Encrypted Video Traffic Clustering Demystified

Amit Dvir[1], Angelos K. Marnerides[2], Ran Dubin[1], Nehor Golan[1], Chen Hajaj[3,4]
[1]*Cyber Innovation Center, Department of Computer Science, Ariel University, Israel*
[2]*InfoLab21, School of Computing and Communications, Lancaster University, UK*
[3]*Data Science and Artificial Intelligence Research Center, Ariel University, Israel*
[4]*Department of Industrial Engineering and Management, Ariel University, Israel*
Corresponding Author: Dr. Amit Dvir, Golan Heights 65, Ariel, amitdv@g.ariel.ac.il

***Abstract*—Cyber threat intelligence officers and forensics investigators often require the behavioural profiling of groups based on their online video viewing activity. It has been demonstrated that encrypted video traffic can be classified under the assumption of using a known subset of video titles based on temporal video viewing trends of particular groups. Nonetheless, composing such a subset is extremely challenging in real situations. Therefore, this work exhibits a novel profiling scheme for encrypted video traffic with no a priori assumption of a known subset of titles. It introduces a seminal synergy of Natural Language Processing (NLP) and Deep Encoder-based feature embedding algorithms with refined clustering schemes from off-the-shelf solutions, in order to group viewing profiles with unknown video streams. This study is the first to highlight the most computationally effective, accurate combinations of feature embedding and clustering using real datasets, thereby, paving the way to future forensics tools for automated behavioral profiling of malicious actors.**

***Index Terms*—Encrypted Traffic, Video Title, Clustering, YouTube, NLP**

1. Introduction

Governmental law enforcement bodies in developed countries, the UN Counter Terrorism Committee, and the Security Council have stress the importance of tracking terrorists through profiling the Internet-wide behaviour of extremist propaganda groups [1]. Unfortunately, recent events such as the March 2019 mosque shootings in Christchurch, New Zealand, have prompted militant groups to broadcast their hate speech through online videos in various social networking sites and YouTube [2]. In parallel, the various shootings in the US since 2013 up to the latest one in Virginia Beach in May 2019 highlight the correlation of the shooters' behaviour with viewing content through popular video streaming sites such as YouTube and Netflix [3].

With the introduction of General Data Protection Regulation (GDPR), there is a justified act towards the support of human rights. Nonetheless we are also now witnessing the weakness of law enforcement agencies in preemptively tracking extremists and hate groups through the Internet. Thus profiling the online video viewing behavior of such groups has become quite a challenging task due to the encrypted nature of the Internet video streams through protocols such as HTTPS [1]. Therefore intelligence agencies, as well as forensics investigators, are no longer capable of utilizing conventional traffic analysis techniques such as Deep Packet Inspection (DPI).

Given the various unavoidable limitations on traffic classification tasks, a number of studies have looked into the analysis of encrypted Internet traffic. Nevertheless, little has been done in the context of explicitly dealing with encrypted video traffic. Most importantly, the majority of encrypted video classification and clustering studies rely on assumptions that are not necessarily true in some scenarios. For instance, the work by Dubin et al. in [4] considers a pool of video titles for conducting supervised classification of encrypted transport layer flows (e.g., TCP, UDP) obtained by YouTube video streams. The applicability of the proposed scheme was demonstrated in the scenario where an external attacker could identify a video title from the video streams as long as the title of the video stream was from a given video title set. Similarly, the studies in [5] and [6] also utilised a number of supervised Machine Learning (ML) algorithms by considering already known video titles associated with statistical properties of encrypted network flow. Hence, none of these studies consider the pragmatic scenario in which encrypted video streams may be clustered or classified with no knowledge of any video titles. In fact, the majority of encrypted video streams from most content providers include the video titles in the encrypted payload information of network flow, hence the aforementioned techniques would be greatly affected in terms of classification accuracy.

In order to confront the challenging task of clustering encrypted video with no a priori knowledge of video titles, the preliminary work in [7] exploited the usefulness of Natural Language Processing (NLP). The proposed solution was composed of a single clustering method relying on a language derived by meta-statistics of network traffic features per encrypted video stream. However, the selection of statistical features within the clustering process were manually selected by empirically observing the distributional behaviour. Thus, there was minimal automation with reasonably high computational costs in some instances.

Therefore, this work goes beyond previously introduced pieces of work and extends the preliminary findings

1. COBWEBS Technologies on EU GDPR policies: https://www.cobwebs.com/crime-investigations-eu/

in [7], by introducing a seminal synergy of NLP and Deep Auto Encoder-based feature embedding algorithms with refined clustering schemes such as grouping viewing profiles with unknown video streams. The empirical observation of the distribution of statistical features is neglected and an automated feature selection method with three well-known NLP techniques: Bag of Words, Auto-Encoder and a sample embedding using different affinity function is employed. Moreover, off-the-shelf clustering algorithms, Gaussian Mixture Model, Hierarchical Clustering and Spectral clustering are refined to cluster the encrypted network traffic video streams. The complete clustering methodology is evaluated under several metrics such as cluster purity and silhouette value.

The main contributions of this paper are summarised as follows:

1) The first study to assess encrypted traffic clustering when video title information is not available.
2) Novel integration of NLP with refined off-the-shelf clustering that enable high clustering accuracy.
3) An automated procedure for grouping encrypted video traffic in order to aid the design and implementation of future network forensics tools.

The remainder of this paper is structured as follow: Section 2 provides a summary of related work and highlights the novelty behind the scheme, whereas Section 3 describes the dataset used in this work. Section 4 is dedicated to presenting the methodology employed in this work. Section 5 discusses the evaluation undertaken in this paper and, finally, Section 6 concludes and summarizes the paper.

