Classifying streaming provider inside VPN connection using traffic flow statistics and spectral features

Molly Rowland, Jerry Qian, Raimundo Castro, Chang Yuan, Arely Vasquez
Halıcıoğlu Data Science Institute
{mhrowlan,jeq004,rac045,chy238,arv020}@ucsd.edu

March 2021

ABSTRACT

Whether to access another country's Netflix library or for privacy, more people are using Virtual Private Networks (VPN) to stream videos than ever before. However, many of the different service providers offer different user experiences that can lead to differences in the network transmissions. In this paper we will discuss the methods in which we made a classifying model to determine what streaming service provider was being used over a VPN. The streaming providers that the model identifies are Amazon Prime, Youtube, Netflix, Youtube Live and Twitch. This is valuable in understanding the differences in the network patterns for the different streaming service providers. Across all providers, our Random Forest model achieves a 96.5% accuracy in provider classification.

1. INTRODUCTION

Internet Service Providers (ISPs) strive to understand network conditions in order to better user experience. Currently, with more people using virtual private networks (VPN), machine learning algorithms that provide insight into the complex activity between the user and the VPN can provide enormous value to ISPs. In past work with Viasat, we have created classifiers that were successfully able to identify streaming data from browsing data across a VPN. Thus, now we design an algorithm that is able to take raw network data recorded in a VPN and output a streaming provider.

This paper proposes an algorithm known as *Streaming Provider Identifying Classifier Inside a VPN* (SPICIVPN, pronounced "Spicy VPN") which can differentiate between 5 different streaming providers (i.e. Netflix, Youtube, Amazon Prime, Twitch, and Youtube Live) with an accuracy of 96.5%. The strength of SPICIVPN lies in the thirteen features that process the raw data collected from *Network Stats* [1]. Such features are fed to a Random Forest Classifier which on average takes 10 minutes to process the raw data and produce an output.

The first part of our paper consists of the explanation of each feature. Next, we describe our model and hyperparameters in detail. Finally, we present the results obtained from SPICIVPN.

1.1 DATASET

The dataset used in this paper was collected at the University of California, San Diego in partnership with a Viasat, a San Diego based Communications company, and includes packet statistics of traffic flows passing through a VPN. Our data was collected in clips of 5 minutes, and the dataset contains over 500 clips. Our classifier will focus on classifying data as being from Netflix, Amazon Prime, Youtube, Youtube Live, Twitch, or Other, which would be an unspecified streaming service. For the sake of this research, we have chosen to have our Other category composed of Disney+, Discovery+, and Hulu. We chose to use these providers as they are a strong market segmentation of streaming providers available. We also chose to include Video On Demand (VOD) and live data. Live data is when the video is not prerecorded and being sent out as it is recorded, such as Twitch and Youtube Live to distinguish between many different streaming scenarios. Video on demand is any video that has been prerecorded and is being accessed off of a server. The data is collected using network-stats [1], a python script written by Viasat employee Charles Laubach, which outputs internet traffic flow statistics on a per-connection, per-second basis.

For each unique connection pair observed in a given second, the tool produces an output of packet metadata such as the source and destination IPs, application ports, and communication protocol, as well as traffic flow statistics including the aggregated and individual count, size, direction, and arrival time of contained packets. In order to classify the different streaming service providers, additional

fine-grained labels about the streaming activity were collected, such as the streaming service provider, video resolution and playback speed.

1.2 METHODS

The goal of this project is to make a machine learning (ML) model that can classify what streaming service was being used over a VPN. After exploring different models, we decided to use a Random Forest Classifier utilizing engineered features. It is important to incorporate meaningful features to help improve the accuracy of our model and keep variance low.

In creating the classifier, we collected network traffic data to train and test the model. The data was collected from a variety of different sources such as Netflix, Amazon Prime, Youtube, Twitch, Youtube Live, Disney+, Discovery+, and Hulu. All the data was collected at 1 times speed under clean network conditions, meaning no other traffic was being recorded besides the streaming capture. Each file notes at least the streaming platform and user who collected the data.

In creating features, we began by conducting exploratory data analysis on the data from the different streaming platforms. We compared the different platforms by analyzing the differences in their packet data, spectral analysis, and a variety of other features which we deemed meaningful for identifying the different streaming platforms.

An initial approach when looking at unique patterns between the different streaming platforms was looking at the packet sizes being transferred. When graphing the frequency of packet sizes for each streaming provider, there was a distinct pattern for each provider. Specifically when analyzing the ratio of packets from different ranges to all of the packet sizes. These ranges included [0-200] , [200-400] and [1200+] bytes. This was the initial process that led to the creation of three features used in the final model of ratio of small, medium, and large packets.

