1 direction multiplied by 100 then divided by the length of the binned 200 ms intervals.

7) Max Prominence / Mean - This feature looks at the dataset in the frequency domain, then finds the maximum height present between a peak and trough and normalizes it. This normalized comparison is done in the 2->1 direction, and is calculated by binning the 2->1Packet Sizes into 200 millisecond intervals, applying Welch's method to transform the data to the frequency domain, and then using a find peaks method to find the max prominence present in the dataset.

8) peak_0.1Hz_norm - This feature grabs the minimum .1Hz value found within the transformed data in the frequency domain and normalizes it. This spectral feature is calculated by binning the 2->1Packet Sizes into 200 millisecond intervals, and applies Welch's method to transform the data to the frequency domain before looking at all the values occurring every 0.1Hz to find the minimum.

9) peak_0.2Hz_norm - This feature grabs the minimum .2Hz value found within the transformed data in the frequency domain and normalizes it. This spectral feature is calculated by binning the 2->1Packet Sizes into 200 millisecond intervals, and applies Welch's method to transform the data to the frequency domain before looking at all the values occurring every 0.2Hz to find the minimum.

D. Documentation

10) Welch's Method - Welch's method computes an estimate of the power spectral density by dividing the data into overlapping segments, computing a modified periodogram for each segment and averaging the periodograms. This description is pulled from `https://docs.scipy.org/doc/scipy/reference/generate\ \d/scipy.signal.welch.html` and it explains how we use Welch's method to estimate the power spectral density which converts the data we have to the frequency domain.

E. Model

11) Scale function: uses $1/(nfeatures * X.var())$ as value of gamma