2. Related Work

Numerous studies have shown that traffic encryption is not sufficient for a large number of cases in both desktop and mobile devices since sensitive meta-data can be derived with the use of various tools and methods. The majority of reported studies focused on classification schemes to profile features such as the operating system version, user's browser, applications and user activity [8]–[11].

A smaller number of studies aimed at clustering groups of encrypted traffic using only network statistics [12]–[14]. For instance, Erman et al [12] identify Internet application protocols (e.g., HTTP, FTP) by clustering meta-features obtained by transport layer protocol (i.e., TCP, UDP) flow statistics and highlight the usefulness of unsupervised clustering schemes in such tasks.

Bacquet et al. [13] introduce the applicability of the Multi-Objective Genetic Algorithm (MOGA) for grouping application protocols from encrypted traffic whereas Hochst et al. [14] use a neural auto-encoder in order to classify traffic flows originated by mobile applications.

For more than a decade various studies were conducted explicitly on YouTube traffic patterns. Hence, there were works ranging from the identification of YouTube server locations up to the comparison with alternative Internet video streaming applications and profiling of desktop against mobile user YouTube access patterns [15]. Moreover, YouTube-specific analysis was performed in the context of Quality of Experience (QoE) as well as general transport layer traffic characterisation [16].

A fairly small proportion of studies addressed the profiling of encrypted video traffic streams. As far as can be known, all pieces of work performed supervised classification or clustering schemes [4], [5].

Stikkelorum [5] used match video segments and reward segments in order to identify the video title while Reed and Kranch [6] used direct network observations to identify Netflix video streaming. Schuster et al. [17] implemented a Convolutional Neural Network (CNN) in order to accurately identify video streams based on their uniquely characterized burst patterns.

Nonetheless, all the aforementioned studies were restricted to classifying video streams with access to a known set of labeled video titles. Hence, as far as can be known, there is no method that explicitly attempted to profile user behaviour by clustering encrypted video streams with no prior knowledge of video titles. Most importantly, none of these studies highlighted the usefulness of such schemes in the context of automated digital forensic applications.

3. Dataset Description

As depicted in Table 1 the dataset consists of 10,000 YouTube video streams (100 video titles, each title accessed 100 times). Over a period of 6 months Chrome was used as a browser with a real-world Internet connection under different real-world network conditions. The previous work in [4] demonstrates that the downloading rate behaviour of YouTube videos is browser-independent, thus the data pulling method considered the Chrome browser. In parallel, Chrome was considered since it has been shown to contain a large pool of open-source plugins that facilitate web automation tools. Hence, the synergistic use of Selenium[2] and the ChromeDriver[3] enabled the researchers to emulate normal user downloading behaviour and build up a realistic dataset. The auto mode was utilised in each download stream, where the player decides which quality representation to download based on estimations of the client's network conditions and application parameters.

TABLE 1: Dataset Parameters

Total number of video streams	10,000
Video types	News, Sports, Nature, Trailer, GoPro
Browser	Chrome
Automation	Selenium
YouTube player	Auto Mode

In general, the analysis does not consider any prior knowledge regarding the transport flow variability over each video stream. Hence, the dataset consists of a mixture of TCP or UDP flows using one or more connections as well as varying video quality types. Moreover, the dataset includes popular YouTube videos from five different categories in order to enrich the realistic assessment in the study as a whole. Essentially, the consideration of diverse categories firstly enables a better approach to assess the

2. Selenium: https://www.seleniumhq.org/
3. ChromeDriver: http://chromedriver.chromium.org/

clustering outputs presented in the following sections, as well as highlighting the usefulness of the scheme in real-life automated forensics activities when user-centered profiling is required.

4. Methodology

Fig. 1 exhibits a high-level view of the methodology as conducted in this study. As depicted, firstly a sanitisation module is triggered where re-transmissions and audio packets are eliminated from the video stream in a similar fashion as in the previous work in [7]. Moreover, Bits per Peak (BPPs) are computed for each encrypted video stream via re-assembling network flows by using the 5-tuple packet-level information [4] between two Off periods within the observed peaks [7].

The following step focuses on feature engineering such as firstly identifying the most statistically significant features related to the raw measurements in the dataset. Therefore, data mining and embedding techniques are utilised initially in order to filter out insignificant raw measurements and reduce data dimensionality. Subsequently, an NLP-based approach is adopted in order to compose meaningful meta-features envisaged for use within clustering components, as discussed in Section 4.1.

The fourth step, as evidenced in Fig. 1, is concerned with triggering the selected unsupervised clustering algorithms discussed in Section 4.2. Essentially, the purpose of this module is to group the video viewing behaviour of the encrypted video streams and indicate the corresponding clusters of common viewing patterns.

In order to assess the significance of the clusters and validate the accuracy of viewing groups each clustering procedure is evaluated by calculating well known measures such as the number of pure clusters (NPC), cluster purity (CP), average silhouette value (ASV), number of streams in clean bins (NSCB), and cleanly clustered streams (CCS). Note that, the methods described are different alternatives, i.e., both an embedding and a clustering algorithm must be chosen. Combining different embedding techniques, or different clustering techniques is not considered.

4.1. Embedding and Feature Engineering

The direct use of raw data in a machine learning method is problematic since raw data tend to be unstructured and they normally contain redundant information. A natural, clear solution is to initially perform feature extraction and then feature selection in order to create a structured representation that in parallel is tailored to the specific problem domain.