2. RESULTS

When making the model, we first wanted to see the potential differences between the different service providers. We began by looking at the differences between Amazon Prime, Netflix, and Youtube's network traffic. All of the figures in this section are plots of every file we collected for a provider plotted together. The varying colors indicated different recordings. In using these plots we can see generalized patterns across the 100 recordings.

2.1 AMAZON PRIME

We began by looking at the upload and download patterns of Amazon Prime. Some interesting finds from the data were the surprising shelf found in the upload byte rate of the data, and the max download byte of the data. In looking at the graph of the download bytes of all the collected Amazon Prime data in Figure 1, we can see that Amazon Prime download rates can spike up to and over 8 MB, which we found to be higher than any of the other video on demand providers. These spikes were at the start of the videos, which help show that Amazon Prime has larger initial downloads rates to help get video content to start the videos. After this large initial download, the rate drops significantly to be around 1 MB or below.

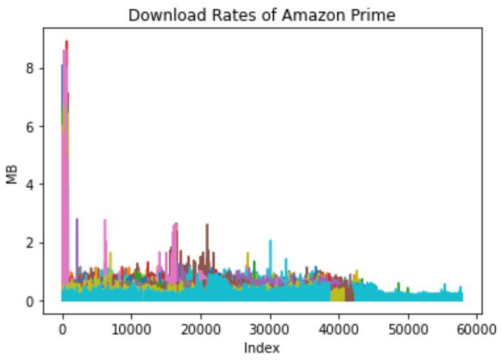


Figure 1: Download Byte Rate of Amazon Prime

The upload bytes also had an interesting pattern. When looking at the graph of upload bytes of all the collected Amazon Prime data in Figure 2, we can see clearly that there is a shelf around .06 MB. This is a consistent pattern that can help the model identify Amazon Prime Videos.

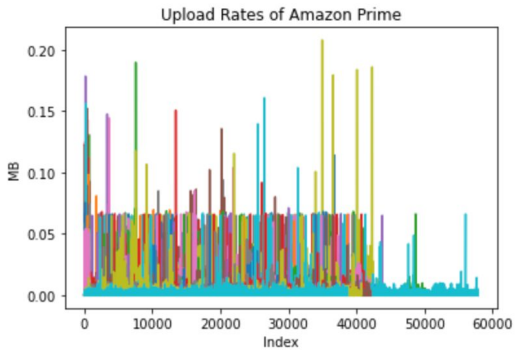


Figure 2: Upload Byte Rate of Amazon Prime

2.2 NETFLIX

Then we repeated the same process for Netflix. We found that both the upload and download patterns of Netflix are very different from those of Amazon Prime. As we can see from Figure 3, there are several spikes in the download bytes in the beginning of the collected videos, with the largest spikes located at sometime after the very start of the videos, unlike the case with Amazon Prime. Also, the general download bytes rate can go above 2MB, making the pattern more constant than that of Amazon Prime because the spikes are not as much larger than the bytes for the rest of the time. This shows that Netflix is making more constant downloads along with the videos playing and also has some initial large downloads to get the content of the videos.

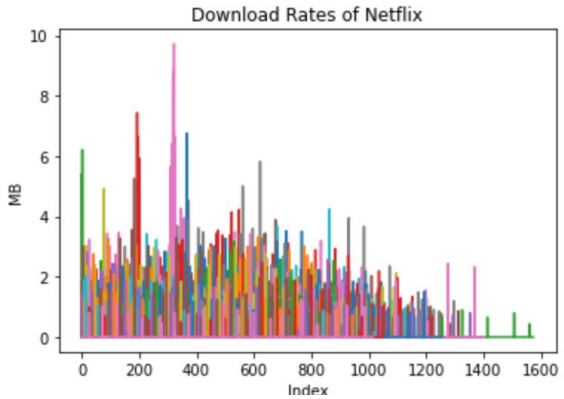


Figure 3: Download Byte Rate of Netflix

The upload rates of Netflix looks very similar to its download pattern, as seen in Figure 4. However, it is clearer that the values of upload bytes are decreasing as the videos play and there is no apparent shelf like Amazon Prime does. This pattern is rather unique and distinct to Netflix upload rates.

