# Classification of Encrypted different Video Streams

**Objective:** Our goal is to develop a system capable of classifying network traffic using machine learning techniques, The streaming video standard has a flaw in that its recommended segmentation leads to content-dependent packet bursts even in encrypted streams. Our classification approach demonstrate that burst patterns, which are characteristic of many video streams, classifiers based on convolutional neural networks, provided that the network measurements are relatively to each other.

# Background-

**Limitations of Traditional Identification Techniques**

* Packet Inspection Traditional methods of packet inspection and deep packet analysis are

rendered ineffective by the use of end-to-end encryption, which obscures the video content and metadata.

* Statistical Analysis Relying on statistical features of the video stream, such as bitrates and frame sizes, has proven to be unreliable in the face of modern adaptive streaming technologies.

# The Concept of "Beauty and the Burst"

* **Video Encoding** The encoding process of video streams creates distinct "burst" patterns in the network traffic, even when the content is encrypted.
* **Burst Analysis** By examining these burst patterns, researchers have discovered a new approach to identifying the source and content of encrypted video streams.
* **Non-invasive Monitoring** This "Beauty and the Burst" technique allows for the remote

identification of encrypted video streams without the need for deep packet inspection or access to the encryption keys.

**Data Collection**: We concentrated on four well-known streaming services: We carefully selected a small number of popular TV shows, with up to a few episodes per series, from each provider for our proof-of concept tests. In all, we used titles from Vimeo, YouTube, and other sources. We started a Chrome browser instance for each title and utilized a "rewind" method unique to the service to start playing from the beginning of the material. The first buffering phase of films with an introductory title sequence

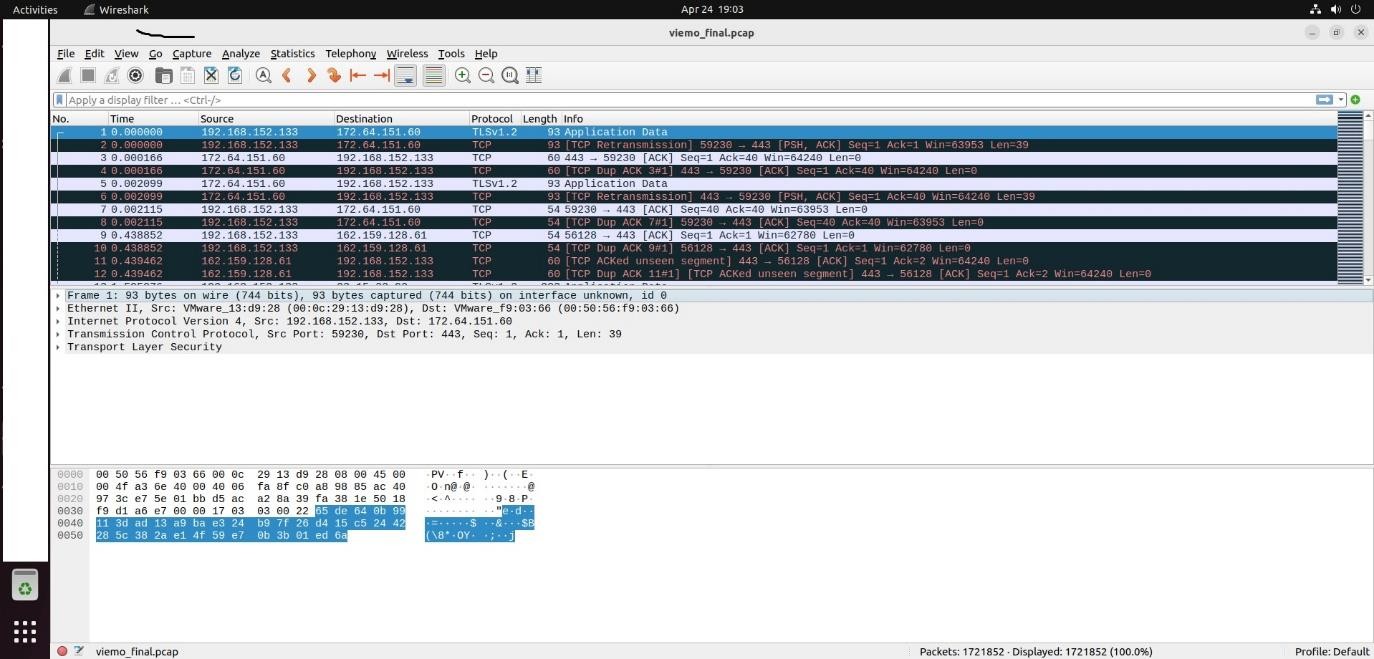
downloads this (non-unique) sequence; the bursts in the on-off phase correspond to the unique content segments. Using Wireshark's tshark, tcpdump we recorded each streaming session's network traffic for a specific amount of time.