There follows a novel synergy of embedding techniques for feature extraction. The use of the base NLP and embedding algorithms is then justified further indicating the contributions towards refining them such as to address the explicit problem of encrypted video traffic clustering. Table 2 summarizes the abbreviations and notations used in this paper.

4. The 5-tuple packet-level information is defined by the source/destination IP port, source/destination IP address and the protocol (i.e., TCP/UDP).

DPI	Deep Packet Inspection
ML	Machine Learning
NLP	Natural Language Processing
BPP	Bit Per Peak
NPC	Number of Pure Clusters
NSCB	Number of Streams in Clean Bins
CP	Clustering Purity
ASV	Average Silhouette Value
CCS	Cleanly Clustered Streams
AHC	Agglomerative Hierarchical Clustering
CNN	Convolutional Neural Network
GMM	Gaussian Mixture Models
AFF	Affinity
w_i	i'th word
win	Word2vec window size
$C_i(win)$	Words in the i'th window
K	Number of Clusters
X	Matrix of samples
X'	Matrix of predicted samples by the Auto Encoder
X_i	i'th sample
S, S'	Streams as sets of BPPs
L_2	Norm
m_k	Centroid of k'th cluster
C_k	The set of Samples in the k'th cluster
μ_i	The mean of the i'th component
σ_i	The variance of the i'th component

TABLE 2: List of abbreviations

4.1.1. Word2Vec. The core principle of NLP is to understand the meaning of a given word. Although human-like ability to understand languages remains elusive, certain methods have been successful in capturing similarities between words. Recently, such approaches based on neural networks in which words are embedded into a low dimensional space have been proposed by various studies [18]. The proposed models represent each word as a *d-dimensional* vector of real numbers, where vectors that are close to each other are shown to be semantic.

Mikolov et al. [19], [20], enabled an efficient embedding of words that achieves striking results on various linguistic tasks by a skip-gram with a negative-sampling training method. Unlike most previously used neural network architectures for learning word vectors, in [19], [20] the training process of the skip-gram model does not involve dense matrix multiplications. Rather, the models operate in large amounts of unstructured text data to learn high quality vector representations of words. Thus, for a sentence of n words $w_1, \ldots, w_n$, contexts of a word w_i come from a window of size win around the word: $C_i(win) = \{w_{(i-win)}, \ldots, w_{(i-1)}, w_i, w_{(i+1)}, \ldots, w_{(i+win)}\}$, where win is a parameter for the window size and C_i is the context. The window size win can be either static or dynamic; if dynamic, win denotes the maximal window size and for each word in the corpus, win is sampled uniformly from $[1, n]$.

Goldberg et al. in [21] show that smaller windows induce more synonymic and functional models while larger windows induce embeddings that are more associative or topical. However, the window size effect on the encrypted network traffic clustering is unknown and is further discussed here. In the problem domain, clustering unknown video titles, a BPP (number of bits between On/Off period) is defined as a word w_i by converting the integer value of the BPP to a word. Similarly, a vector of BPPs becomes a sentence of n words by converting a set of BPPs from

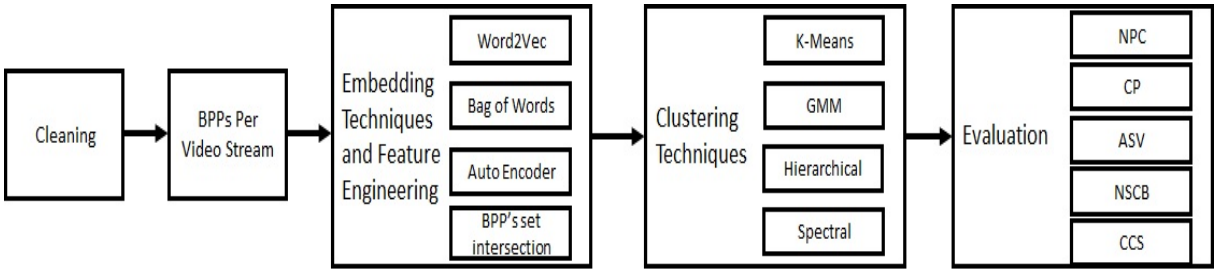


Figure 1: Methodology steps for clustering encrypted YouTube video streams

integers into a set of words.

In order to optimise the feature engineering process was necessary to refine the Word2Vec modeling approach while fine-tuning a number of parameters including the window and embedding size, at the same time. Moreover, a recursive solution that evaluates the impact of the affected parameters had to be defined such as to assess the overall performance of the refined Word2Vec model. The naïve solution would be to define a validation set on which to evaluate the effect of those parameters. However, this solution is not valid in the world of unsupervised learning due to a lack of pre-defined labels for validation.

Therefore, it was decided that the best course of action would be to evaluate the effect of the parameters on the clustering using the average Silhouette value (ASV) [22]. The ASV metric lies in the range $[-1, 1]$ and indicates the tightness of the clustering separation. For each data point, the Silhouette value indicates how close the given data point is it to it's neighboring clusters. When $ASV = -1$ it is indicated that the point is close to a neighboring cluster. In general the value of 1 is preferred in order to exclude any relationship to neighboring clusters.

As depicted through Algorithm 1, for each video stream (sentence) the BPPs (words) were extracted and transformed into a vector of strings instead of a vector of integers (Algorithm 1 lines 2-8).

A grid search was performed using all possible tuples of the model parameters and measuring the average Silhouette value that the clustering algorithm achieved (Algorithm 1 lines 11-21). When all the results were gathered in, the model associated with the parameters that performed the best, was chosen and evaluated in order to estimate the clustering accuracy performance.