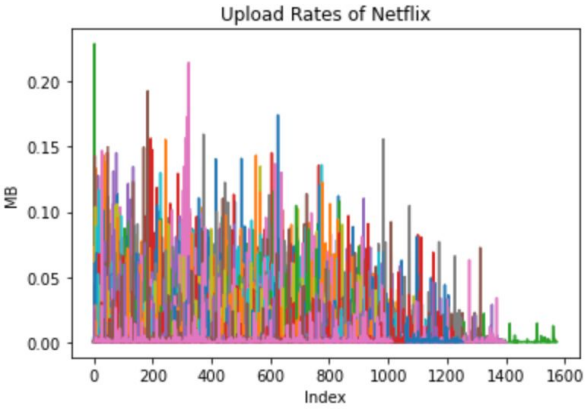


Figure 4: Upload Byte Rate of Netflix

2.3 YOUTUBE

For the download trends of Youtube, we found that there are obvious shelves around 2.9 MB and 1.9 MB. The larger shelf appears at earlier stages of the videos than the smaller one. This is unique in that Youtube downloads the videos at the most stable and consistent rate. And even though we can see some spikes in the beginning of the videos, they are only half large as those of Netflix and Amazon Prime, and very close to the download rate for the rest of the time. And unlike Netflix which has more gradual decreases of download rate over time, Youtube has decreases that are more noticeable because it has two apparent shelves as mentioned before. The small spikes and stepwise decrease over time are two important characteristics that separate Youtube data from other providers.

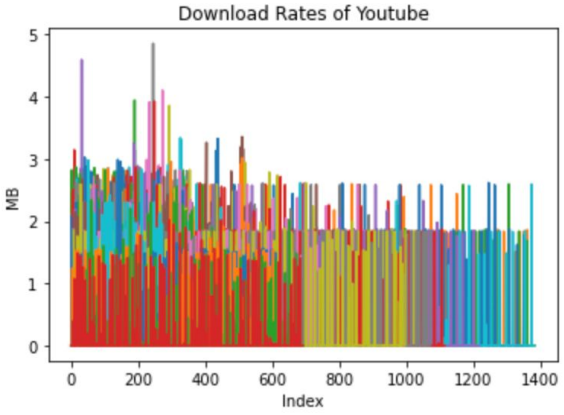


Figure 5: Download Byte Rate of Youtube

The upload byte rate of Youtube, as shown in Figure 6, is very similar to its download rate. There are no apparent

shelves in the pattern. But we can see that the rate is decreasing in two or three approximate stages. Also, the upload bytes can spike up beyond .30 MB, which are the largest among the non-live video providers we investigated and help the model distinguish Youtube videos.

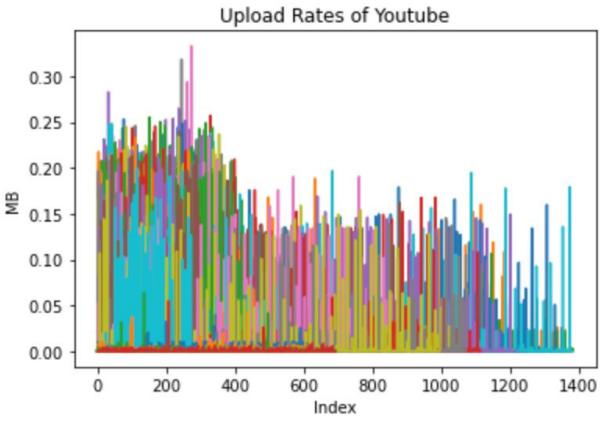


Figure 6: Upload Byte Rate of Youtube

2.4 TWITCH

Section 2.4 and 2.5 look at live video streaming. When analyzing the download rate of Twitch we found no clear pattern in the data. In fact, most files present spikes of varying length at different points in time. Moreover, it appears that the majority of the data is downloaded earlier on the streaming while maintaining a steady flow of packets toward the end. At a first glance, one would be able to tell apart Twitch's live download rate versus download rate from on demand providers.

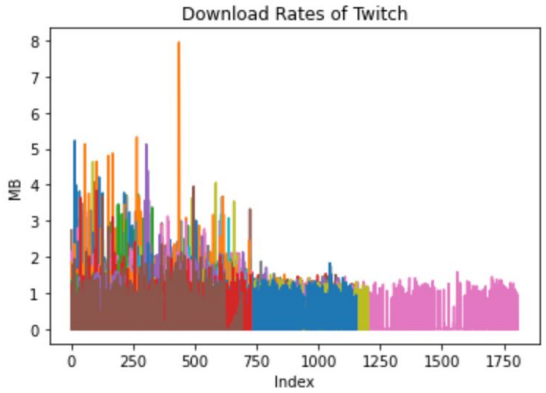


Figure 7: Download Byte Rate of Twitch

The upload rates of Twitch shown in Figure 8 stands out for there being data sent back only in the beginning of the streaming. All the files we collected show a similar behavior. In fact, at point 750 in time, the number of bytes travelling from our machine to the Twitch's server almost drops to zero with no exception.

