Received 21 June 2022, accepted 15 July 2022, date of publication 19 July 2022, date of current version 26 July 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3192458

RESEARCH ARTICLE

Fingerprinting Technique for YouTube Videos Identification in Network Traffic

WALEED AFANDI[1], SYED MUHAMMAD AMMAR HASSAN BUKHARI[1], MUHAMMAD U. S. KHAN[1], (Member, IEEE), TAHIR MAQSOOD[1], AND SAMEE U. KHAN[2], (Senior Member, IEEE)

[1]Department of Computer Science, COMSATS University Islamabad, Abbottabad Campus, Abbottabad 22060, Pakistan
[2]Department of Electrical and Computer Engineering, Mississippi State University, Starkville, MS 39762, USA

Corresponding author: Muhammad U. S. Khan (ushahid@cuiatd.edu.pk)

This work was supported in part by the National Centre of Cyber Security (NCCS), Pakistan; and in part by the Higher Education Commission (HEC) under Grant RF-NCCS-023.

ABSTRACT Recently, many video streaming services, such as YouTube, Twitch, and Facebook, have contributed to video streaming traffic, leading to the possibility of streaming unwanted and inappropriate content to minors or individuals at workplaces. Therefore, monitoring such content is necessary. Although the video traffic is encrypted, several studies have proposed techniques using traffic data to decipher users' activity on the web. Dynamic Adaptive Streaming over HTTP (DASH) uses Variable Bit-Rate (VBR) - the most widely adopted video streaming technology, to ensure smooth streaming. VBR causes inconsistencies in video identification in most research. This research proposes a fingerprinting method to accommodate for VBR inconsistencies. First, bytes per second (BPS) are extracted from the YouTube video stream. Bytes per Period (BPP) are generated from the BPS, and then fingerprints are generated from these BPPs. Furthermore, a Convolutional Neural Network (CNN) is optimized through experiments. The resulting CNN is used to detect YouTube streams over VPN, Non-VPN, and a combination of both VPN and Non-VPN network traffic.

INDEX TERMS Video identification, fingerprinting, deep learning, classification, variable bitrate.

I. INTRODUCTION

With the advancement of technology and availability of mobile devices, the past few years have seen an increase in video network traffic. CISCO claims video streaming to be the leading consumed media that has become the major contributing factor to internet traffic [1]. For the security and privacy of clients, internet traffic is encrypted, leaving little or no possibility of monitoring stream content. Minors and adolescents can be induced to inappropriate content with unmonitored traffic [2], [3]. Most video streaming platforms, such as YouTube, Facebook, and Twitch, have adopted dynamic adaptive streaming over HTTP (DASH) technology to enhance the client's quality of experience (QoE). DASH uses the Variable Bitrate (VBR) encoding technique to stream video content to clients to ensure a smooth streaming experience [4]. The popularity of DASH resulted in multiple industries starting to invest in this direction. Furthermore, the Google search engine also ranks video streaming websites on the first page of search results that adopt DASH streaming technology [5].

The previous video identification frameworks rely on IP packet headers, ports, and content information to identify individual videos. However, with the rising popularity of such frameworks and their threats to user privacy and security, video streaming service providers started encrypting their streams to mitigate security issues. Most of the traffic flowing between clients and servers is secured by Secure Socket Layer (SSL) and Transport layer security (TLS) encryption technology over HTTPS protocol. In conclusion, such encryption approaches restrict techniques including Deep Packet Inspection (DPI) [6] to identify individual videos streaming over a network. Furthermore, with the upsurge in the availability of free Virtual Private Networks (VPNs),

The associate editor coordinating the review of this manuscript and approving it for publication was Shihong Ding.