4.1.2. Bag Of Words. Another common NLP technique, the "Bag of Words", was applied in order to create a well-structured feature vector with a defined size for the encrypted video streams. In the "Bag of Words" technique, each sample is defined by a vector equal to the vocabulary size, meaning the number of unique words (Algorithm 2 lines 1-4). The vectors thus produced are initialized with zeros (Algorithm 2 line 4), and for every word in the sample, 1 is added (the number of appearance is counted) in the appropriate cell (Algorithm 2 lines 5-6).

Note that in some implementations (e.g. in [23]), as well as in this paper, the cells act as indicator variables stating whether a word appears or not, without regard to the number of appearances. Therefore, this methodology holds whether specific words appear in the sentence, not their position or relationship to the position.

Algorithm 1 Word2Vec Embedding

Input: $Streams$ - a list of all the video streams, each stream is a list of BPP's
Output: The embedded streams

1: $str \leftarrow \{\}$
2: **for each** $stream \in Streams$ **do**
3: $\quad Curr\_Stream \leftarrow \{\}$
4: $\quad$ **for each** $BPP \in Stream$ **do**
5: $\quad\quad Curr\_Stream.append(string(BPP))$
6: $\quad$ **end for**
7: $\quad str.append(Curr\_Stream)$
8: **end for**
9: $BestEmbd = \{\}$
10: $BestSLV = -1$
11: **for each** possible $window\_size$ **do**
12: $\quad$ **for each** possible $embd\_size$ **do**
13: $\quad\quad embd \leftarrow W2V(win\_size, embd\_size, str)$
14: $\quad\quad Clusters \leftarrow Clustering(embd)$
15: $\quad\quad Curr\_slv \leftarrow Silhouette\_value(Clusters)$
16: $\quad\quad$ **if** $Curr\_slv \leq BestSLV$ **then**
17: $\quad\quad\quad BestSLV \leftarrow Curr\_silhouette$
18: $\quad\quad\quad BestEmbd \leftarrow Curr\_Embedding$
19: $\quad\quad$ **end if**
20: $\quad$ **end for**
21: **end for**
22: **Return** $BestEmbd$

Algorithm 2 Bag of Words

Input: $Streams$ - a list of all the video streams, each stream is a list of BPP's
Output: The embedded streams

1: $Max \leftarrow MaximalPeak(Streams)$
2: $BagOfWords \leftarrow \{\}$
3: **for** $Stream \in Streams$ **do**
4: $\quad Curr \leftarrow \{0, 0, ..., 0\}$ ▷ An array of length Max
5: $\quad$ **for** $BPP \in Stream$ **do**
6: $\quad\quad Curr[BPP] \leftarrow Curr[BPP] + 1$
7: $\quad$ **end for**
8: $\quad BagOfWords.append(Curr)$
9: **end for**
10: **Return** $BagOfWords$

4.1.3. Deep Auto Encoder embedding. As revealed through the feature engineering process, the number of a possible BPP value is much higher in comparison to the number of BPPs in each encrypted stream. Consequently, every produced "Bag-of-Words" vector had high sparsity properties. In order to address matrix sparsity and further cope with the high dimensionality and lack of relevant information, Deep Auto Encoder outputs are used [24].

The goal of Deep Auto Encoders is to assemble a neural network with the aim of producing a compact data embedding output. This aim is achieved by setting each layer to be a smaller size until reaching a bottleneck. The bottleneck is then connected to layers of increasing size, and the network output is set to be equal to the size of the original input.

Let X be the sample matrix and X' the matrix of the re-encoded samples. The loss of the neural network is obtained by:

$$||X - X'||_2^2 \tag{1}$$

Hence, the network will try to embed the data into a lower yet information-preserving dimension and then reassemble it. After training the network, all the layers going out of the bottleneck are removed, and each sample is run through the modified network, in order to obtain a compact data embedding output.

In contrast to other pieces of work (e.g. [24]), where clustering and embedding is done separately, the Bag of Words is used in synergy with a deep auto encoder to enable uniform embedding and clustering technique as well as automated optimization for the clustering process. Moreover, a direct assessment of the clustering accuracy performance is achieved under the aforementioned synergy.

4.1.4. BPPs set Intersection. The earlier work in [4] suggested supervised learning techniques for classifying encrypted streams with a priori knowledge of known video titles. In parallel, an unsupervised clustering scheme derived by the k-nearest neighbors (KNN) algorithm was also utilised. In this work, this KNN formulation is exploited with a new affinity function. In practice, an affinity function is a measure of similarity between two samples. Ultimately, two distinct samples should have lower values while similar samples should have higher values.

The affinity score between two BPP sets, S and S', is defined as the cardinality of the intersection set, where M, and M' are BPP sets:

$$AFF(S, S') = |S \cap S'| \tag{2}$$

The defined affinity function is employed within the evaluation phase for the Spectral and Hierarchical clustering algorithms as described in Sections 4.2.3 and 4.2.4.

4.2. Clustering Techniques

Unsupervised clustering methods learn a function from a set of unlabeled examples by using heuristics. Based on the problem domain reported here, unknown encrypted video streams are clustered to determine whether there are streams of the same video title, without an a-priori knowledge of the video title. The following is a description of the algorithmic properties of the unsupervised learning methods employed in this work.

4.2.1. K-means. The k-means algorithm approximates a division of the dataset into K distinct clusters of equal variance where each cluster is described by its mean. Hence, K-means attempts to find the mean values that minimize the intra-cluster sum of squares; i.e., the squared Euclidean distance between all samples within a cluster and its respective mean.