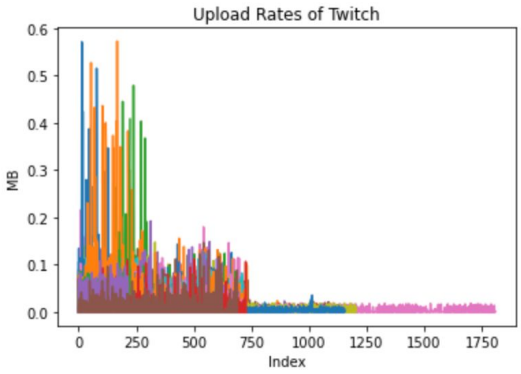


Figure 8: Upload Byte Rate of Twitch

2.5 YOUTUBE LIVE

Youtube Live is the other live streaming provider we collected data from. From the download rate we can notice a lower amount of megabytes being received as compared to Twitch. In addition, as opposed to Twitch, the download rate reveals a clearer pattern with more consistent and regular arrival of data. Moreover, around point 800 in time there appears to be a decrease in the download rate. Finally, there is an outlier that spikes several times.

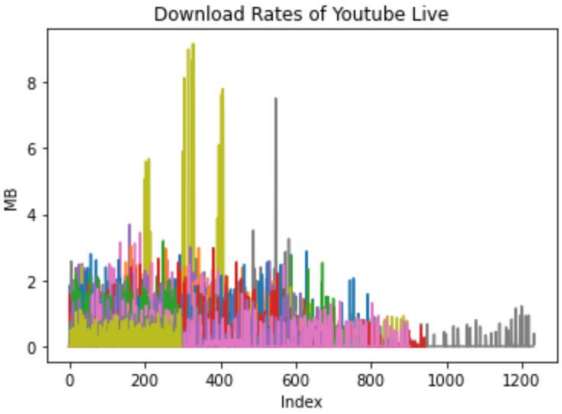


Figure 9: Download Byte Rate of Youtube Live

Youtube Live's upload rate shown in Figure 10 assimilates to Twitch's upload rate in the fact that bytes are being transferred back from the local machine up to a certain point in time before coming to a stop. However, Youtube Live

presents more constituency and regularity than Twitch. In fact, most of the files overlap with the exception of one outlier.

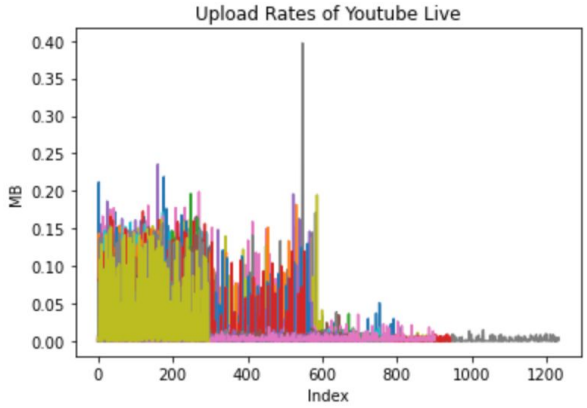


Figure 10: Upload Byte Rate of Youtube Live

2.6 OTHER PROVIDERS

At last, we looked at the download and upload patterns of Disney+, Discovery+, and Hulu, which we treated as the 'Others' category. We can see that the download pattern fairly resembles the Amazon Prime download pattern such that it also has large spikes in the beginning and then they drop below 1 MB. However, the difference between spikes and the constant rate periods are much larger than that of Amazon Prime. This can make videos from 'Others' providers identifiable but also increase the number of mistakes our model will make because of the similarity with Amazon Prime. In fact, the most misclassified cases are between 'Others' providers and Amazon Prime.

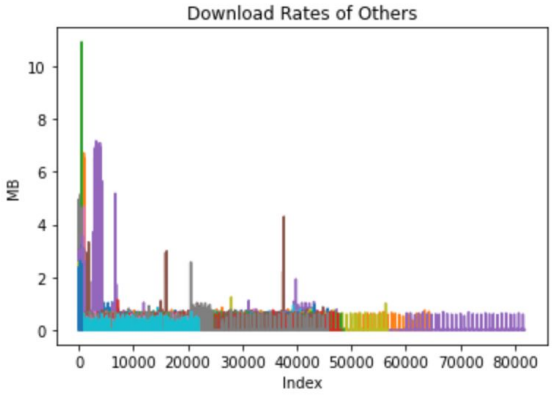


Figure 11: Download Byte Rate of Others

In the upload bytes pattern, we can see that there are multiple spikes at different times throughout the graph in Figure 12. This is potentially caused by the fact that we used three different providers Disney+, Discovery+, and Hulu, to make up the 'Others' category. Since they each may have differences in upload rate patterns, the combined pattern can be very unexpected when putting them together. Nevertheless, we can also see that mostly the upload bytes are small and they form really large height gaps with the extreme spikes. The overall pattern is still very identifiable as it is different than the upload patterns of the other providers we have looked at before.