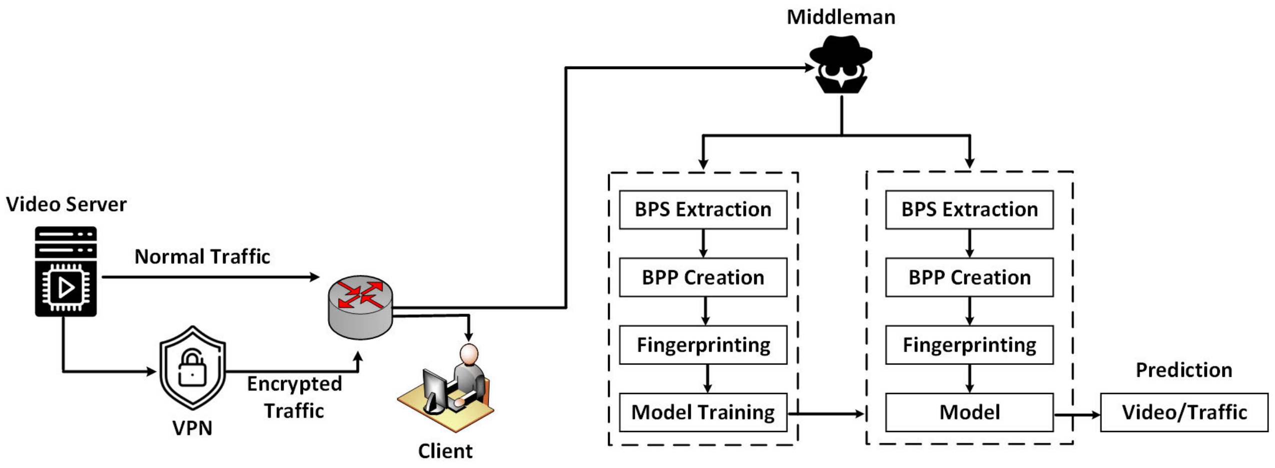

FIGURE 1. Experimental platform of video identification pipeline.

more clients and adversaries are inclined to use them to hide their network activities, which aggravates the streaming video identification.

VPN and SSL protocols prevent a middleman from viewing the content of network traffic between the two communicating parties. Although an SSL protocol only encrypts the packet, the whole packet is encapsulated within another packet in a VPN, which allows the client to bypass server blockage at the gateway by tunneling through a remote machine. A VPN is classified into two types based on its security protocol; SSL VPN and IPsec VPN. The VPN used in this research is OpenVPN [7] which is an SSL type VPN that uses a Hash-based Message Authentication Codes (HMAC) with a SHA1 hashing algorithm to ensure that the contents in the packets are intact. However, plentiful information regarding streaming video can be obtained via flow-based features, including the number of packets, packet sizes, burst sizes, and quantity of packet bursts. Machine learning and deep learning models [8], [9] can leverage these features to identify streaming videos.

Over the years, deep learning-based neural networks have outperformed traditional machine learning algorithms. A Convolutional Neural Network (CNN) is a particular type of artificial neural network that applies convolutional operation in at least one of its layers. Its high accuracy and efficiency have introduced many real-world applications, including human activity detection [10], natural language processing [2], and bot detection [11]., energy consumption prediction [12], smart city policing [13], risk assessment [14], and text simplification [15].

DASH streaming follows a specific pattern to send the videos to the client. In DASH, each video is divided into small segments, sometimes called chunks, and delivered to clients. This technique is used to increase the Quality of Experience (QoE) of the client. However, these segments are delivered to the client's device in a specific method, leaving a delivery pattern in the network traffic. This

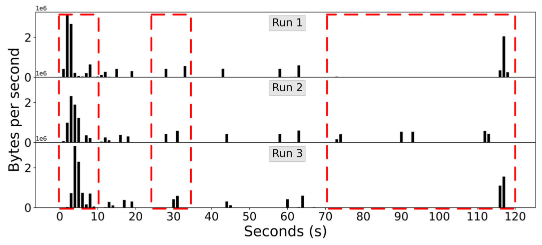

FIGURE 2. Inconsistent bytes arrival time in three separate runs.

pattern can be used to identify the video in the network traffic [16], [17]. Many studies have exploited the DASH streaming pattern to identify the videos in the network traffic [18]–[21]. However, VBR encoding produces inconsistency in the streaming pattern as shown in Figure 2. These abnormalities sometimes create difficulties for the researchers to identify the video. Moreover, the client's variable network conditions also complicate the video identification process.

To address the aforementioned challenges, i.e., (a) video identification in the variable environment and (b) handle the abnormalities and inconsistencies cropped up by the DASH streaming technology, a fingerprinting method, Simple Difference Fingerprint (SDF), is proposed. This method is used to generate a stable fingerprint of a video. For this purpose, the Bytes per second (BPS) of the video stream are extracted, aggregated into periods, and used for video identification as shown in Figure 1. The details of the creation of periods and fingerprints are provided in Section III.