**Automated capture** by selenium. For each title, we spawned a Mozilla browser instance

* Selenium is a powerful tool for controlling web browsers through programs and performing browser automation. It is functional for all browsers, works on all major OS and its scripts are written in various languages
* selenium Firefox can be controlled by Python. To do this you need the selenium module and a web driver.
* Gecko driver is what is between Selenium and the Fire Fox browser. It lets you control the Firefox web browser from Python code

# Traffic Captured-

* We captured the network traffic of each streaming session for a certain duration through TCP dump. For Amazon, Vimeo, you tube
* From each capture, we kept only the TCP flow.
* We streamed and captured each of the 20 selected video different title, and each of from the automated crawl. Bitrate for YouTube videos varies. we took 3-minute tcp dump captures and cropped the captured streaming flows to 3 minutes. This is for all you tube Vimeo, amazon.
* Data contains the following field
  + Time
  + Source
  + Destination
  + Protocol
  + Length and info



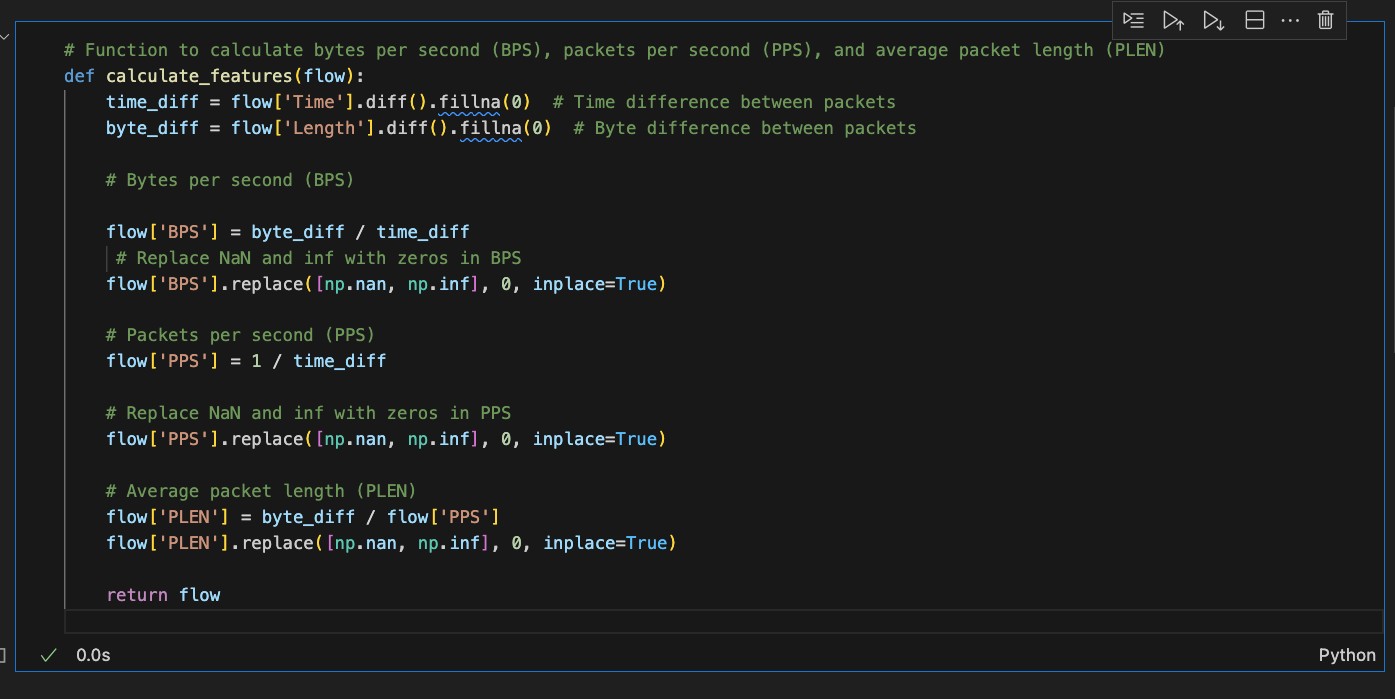
# Figure: Data collection with PCAP file