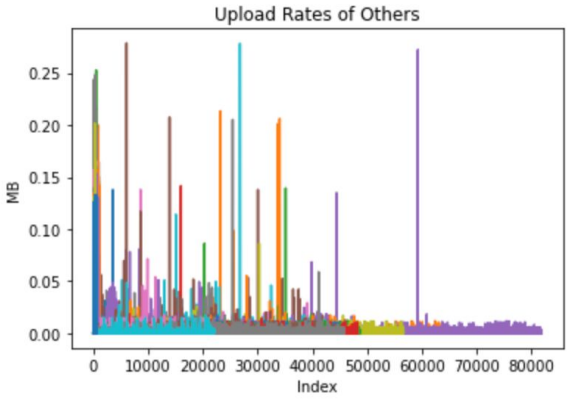


Figure 12: Upload Byte Rate of Others

2.7 CLASSIFIER

In our random forest classifier, we were able to make a model that performed with high accuracy. We chose to select a random forest classifier based on previous work. Although we all made separate classifiers for classifying streaming data versus non streaming data, we each made a model utilizing the random forest classifier. Therefore we chose to use it for this data as we saw that they already had success on similar data. We utilized features such as mean packet size, rolling delays, small and large packet ratios, byte coefficient of variation and maximum frequency prominence. The following features will be discussed in order of feature ranking as determined by the Random Forest Classifier feature importance method.

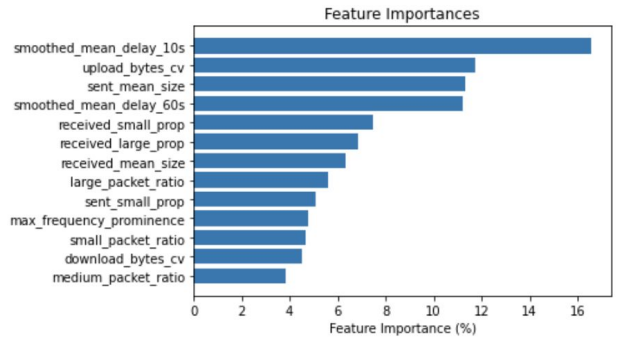


Figure 13: Feature Importances

The most significant feature in our model is the uploaded mean packet size. This is a simple, but informative metric that accounts for 16% of all feature importances. Our intuition behind the strength of this feature lies in the belief that different streaming providers would require clients to upload packets of varying sizes. Since most streaming services utilize internet protocols such as Transmission Control Protocol (TCP), for every two packets received, the client must also send in Acknowledgement packet (ACK). Although the minimum ACK packet size is 40, each streaming provider may have different ACK sizes, thus leading to a feature that can discern between providers. Similarly, mean download packet size will also vary between providers. However, we believe that this feature underperforms due to the fact that the client is constantly downloading packets with a full payload during the presence of video streaming to maintain quality and consistency.

Next, we hypothesized that each provider would transmit packets at different timings. To test this hypothesis, we first developed inter-packet delay, which measures the amount of time between the arrival of the previous packet and the following packet. To summarize the inter-packet delay for a given 90 second clip, we calculated the mean of the inter-packet delay over rolling windows of 10 and 60 seconds. These features serve to examine inter-packet arrival times in both small and large windows of the clip, which can reveal periodic patterns in how each provider transmits packets.

To further explore periodic patterns, we use signal processing methods in the frequency domain. Due to the nature of streaming, all packets should arrive at a stable frequency. Therefore, we use Welch's method to compute the power spectral density (PSD) of downloaded packet sizes. First, we resample our dataset at a consistent sample rate of 500ms. Next, we transform PSD into amplitude spectral density, which is defined as the square of PSD. This allows us to examine the prevalence of unique signaling frequencies. For example, we found that Youtube exhibits a strong download frequency at 0.2Hz. This means that for every 5 seconds, Youtube is transmitting a stable packet size. To extract this information into a feature, we calculate the maximum prominence (or magnitude of the peak) of that frequency at 0.2Hz.