The proposed SDFs are used to train a convolution neural network (CNN) to classify the VBR video. Initially, the CNN is fine-tuned through rigorous experiments to deduce the perfect hyperparameters for different layers, including convolutional, pooling, dropout, and dense layers, alongside other model hyperparameters such as batch size, optimizer, and the number of epochs. After tuning, the optimized model is used for video classification with different traffic combinations of Virtual Private Network (VPN) and Non-VPN network traffic. The main contributions of this study are the answers to the following research questions:

- Can a Sequential Convolutional Neural Network (SCNN) be used to identify the video in the network traffic?
- How does SDF fingerprinting technique fare with other fingerprinting methods, and which technique is the most effective?

The rest of the paper is organized as follows. Section II presents a summary of previous works and Section III presents the method for fingerprint creation to handle the inconsistencies in the network traffic. Section IV presents the experimental setup and hyperparameter tuning of CNN. Section V presents the comparison of different fingerprinting methods and finally Section VI concludes the paper.

II. RELATED WORK

Hypertext Transfer Protocol Secure (HTTPS) started gaining attention as Google was one of the earliest adopters of HTTPS. In contrast to its predecessor, HTTP, it provides a secure environment and captures the interest of many researchers to find vulnerabilities in this protocol. Some research has been conducted to identify video streaming on a client computer.

Chen *et al.* [22] demonstrated the severity of side-channel attacks even with modern encryption techniques. Certainly some hardware-level attacks are possible, as shown in [23]. Several works have been done for attacking network traffic of Skype to identify user actions [24], [25]. Furthermore, it is possible to identify the website that is being viewed on the network [26]–[28]. Moreover, user activities can be revealed by the network traffic [10], [29]–[31]. Private information can also be leaked in location-based applications [32]–[35]. WiFi signals can be sniffed [29], and routers can be hacked to sniff packets if the adversary is present inside LAN [36]. At first, video identification researchers leveraged QoE metrics to optimize network bandwidth sharing. Mangla *et al.* [37] predict these QoE metrics of video streams by weighing packet headers in network traffic.

In contrast, [38] uses a set of statistical features that include the quantity and size of the packets to classify the resolution and bitrate of the video streams. Statistical features are also used by [39] to identify the flow of video in the network. Gutterman *et al.* [40] predict quality metrics for YouTube encrypted videos by exploiting chunk statistics, including chunk length and chunk duration, as well as flow statistics such as flow duration and direction. Chunk statistics are also leveraged by [41] to identify variable bitrate adaption under HTTP and QUIC protocol.

Ameigeiras *et al.* [42] described a characteristic burst feature in the YouTube network streams. These bursts are of two types: a long burst and a short burst. At the beginning of streaming, there is a long burst, after which the video is buffered in smaller bursts. These characteristics are also discussed by Rawattu and Balasetty [43]. The On-Off period between each burst is discussed by Rao *et al.* [44], and Liu *et al.* [45] leverages these On-Off periods to identify the streaming video using traditional machine learning approaches.

BPS is an important feature that plays an essential role in video identification. Khan *et al.* [21], [46] extracted the BPS of a video stream multiple times in different video qualities and used them as a feature. This feature is used to train different machine learning models, including Naïve Bayes, SVMs (Support Vector Machines), and CNN. However, extracting the BPS and using it as raw data to identify the video in network traffic is not enough to deal with the irregularities in video identification caused by the VBR.

To address the irregularities of VBR, the study most related to our work [20] discusses the method of differential fingerprints. The authors propose an algorithm to aggregate BPS into several periods. This approach reduces the inconsistency that occurs due to VBR. However, they only used the feature distance measuring technique to predict the queried video. Furthermore, the differential fingerprinting method presented by the authors ignores the condition of dividing by zero, and the dataset is missing videos streamed over a VPN. Furthermore, the dataset consists of only Facebook videos that stream over 180 seconds on Non-VPN traffic. Based on the limitations of previous studies mentioned above, this paper aims to modify the algorithm proposed in [20] to handle the cases of zero as the denominator. The dataset in this research contains both VPN and Non-VPN streamed videos, and the video stream length is 120 seconds. Furthermore, the baseline convolutional neural network presented in [21] is fine-tuned, and hyperparameters are changed to improve the accuracy of the results. The accuracy of the baseline model on our dataset is 54.77%.