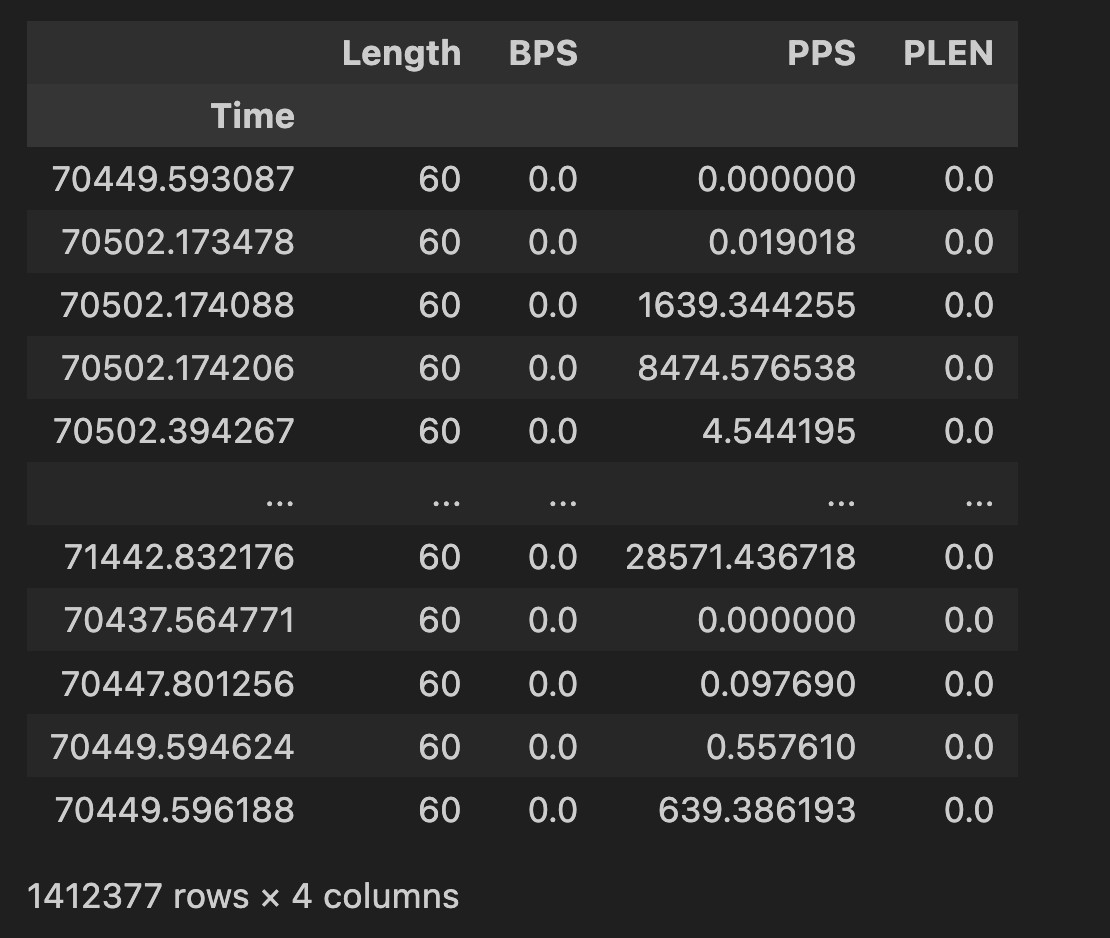
**Data Preprocessing**: Upon capturing the data, we are thinking of applying preprocessing steps to prepare it for feature extraction and classification (prob based on CNN). Data preprocessing is related data

cleaning also that is removing NA values, normalization of data etc.