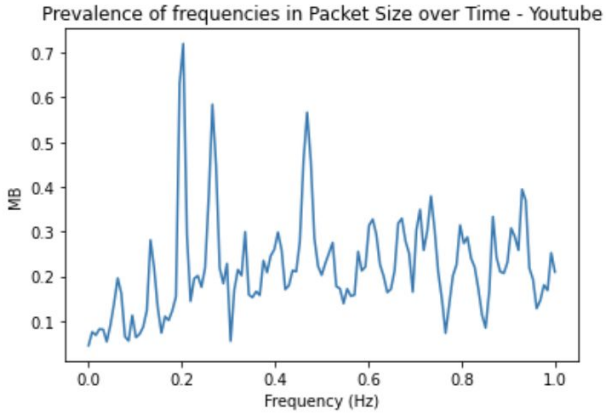


Figure 14: Prevalence of frequencies in Packet Size

Another feature is the coefficient of variation of the uploaded bytes. The coefficient of variation is the ratio of the standard deviation to the mean. By calculating the ratio for each small chunk of data, we can get the variability in regions of given datasets and find similar features to the providers our model is trained to differentiate. The upload bytes coefficient of variation patterns proved to be a key feature for our model.

Next, we focused our analysis on ratios of varying packet sizes. We used the ratio of small uploaded packets (less than 200 bytes) over the count of all uploaded packets, as well as the ratio of large downloaded packets (greater than 1200 bytes) over the count of all uploaded packets. These features allow us to analyze how each streaming provider utilizes small and large packets when receiving data from the client. Similarly, we calculated the ratio of large downloaded packets over the count of all downloaded packets. However, we chose to omit the ratio of small downloaded packets over all downloaded packets as it was not a significant feature in our model.

The next feature included in our classifier was the ratio of small packet sizes uploaded compared to the entire number

of packets. For this particular feature, it included the number of packets that were less than 200 bytes over the total number of packets in a particular dataset. Similar to the ratio of small packets, another feature incorporated in the classifier consisted of the ratio of medium packets. This means the number of packets that were between the range of 200 and 400 bytes per packet over the total number of packets in a particular dataset. The final feature follows this trend as the ratio of large packets including the number of packets that were larger than 1200 bytes per packet over the total number of packets in a particular dataset. These features were all calculated over the entire 5-minute data.

3. DISCUSSIONS

The model performed well on our test data, with 96.5% accuracy. We were able to accomplish the goal of creating a classifier that was able to differentiate between the different chosen service providers. However, there are limitations to our model and its performance.

	precision	recall	f1-score
other	0.97	0.93	0.95
youtube	0.95	0.96	0.96
amazonprime	0.93	0.97	0.95
netflix	0.98	0.99	0.98
youtube-live	0.97	0.98	0.97
twitch-live	0.97	0.93	0.95
accuracy			0.96
macro avg	0.96	0.96	0.96
weighted avg	0.96	0.96	0.96

Figure 15: Precision, Recall, F1 for each provider

One of the trends we noticed with our classifier was that it had the lowest accuracy on Amazon Prime data. When we ran a precision, recall, F1-score and support command for the different classes, we found that Amazon Prime had a precision of .93, Youtube had a precision of .95, the other category, Youtube Live, and Twitch had .97, and Netflix had a precision of .98. For our dataset and classification purposes, we understand that precision is more important than recall because the cost of false positives is higher than the cost of false negatives. Therefore, even though the Other and Twitch Live categories have lower recall scores, their high precision makes up for that loss. Moreover, with the F1 score being a balance between precision and recall, we see that our model's success in being able to achieve at least a 95% F1 score for all classes.

We then looked at the confusion matrix, seen in Figure 15 and could see that most of the misclassifications occurring were predicting that Other, Youtube, or Youtube Live was Amazon Prime. We also can see from the confusion matrix that there were a few data segments incorrectly identified as Youtube. As the other class had the most misclassifications for Prime, it is possible that the other data looked like the Prime data. This could potentially lower model accuracy, if the providers in the other category became classes in the model.