K-means split the data into K clusters by iteratively optimizing the following problem: given set of points $X = \{x_1, \ldots, x_n\}$, where $m_k = \sum_{i \in C_k} x_i / n_k$ is the centroid of cluster C_k the algorithm tries to minimize the following cost function:

$$J_k = \sum_{k=1}^{K} \sum_{i \in C_k} (x_i - m_k)^2 \tag{3}$$

4.2.2. Gaussian Mixture Models. Gaussian Mixture Models (GMM) are probabilistic models that have demonstrated their applicability in the context of unsupervised clustering. Within a GMM-based learning formulation, data inputs are mapped as a weighted sum of Gaussian components. The assumption for a GMM-based clustering approach is that each point is clustered to all of the Gaussian components. For a GMM with K components, each component $1 \leq i \leq K$ is associated with a mean μ_i, and a variance σ_i. All in all, the model is defined as:

$$p(x) = \sum_{i=1}^{K} \phi_i N(x|\mu, \sigma_i) \tag{4}$$

$$N(x|\mu, \sigma_i) = \frac{1}{\sigma_i \sqrt{2\pi}} exp(-\frac{(x - \mu_i)^2}{2}) \tag{5}$$

$$\sum_{i=1}^{K} \phi_i = 1 \tag{6}$$

For this experimentation, each data point is assigned to the Gaussian distribution demonstrating the highest probability. Note that a point can be assigned to any cluster with some probability; however, when a clear separation of clusters exists, most of the probabilities will be negligible.

4.2.3. Spectral clustering. A spectral clustering algorithm similar to the work presented in [25] is employed. Within this formulation, cluster points are defined by eigenvalues matrices derived from the original data input. Note that this clustering algorithm may take any affinity function, therefore different assumptions can be used over what makes points statistically similar. As discussed before, in this work the customised special affinity function (i.e., Equation 2) is used, due to its ability to adapt on a different affinity, thus handle unstructured inputs.

4.2.4. Agglomerative Hierarchical clustering. For the purpose of this work and due to the unstructured nature of the initial datasets, Agglomerative Hierarchical Clustering (AHC) is used. AHC initially computes a proximity matrix and further considers each individual input data point to compose its own cluster. Under a repetitive procedure, each cluster point joins another cluster while the proximity matrix is simultaneously updated with new cluster distances. The AHC outputs were visualised using

Metric	Number of Pure Clusters	Clustering Purity	Average Silhouette Value	Average Number of Streams in Clean Bins	Cleanly Clustered Streams
Abbreviation	NPC	CP	ASV	NSCB	CCS
Definition	Number of clusters that include only streams from one cluster	Average value of the ratios between the largest class in each cluster to the size of the cluster	Indicates how tight the separation is between the clusters. Each point has a value in range [-1,1] which indicates how close it is to other cluster. This metric is the average value for all the data points	Average number of streams that were clustered into clean clusters (clusters with only one class)	Overall number of streams that were clustered in the clean clusters (clusters with only one class)
Range	$[0, 1]$	$(0, 1]$	$[-1, 1]$	$[0, 1]$	$[0, 10000]$
Preferred Value	1	1	1	1	10000

TABLE 3: Clustering performance evaluation metrics

a dendrogram where cluster outputs are viewed as graph nodes.

In order to identify the closest cluster points two similarity functions are used, namely the L_2 norm $L_2 = \sqrt{\sum_{i=1}^{|x|} |x_i|^2}$ and the $cosine$ approach. L_2 norm quantifies similarity based on the Euclidean distance, while the $cosine$ approach quantifies similarity as the difference between the direction and the BPPs set intersection, as defined in Section 4.1.4.

$$cosine(x, y) = \frac{x \cdot y}{||x||||y||} \tag{7}$$

5. Performance Evaluation

In order to assess the effectiveness of the proposed synergistic embedding and clustering scheme for encrypted Youtube video streams, an extensive set of metrics is utilised, as depicted in Table 3. The evaluation results of each embedding technique using the above clustering algorithms are presented below.

5.1. Clustering Performance

5.1.1. Word2Vec Embedding. The average Silhouette value (ASV) for each tuple (w, e), where w is the window size and e is the embedding size, was extracted using each of the clustering schemes (i.e., K-means, GMM, AHC L2, and AHC cosine).