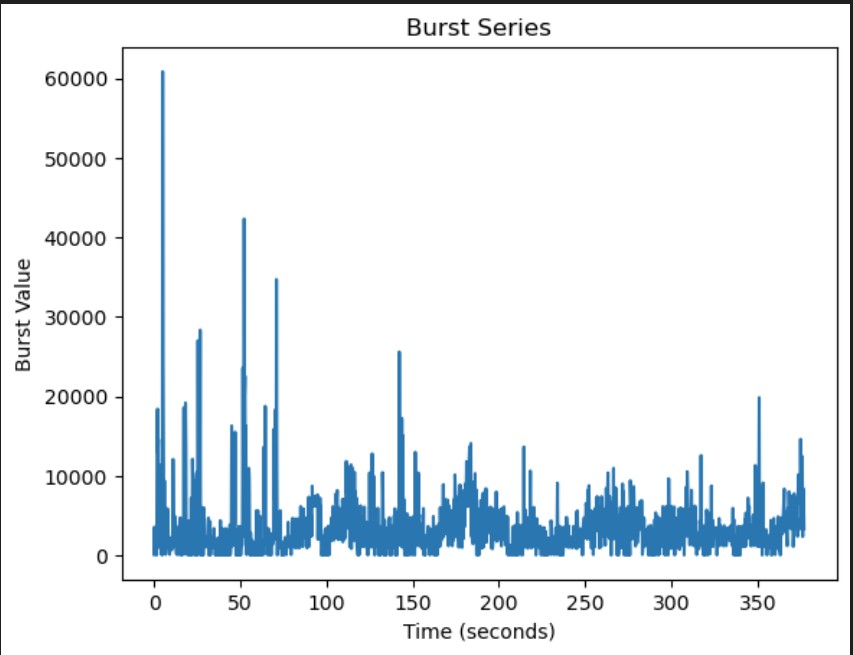
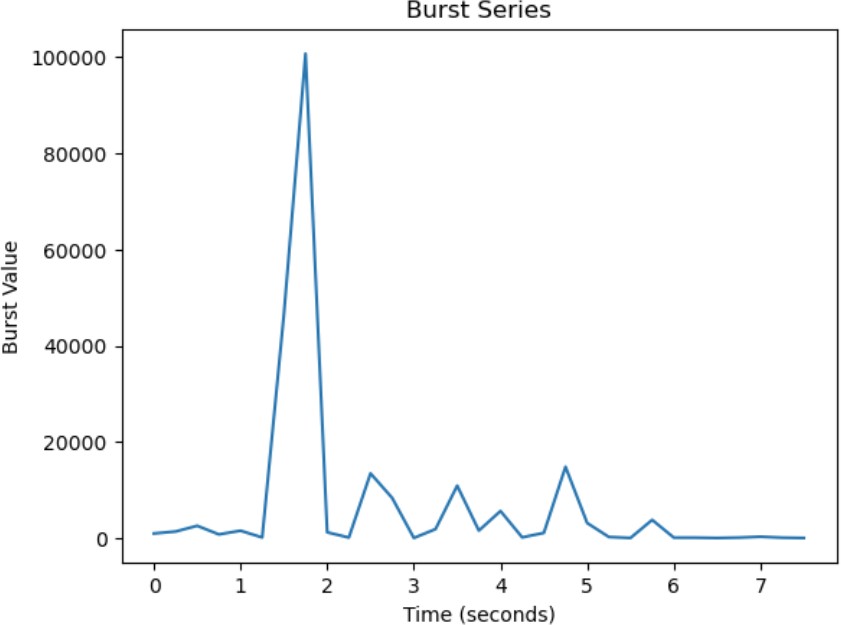
**Feature Extraction**: We will extract relevant features from each packet in the captured data. Features will include packet length, time since the last packet, and protocol type, IAT, etc. among others. We will extract the time series of the following flow attributes from each capture: down/up/all bytes per second (BPS), down/up/all packet per second (PPS), and down/up/all average packet length (PLEN). We retained just the TCP flow with the highest bit count from each capture. We averaged across 0.25-second intervals to aggregate the series into 0.25-second chunks for the purpose of creating vectors with consistent sizes.