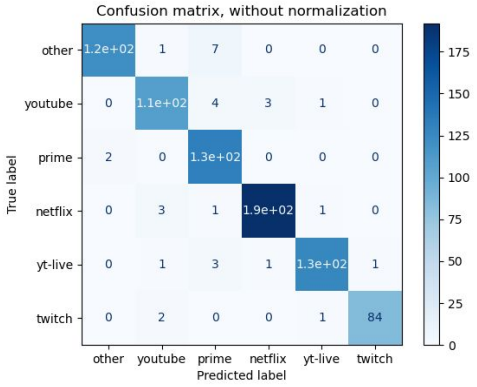


Figure 16: Confusion Matrix without Normalization

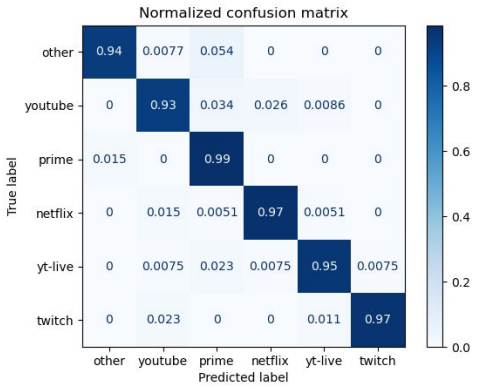


Figure 17: Normalized Confusion Matrix

Another limitation is our model was trained on clean, one time speed data, which could affect the model performance. To deploy this model for more usage cases, we would need to test it on data recorded in more conditions, such as noisy conditions, different playback speeds, and different resolutions. While it is easier to control the playback speed and noise levels during the recording, only a couple of providers allow for manual selection of resolution, so we allowed each program to select resolution based on the internet speed. Training the model on more data that covered these features would create a more robust model.

As more streaming provider platforms get released, which seems to be an increasing trend, data could be collected and used to train and test the model. Understanding the intricacies of the different streaming providers network patterns would be beneficial for ISPs to understand how to best deliver the content.

Another thing to consider in creating this model are the ethical implications. While our project is being used in an educational capacity, there are ethical implications to tracking individuals data use, especially when they are using a VPN. Although people may use VPNs to have secure connections, some positive ways to use this as an ISP would be to help based network configurations on streaming data. On the other hand, this could also allow for ISPs to learn what streaming providers their clients are using, which could be considered a breach in privacy. Even though this project could pose some ethical implications, we think understanding the network conditions that could optimize streaming for clients over a VPN is beneficial for their streaming experience.

In conclusion, there are several key takeaways from this project. First, we are able to achieve overall high accuracy in our model and other metrics like precision and F1-score are also high. Second, the high accuracy we get is a result from the features we used. We eventually decided to keep 13 features because each one contributes to making the model perform better by capturing the various differences between different providers. Also, they help increase the generalizability of our model such that the classification is not merely dependent on some features fitted exactly to the streaming providers which we have in our training data. To prove that, we added Twitch live and Youtube live to the scope of classification. Initially we had started off with just Netflix, Youtube and Amazon Prime and had an accuracy of 97%. Since our model works well on them too, we can see that the features are very generalizable as they even have good performance on live data. At last, taking the limitations of this project into consideration and trying to use this project as a basis for a working tool can be helpful for ISPs like Viasat to know their performances and improve their services.

4. REFERENCES

[1] Charles Laubach (2020) "network-stats", GitHub repository. https://github.com/Viasat/network-stats

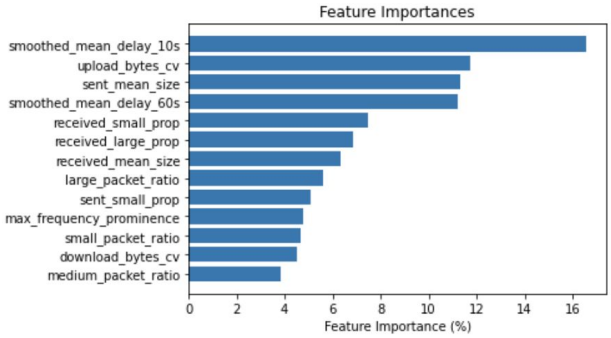
5. APPENDIX

Appendix A. Feature Calculations

1. Smoothed Mean Rolling delay 10 Seconds
    a. Mean of inter-packet delay (difference in arrival time of previous and next packet) over rolling windows of 10 seconds
2. Upload Byte coefficient of variation
    a. The Upload Byte coefficient of variation is the ratio of the standard deviation to the mean, $\boldsymbol{\sigma/\mu}$, of the uploaded byte rates
3. Mean Upload Packet Size
    a. The mean upload packet size is calculated by taking the sum of the upload packet size over the total amount of packets to get the average packet size.
4. Smoothed Mean Rolling delay 60 Seconds
    a. Mean of inter-packet delay (difference in arrival time of previous and next packet) over rolling windows of 60 seconds
5. Received Small Proportion
    a. Ratio of small downloaded packets (<200 bytes) over all downloaded packets
6. Received Large Proportion
    a. Ratio of large downloaded packets (>1200 bytes) over all downloaded packets
7. Downloaded Mean Size
    a. Mean packet size of all downloaded packets
8. Large Packet Ratios
    a. The ratio of the count of uploaded packet sizes in the size range of 1200+ bytes and the overall total number of packets. This feature is calculated over the entire dataset collected.
9. Sent Small Proportion
    a. Ratio of small uploaded packets (<200 bytes) over all uploaded packets
10. Max Frequency Prominence
    a. Using Welch's method to compute the power spectral density of downloaded packets, transformed into amplitude spectral density, then calculating the

magnitude of the peak (in bytes) at the most prominent frequency (Hz)