III. METHODOLOGY

This section illustrates the methodology used for data collection, fingerprinting methodologies, and producing a list of predictions through various classifiers as shown in Figure 3. The methodology is defined in steps as (a) data collection, (b) preprocessing of data, (c) bytes per period, (d) fingerprinting, and (e) summary of neural network.

A. DATA COLLECTION

We use Wireshark to capture the network traffic and generate packet capture (PCAP) files against each video to generate the dataset of video streams. We utilize the Chrome browser to play the YouTube videos and Selenium for automation. We selected 43 random videos from YouTube and each video is downloaded 55 times. A desktop client SurfShark is used for capturing the VPN streams. In conclusion, the resultant data set consists of 86 total labels - 43 non-VPN titles and 43 VPN titles and each video stream is captured for 120 seconds.

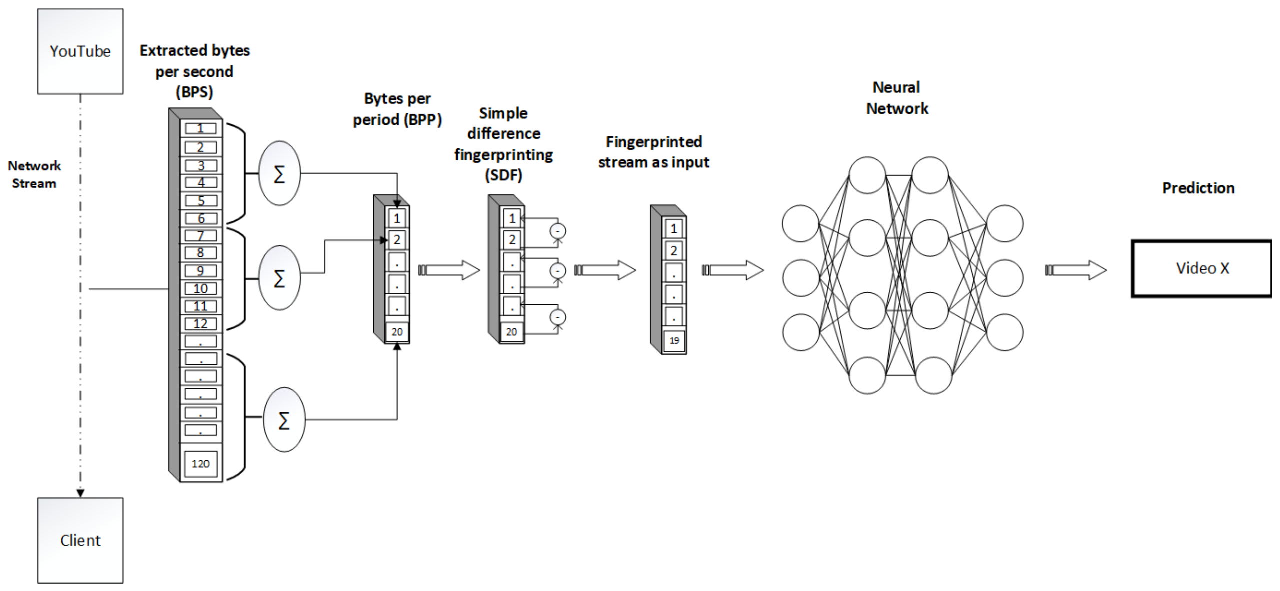
YouTube
Extracted bytes per second (BPS)
Network Stream
Client
Bytes per period (BPP)
Simple difference fingerprinting (SDF)
Fingerprinted stream as input
Neural Network
Prediction
Video X
120
20
19

FIGURE 3. Architectural design of video identification pipeline.

TABLE 1. List of acronyms used in the paper.

Acronym	Explanation
ADF	Absolute difference fingerprint
BPP	Bytes per period
BPS	Bytes per second
CISCO	Commercial & Industrial Security Corporation
CNN	Convolutional neural network
CPU	Central Processing Unit
DASH	Dynamic adaptive streaming over HTTP
DF	Differential fingerprint
DPI	Deep packet inspection
GB	Gigabyte
GPU	Graphics Processing Unit
GTX	Giga Texel Shader
HMAC	Hash-based Message Authentication Codes
HTTP	Hypertext transfer protocol
HTTPS	Hypertext transfer protocol secure
IP	Internet protocol
LAN	Local area network
PCAP	Packet capture
QoE	Quality of Experience
QUIC	Quick User Datagram Protocol Internet Connections
RAM	Random access memory
ReLU	Rectified Linear Unit
SCNN	Sequential Convolutioanal Neural Network
SDF	Simple difference fingerprint
SHA	Secure hash algorithm
SSL	Secure socket layer
SVM	Support vector machines
TLS	Transport layer security
VBR	Variable bitrate
VPN	Virtual private network