1. Filter the TCP
2. Group TCP flows by Source and destination
3. Calculate the BPS (bytes per second), PPS (packet pr second and PLEN (Packet Length)
4. Drop Source IP, Destination IP, Protocol and Info fields





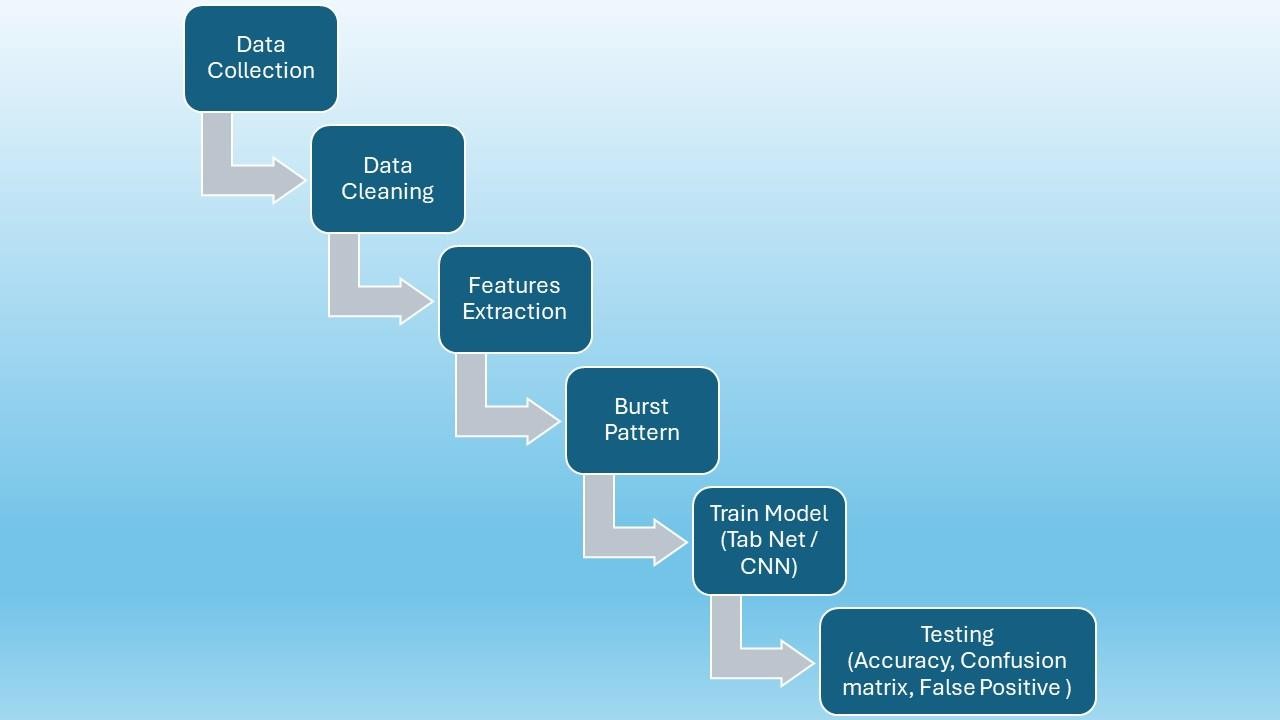
* A burst series is a series where every point corresponds to a burst.
* In our experiment, we used I = 0.5 seconds



**Model Selection and Training**: TensorFlow is used in conjunction with the Keras front end to construct the classifier. We will apply the 0.7-0.3 train-test split, randomly shuffle the samples, then train for a