11. Small Packet Ratios
    a. The ratio of the count of uploaded packet sizes in the size range of 0-200 bytes and the overall total number of packets. This feature is calculated over the entire dataset collected.
12. Download Byte Coefficient of Variation
    a. The Download Byte coefficient of variation is the ratio of the standard deviation to the mean, **σ/μ**, of the downloaded byte rates
13. Medium Packet Ratio
    a. The ratio of the count of uploaded packet sizes in the size range of 200-400 bytes and the overall total number of packets. This feature is calculated over the entire dataset collected.



Appendix B. Definitions

- **Virtual Private Network (VPN)**: creates a private network across a public network
- **Packet:** formatted unit of data carrying information on where to send data and the payload of data
- **Byte:** data contained in the packet, group of 8 bits
- **Live video:** video that is being created and streamed at the same time
- **Video on Demand:** video that is created and stored on a server accessible at a later time
- **Uploaded Data:** any data uploaded by the computer to the server either requesting information or sending an acknowledgement of receiving data
- **Downloaded Data:** any data received by the computer from the server
- **Power spectral density:** describes the distribution of power into frequency components composing that signal. It is the measure of signal's power content versus frequency.

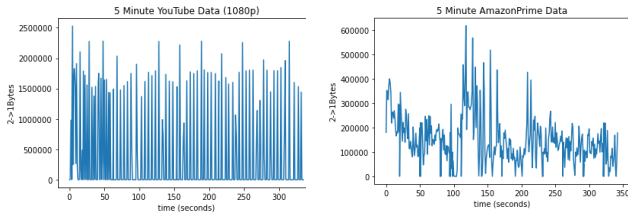
Appendix C. Project Proposal

Throughout this past quarter, we have worked with Viasat to build classifiers that are able to identify if there is video being streaming in a VPN. Using flow level data, packet level data, and self-engineered features, we have built an understanding of video patterns and signatures over a VPN. However, our current model only identifies whether a VPN user is streaming video. As an extension to our Q1 progress, this project will take a further look at classifying the streaming provider a user is using while connected to a VPN. This includes differentiating whether a streaming video is from Youtube, Netflix, Amazon Prime or others. If our classifier is successful, ISPs like Viasat can detect the presence of streaming and determine the streaming providers and use that information to optimize internet plans for certain streaming services.

As in quarter 1, we would be generating our own data using the network-stats tool provided to us by Viasat. In order to collect an abundant amount of data for our models to be trained accurately, we will use network-stats in conjunction with scripted and automated browsers to capture streaming data on various providers. For this quarter, we will be collecting VPN encrypted data and will vary the collection process on streaming providers. Similar to Q1, this data is capable of addressing our problem as it contains data on each flow and packet, including client and server IPs and ports, packet size, time and direction. It will go through similar cleaning and preprocessing steps, as well as build on top of the binary classifier from the previous quarter since we need to ensure there is video streaming present in the network data.

While there are trends across the different video service providers, each has different networking requirements as to optimize the experience for the user. At our brainstorming stage, we are considering using payload sizes and packet transfer patterns as features for our model to distinguish between providers. We hypothesize that Amazon Prime and Youtube, for example, may send packets with different sizes, interpacket delays, or size of packet clusters. Another potential feature would be buffering patterns for various providers. For example, the time-series visualization of downloaded bytes shows that Amazon Prime will buffer slowly at the start of the video, but keep the stream consistent and clear throughout the video to create an

immersive experience. On the other hand, Youtube will buffer fast at the beginning so that users may start watching their video sooner, but may continue to buffer or even slow down their buffering later into the video. This can transform into a feature by looking at packet patterns right when we begin the video stream.



As a stretch goal, we also wish to determine whether the streaming provider is sending the video at their maximum possible resolution. We would also like to add more functionality to the classifier by training it to classify more providers, such as Hulu, HBOMax, etc. Our project will be summarized in a paper explaining our findings of our machine learning model. In our paper, we will include our data collection process, EDA and feature engineering process, model selection, fine-tuning, and outputs to best communicate our findings. In addition, the model will be the main output since it will be able to run any network-stats data and classify the presence of streaming as well as the streaming provider.

Something to consider in doing this project are the ethical implications. While from an academic perspective, it is interesting to be able to understand these trends, there is concern that creating a classifier to know what streaming service you are using within a VPN could be considered an invasion of privacy. Although we are working with Viasat to understand these mechanisms, we do not intend for our classifier to be used on real client data, solely our educational dataset.