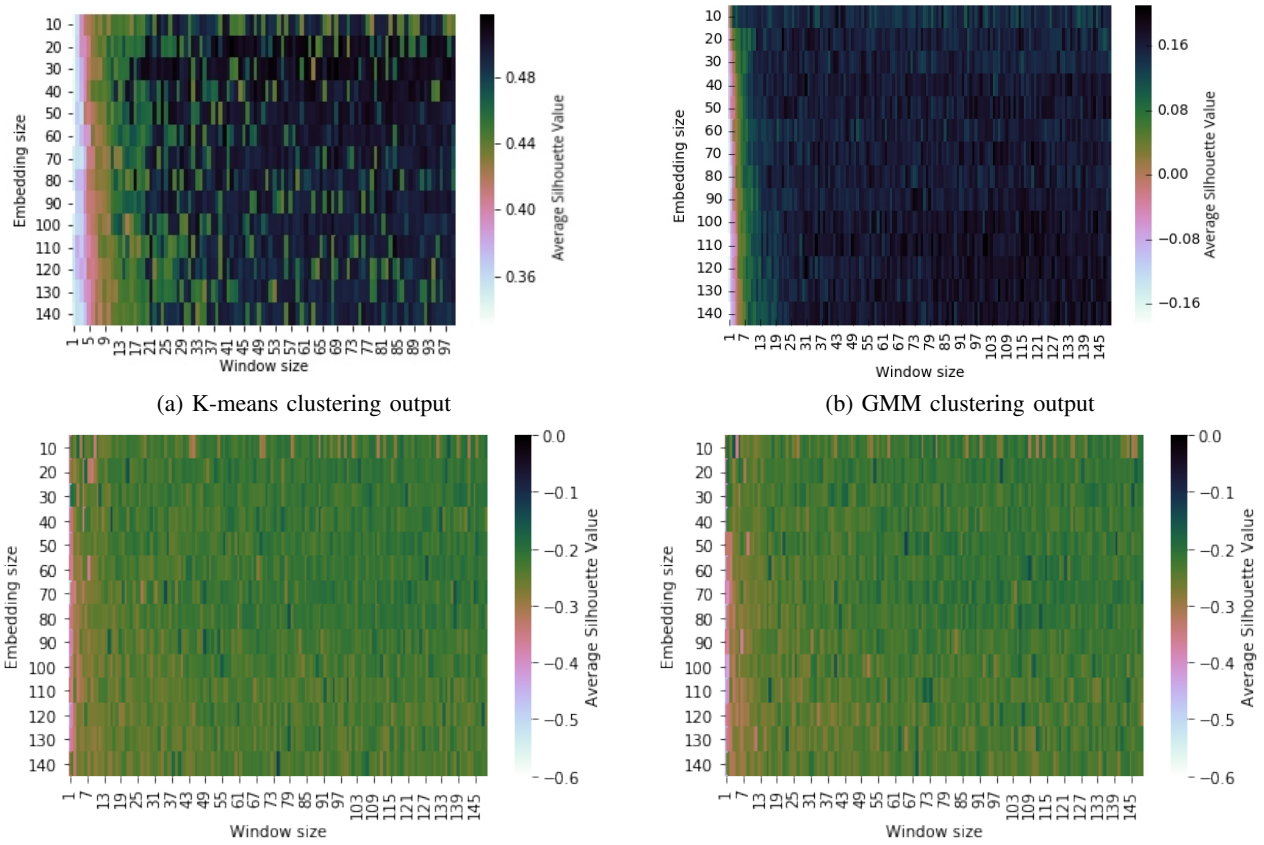
Using a variety of windowing and embedding sizes over the assessed clustering schemes this evaluation produced heat-maps as illustrated in Figures 2a - 2d.

By employing a grid search through the hyper-parameters (optimal window size equal to 82 and embedding size equal to 50) of the Word2Vec results and as demonstrated in the aforementioned figures, it can be seen that the Word2Vec approach tends to map similar samples around the same angle. Hence, the AHC-cosine, K-means and GMM clustering algorithms perform well and quite similarly in terms of the various metrics. Note that both K-means and GMM identify the sample angle due to their statistical nature. However they are also capable of separating samples that are on the same cosine but with different magnitudes. Table 4 summarizes the results obtained and demonstrates that the combination of K-means with embedding outperforms the rest in term of NPC and ASV. Hence, grouping similar patterns in terms of video viewing under k-means is more effective than the rest of the schemes assessed.

5.1.2. Bag of Words Embedding. Due to computational and memory constraints, each bucket in the Bag of Words is assembled to contain 100 consecutive BPP values. By accounting that the average number of peaks per stream is 16 and the vector size is on average $45,643$, the overall process yields a sparse dataset. Therefore, it was crucial to utilise the Principal Component Analysis (PCA) method in order to reduce the obtained high dimensionality while retaining 99.9% of the dataset's variance throughout its statistical meta-features. At this point, data is received with constant size samples of $4,675$. Therefore, it is feasible to optimally apply all the aforementioned clustering techniques as well as the Deep Encoder-based clustering approach.

As evidenced by Table 4, the K-means and the GMM formulations performed well and relatively similar throughout all the clustering performance metrics. The K-means formulation outperformed GMM on the NPC metric reaching 0.93, thus closer to a desired 1 whereas GMM achieved a higher CP metric indicating that each GMM-based cluster has greater purity than any clusters composed by K-means. In terms of the CP metric it can be seen that the AHC achieved the highest performance with 0.99 and 0.98 under the cosine and L2 formulation respectively. However, in contrast to K-means and GMM, both AHC-based formulations have low CCS output of 103 and 100 for cosine and L2 respectively demonstrating that there exists a large number of clusters with a small number of streams. Consequently it can be seen that each AHC-based cluster under this embedding approach would have a minimal set of real traffic streams that can be assigned to corresponding viewing behaviours, thereby, increasing the aspect of complexity on determining distinct viewing characteristics.

5.1.3. Deep Auto Encoder. In contrast to the outputs of Bag of Words and Word2Vec, Table 4 shows that the Deep Encoder-based embedding scheme, that operates strictly with K-means clustering, enforces a very distinct separation of clusters, i.e. the maximal value of streams in clean bins throughout the conducted evaluation. Evidently, the Average Silhouette Value (ASV) value reaches almost a perfect score with 0.99 since it is very close to the ideal values of 1. Nonetheless, it is in parallel shown that



(a) K-means clustering output

(b) GMM clustering output

(c) AHC with L2-based similarity function

(d) AHC with cosine-based similarity function

Figure 2: The effect of the Word2Vec window size and the embedding size over the clustering algorithms in terms of the average Silhouette Value