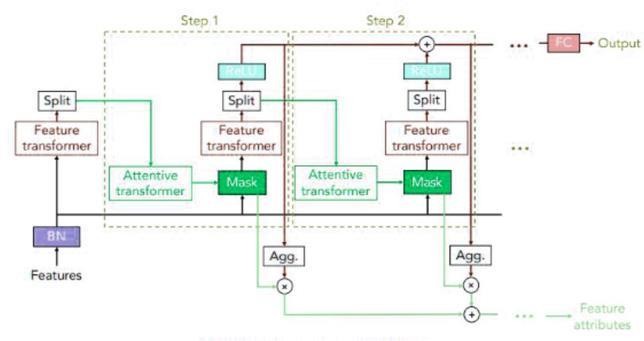
predetermined number of epochs for each job. The dataset was normalized on a feature-by-feature basis

by dividing the maximum of the aggregated values of each sample's time-series vector representing a particular feature.



# Tab Net

Tab Net was proposed by the researchers at Google Cloud in the year 2019. The idea behind Tab Net is to effectively apply deep neural networks on tabular data which still consists of a large portion of users and processed data across various applications such as healthcare, banking, retail, finance, marketing, etc.

fdsfsdf

dsf

f

**Summary Report**: This report summarizes the performance metrics obtained from multiple experiments conducted on the dataset. Each row corresponds to a separate experiment, and the columns represent different performance metrics.

| **Experiment** | **Accuracy** | **Precision** | **Recall** | **F1 Score** |
| --- | --- | --- | --- | --- |
| Experiment 1 | 0.9713 | 0.9822 | 0.9770 | 0.9796 |
| Experiment 2 | 0.8994 | 0.9812 | 0.8750 | 0.9250 |
| Experiment 3 | 0.8429 | 0.8421 | 0.9560 | 0.8954 |
| Experiment 4 | 0.9671 | 0.9831 | 0.9695 | 0.9762 |
| Experiment 5 | 0.8879 | 0.9851 | 0.8542 | 0.9150 |

**Overall Summary:**

* **Accuracy:** Mean = 0.9135, Std = 0.0450
* **Precision:** Mean = 0.9547, Std = 0.0606
* **Recall:** Mean = 0.9264, Std = 0.0442
* **F1 Score:** Mean = 0.9382, Std = 0.0291

The overall summary provides an analysis of the performance metrics obtained from a series of experiments conducted on the dataset. These experiments involved training and evaluating machine learning models for a specific task.

* **Accuracy:** The accuracy metric measures the proportion of correctly classified instances out of all instances in the dataset. The mean accuracy across all experiments is 0.9135, with a standard deviation of 0.0450. This indicates that, on average, the models achieved a high level of accuracy in predicting the correct class labels. However, there is some variability in performance across different experiments, as evidenced by the standard deviation.
* **Precision:** Precision quantifies the ability of the model to correctly identify positive instances out of all instances predicted as positive. A high precision value indicates that the model makes fewer false positive predictions. The mean precision across all experiments is 0.9547, with a standard deviation of 0.0606. This suggests that, on average, the models demonstrated a strong ability to avoid false positive errors, although there is some variation in precision across experiments.
* **Recall:** Recall, also known as sensitivity, measures the proportion of true positive instances that were correctly identified by the model out of all actual positive instances. A high recall value indicates that the model captures a large proportion of positive instances. The mean recall across all experiments is 0.9264, with a standard deviation of 0.0442. This implies that, on average, the models exhibited a high level of sensitivity in identifying positive instances, although there is some variability in recall across experiments.
* **F1 Score:** The F1 score is the harmonic mean of precision and recall, providing a balance between the two metrics. It is a single metric that considers both false positive and false negative errors. The mean F1 score across all experiments is 0.9382, with a standard deviation of 0.0291. This indicates that, on average, the models achieved a good balance between precision and recall, although there is some variability in F1 score across experiments.

Overall, the performance metrics demonstrate that the models trained on the dataset performed well in terms of accuracy, precision, recall, and F1 score. However, there is some variability in performance across different experiments, highlighting the importance of conducting multiple experiments and considering the variability in model performance.