Embedding and Feature Engineering Technique	Clustering Algorithm	Number of Pure Clusters (NPC)	Clustering Purity (CP)	Average Silhouette Value (ASV)	Average Number of Streams in Clean Bins (NSCB)	Cleanly Clustered Streams (CCS)
Word2Vec	K-Means	0.71	0.57	0.50	0.52	3631
	GMM	0.57	0.72	0.18	0.66	3744
	AHC (cosine)	0.81	0.50	-0.08	0.43	3518
	AHC (L2)	0.88	0.22	-0.18	0.19	1648
Bag of Words	K-Means	0.93	0.44	-0.12	0.4	3706
	GMM	0.88	0.48	-0.08	0.42	3720
	AHC (cosine)	0.99	0.02	0.06	0.01	103
	AHC (L2)	0.98	0.02	0.53	0.01	100
Deep Auto Encoder	K-Means	0.28	0.63	0.99	0.73	2046
BPP's Set Intersection	**Spectral**	**0.84**	**0.74**	Irrelevant	**0.68**	**5724**
	AHC	0.99	0.03	Irrelevant	0.02	222

TABLE 4: Clustering performance evaluation for profiling viewing behaviour of encrypted YouTube video streams

the statistical separation threshold imposed has a negative impact on other metrics. For instance, the number of pure clusters (NPC) reached a low 0.28 and is far below an optimal value that should be closer to 1. Therefore, under this scheme it is feasible to separate groups with high dimensions within their statistical meta-features but it is not guaranteed that each individual data point within each identified cluster represents the overall group. Hence, users with mixed video categories appearing within the same group of a given cluster will potentially be labelled based on the number of the highest data points that are likely to represent users watching a particular category. The reason for these outcomes lies with the tuning of the embedding and the auto encoder, thus an optimisation method that identifies outliers could be of use prior to the employment of the clustering process.

5.1.4. BPPs Set Intersection. The BPP's set intersection is treated here as an affinity function within formulations that parametrically accept such functions. Hence, for this technique there is only provision of results for the spectral and the AHC-based clustering algorithms that do consider affinity functions within their corresponding formulations. By relating the results depicted in Table 4 to the desired metrics discussed in Table 3, it is clear that BPPs produce the optimal outputs through all metrics apart from the NPC. It can therefore be seen that BPP based clustering

methods are capable of producing pure clusters to which viewers are correctly allocated based on their respective video category (i.e., Clustering Purity - CP). It is evident that the composed spectral-based clusters have a high precision range due to the high scores obtained in terms of the number of cleanly clustered streams (i.e. CCS).

Furthermore, AHC-based clustering schemes utilising the BPPs set intersection affinity function do not perform equally well despite the fact that the number of pure clusters through the NPC metric reaches 0.99. Through both schemes it is evident that the average number of streams in each cluster is 0.02. Consequently, the latter result suggests that most of those clean clusters have 2 single users watching the same video and the last unclean cluster has all the remaining users.

5.2. Computational Performance

In order to assess the efficacy of the various algorithms and whether they can be used in practical, real-time scenarios, the study focuses on the computational time required for each to produce a clustering output. Table 5 depicts the resulting computational performance results based on experiments conducted on an Intel I7 machine, 3.5GHz with 11 CPU's and 57G RAM. As illustrated, the majority of embedding and clustering combinations may adequately operate on a close to real-time scenario and provide meaningful profiling of user viewing behaviour. From a pure embedding perspective there is evidence to show that the less intensive clustering outputs are produced when employing the Word2Vec feature engineering and embedding. By contrast, the most computationally intensive scheme is resulted by utilising features from the Deep Auto Encoder approach.

As already discussed earlier, the spectral clustering with the BPPs set intersection outperformed all other models in terms of the CCS metric while maintaining high scores on the NPC metric (i.e. in both close to 1). Nonetheless, despite the high clustering performance for some of the assessed clustering performance metrics, there is a significant computational trade-off in terms of runtime. In fact, spectral-based clustering based on BPPs set intersections requires 24.2 minutes in order to generate any clustering outputs. Hence, such an approach would not be ideal in scenarios where real-time clustering is required. In general, the conducted evaluation suggests that the most optimal solution for addressing real-time grouping of encrypted video streams would be any clustering based on Word2Vec embedding since it takes less than 26 seconds for any algorithm to produce an output. Nonetheless, as discussed in Section 5.1, Word2Vec-based clustering also poses some trade-offs related to the cluster purity (i.e. CP metric) and also the amount of clean traffic streams that are considered to be pure within a given cluster (i.e. NSCB). Therefore, it is recommended to consider such an approach purely on a real-time basis and further compare its outputs with an offline clustering process. Based on the results discussed in Section 5.1, the evaluation identifies clustering based on BPPs set intersection embedding to have the best clustering performance metrics on average, but with a high computational cost in terms of run time.

Embedding	Clustering	Runtime [sec]
Word2Vec	K-Means	25.20
	GMM	25.25
	AHC (cosine)	25.24
	AHC (L2)	25.1
Bag of Words	K-Means	1,461.6
	GMM	3,565.3
	AHC (cosine)	2,864.7
	AHC (L2)	1,913.4
Deep Auto Encoder	K-Means	96,628 $\approx 1.1 days$
BPP's Set Intersection	**Spectral**	**1,451**
	AHC	452.5

TABLE 5: Computational Run Time Performance

6. Conclusion

This work presents an extensive study highlighting a novel, synergistic use of off-the-shelf clustering schemes with NLP and learning-based embedding schemes for profiling encrypted YouTube video traffic streams. In contrast with previous studies, the authors propose clustering with no a-priori assumption on a known subset of titles, and argue in terms of the potential usefulness of such an approach for automated behavioral profiling of malicious actors. The results indicate that it is feasible to obtain extremely high clustering performance under optimal computational costs and may contribute towards forensic investigations required by law enforcement bodies. It can be seen that the combination of an NLP-based embedding scheme with either K-means or GMM-based clustering is an candidate for real-time and scalable grouping of behavioral viewing for end-users over encrypted video streams, and that a combination of learning-based embedding, as well as spectral or hierarchical clustering under BPPs set intersection, can adequately serve the purposes of highly accurate profiling for cases with small datasets.

7. Acknowledgement

This work was supported by the Ariel Cyber Innovation Center in conjunction with the Israel National Cyber directorate in the Prime Minister's Office. In addition, research was supported in part by Security Lancaster, H2020 EC CONCORDIA GA #830927 and the EU H2020 EASY-RES project.

References

[1] United Nations Office on Drugs and Crime. The use of the internet for terrorist purposes. *United Nations*, 2012.

[2] The Washington Post. The Washington Post on militant groups. https://wapo.st/2MnOdgA. Accessed: 2019-03-20.

[3] The Washington Post. The New Zealand shooting shows how YouTube and Facebook spread hate and violent images yet again. https://www.washingtonpost.com/technology/2019/03/15/facebook-youtube-twitter-amplified-video-christchurch-mosque-shooting/?utm_term=.36d4743b83bf. Accessed: 2019-03-15.

[4] R. Dubin, A. Dvir, O. Pele, and O. Hadar. I know what you saw last - encrypted http adaptive video streaming title classification. *IEEE Transactions on Information Forensics and Security*, 12(12):3039–3049, Dec 2017.

[5] Melcher Stikkelorum. I Know What You Watched: Fingerprint Attack on YouTube Video Streams. 2017.

[6] A. Reed and M. Kranch. Identifying https-protected netflix videos in real-time. In *ACM on Conference on Data and Application Security and Privacy*, CODASPY '17, pages 361–368, 2017.

[7] A. Dvir, A. K. Marnerides, R. Dubin, and N. Golan. Clustering the Unknown - The Youtube Case. In *International Conference on Computing, Networking and Communications (ICNC)*, pages 1–6, 2019.

[8] B. Anderson and D. A. McGrew. OS fingerprinting: New techniques and a study of information gain and obfuscation. *CoRR*, abs/1706.08003, 2017.

[9] R. Dubin, A. Dvir, O. Pele, J. Muehlstein, Y. Zion, M. Bahumi, and I. Kirshenboim. Analyzing https encrypted traffic to identify users operating system, browser and application. In *IEEE Consumer Communications and Networking Conference*. IEEE, Jun. 2017.

[10] M. Conti, L. V. Mancini, R. Spolaor, and N. V. Verde. Analyzing android encrypted network traffic to identify user actions. *IEEE Transactions On Information Forensics and Security*, 2016.

[11] M. Husak, M. Čermák, T. Jirsík, and P. Čeleda. HTTPS traffic analysis and client identification using passive SSL/TLS fingerprinting. *EURASIP Journal on Information Security*, 2016(1):1–14, 2016.

[12] Jeffrey Erman, Martin Arlitt, and Anirban Mahanti. Traffic classification using clustering algorithms. In *SIGCOMM Workshop on Mining Network Data*, MineNet '06, pages 281–286, 2006.

[13] Carlos Bacquet, A. Nur Zincir-Heywood, and Malcolm I. Heywood. Genetic optimization and hierarchical clustering applied to encrypted traffic identification. In *2011 IEEE Symposium on Computational Intelligence in Cyber Security, CICS 2011, Paris, France, April 12-13, 2011*, pages 194–201, 2011.

[14] J. Hochst, L. Baumgartner, M. Hollick, and B. Freisleben. Unsupervised traffic flow classification using a neural autoencoder. In *Conference on Local Computer Networks (LCN)*, pages 523–526, Oct. 2018.

[15] A. Finamore, M. Mellia, M. M. Munafò, R. Torres, and S. G. Rao. YouTube Everywhere: impact of Device and Infrastructure Synergies on User Experience. In *Internet Measurement Conference (IMC)*, pages 345–360, 2011.

[16] R. Dubin, O. Hadar, A. Dvir, and O. Pele. Video quality representation classification of encrypted http adaptive video. *KSII Transactions on Internet and Information Systems*, 12(8):3804–3819, 2018.

[17] R. Schuster, V. Shmatikov, and E. Tromer. Beauty and the Burst: Remote Identification of Encrypted Video Streams. In *26th USENIX Security Symposium (USENIX Security 17)*, pages 1357–1374. USENIX Association, 2017.

[18] Ronan Collobert and Jason Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *International Conference on Machine Learning*, pages 160–167. ACM, 2008.

[19] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781, 2013.

[20] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc., 2013.

[21] Yoav Goldberg. A primer on neural network models for natural language processing. *J. Artif. Intell. Res.(JAIR)*, 57:345–420, 2016.

[22] Peter J Rousseeuw. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, 20:53–65, 1987.

[23] Raymond Kosala and Hendrik Blockeel. Web mining research: A survey. *ACM Sigkdd Explorations Newsletter*, 2(1):1–15, 2000.

[24] Junyuan Xie, Ross Girshick, and Ali Farhadi. Unsupervised deep embedding for clustering analysis. In *International Conference on Machine Learning*, pages 478–487, 2016.

[25] Andrew Y Ng, Michael I Jordan, and Yair Weiss. On spectral clustering: Analysis and an algorithm. In *Advances in Neural Information Processing Systems*, pages 849–856, 2002.