B. PRE-PROCESSING

Wireshark exports the captured data in pcap file format. Each generated pcap file contains 120 seconds of the streaming video. As this file contains both uplink and downlink traffic, this dataset is cleaned by applying a filter through the IP address of the server, which in this case is YouTube, and selecting only the downlinks. This process is done for VPN and Non-VPN data. Thus, it mitigates the streaming noise of unwanted applications. This is achieved by the integrated Wireshark filter in the conversation section.

C. BYTES PER PERIOD (BPP)

As mentioned above, DASH uses the VBR encoding, which can cause irregularities in the sequence of BPS, effectively reducing the accuracy of machine learning models. For this reason, the BPS are aggregated into L segments, where L is any constant number. In this paper, the value of L is set to 6 as discussed in [20]. This aggregation compresses the number of features from 120 BPS to 20 BPP. This approach eliminates the VBR inconsistencies as the total size of a given period will remain virtually the same irrespective of the sequence in which the bytes are received. Therefore, the difference between two consecutive periods will remain consistent, discounting the factor of what sequence the bytes are received in a given period.

D. FINGERPRINTING

Fingerprinting is a process of representing a large data by a small bit of string, that uniquely identifies the data in a process. Particularly, fingerprints are the small labels for large data [47]. Due to the effectiveness of fingerprinting, many researchers have effectively utilized the fingerprinting technique in different scenarios. For instance, fingerprinting is actively used in application discrimination [48], video identification [20], Web page recognition [49], user activity monitoring [25], and mobile application identification [50].

In our paper, we utilize the fingerprinting technique for video identification in the encrypted network traffic. For this purpose, we created fingerprints of the BPPs of a video stream. The created fingerprints help to differentiate individual video streams. After generating a BPP sequence of a stream, all the 0 in the sequence are replaced with 1 to resolve the zero division problem encountered during fingerprints creation. For a video consisting of n seconds, we get a sequence denoted as $a = (a_1, a_2, \ldots, a_i, \ldots)$. For two adjacent data amounts a_{i-1} and a_i, fingerprints can be generated as $r = (r_1, r_2, \ldots, r_i, \ldots)$ by applying the one of the following equations described below:

1) SIMPLE DIFFERENCE FINGERPRINT (SDF)

Video fingerprint r_i can be calculated by subtracting the i^{th} term of sequence a with the previous term $(i-1)$ as shown in Equation (1):

$$r_i = a_i - a_{i-1} \tag{1}$$

2) ABSOLUTE DIFFERENCE FINGERPRINT (ADF)

This is a modified form of Equation (1) proposed in [20]. To eliminate the negative values generated by subtracting a_{i-1} from a_i, we take the absolute of the difference as shown in Equation 2.

$$r_i = |a_i - a_{i-1}| \tag{2}$$

3) DIFFERENTIAL FINGERPRINT (DF)

Equation (3) is proposed by [20]. In this equation, the differential of two consecutive periods is calculated as shown below:

$$r_i = \frac{a_i - a_{i-1}}{a_{i-1}} \tag{3}$$

E. CONVOLUTIONAL NEURAL NETWORK (CNN) MODEL

A convolutional neural network (CNN) is a variant of the traditional neural network because it can learn directly from data without manual feature extraction. A CNN generally consists of convolution layers, pooling layers, and a fully connected layer. They are mainly used in pattern recognition and their architecture makes them a preferred model for object detection in image, voice in audio, natural language processing (hate speech detection [2]), activity recognition (bot detection [11], human activity recognition [10], malware detection [3]), and classify digital signals. The CNN model designed in this paper comprises four 1D convolutional layers, each having *ReLU* as its activation function. Each convolutional layer employs distinct kernels (also called filters) that independently convolve the input data and produce a feature map as the output. The kernel size is assigned a small number relative to the input size. The smaller kernel size helps the model learn more feature maps and improve the overall prediction accuracy. The generated feature map is passed through an activation function (*ReLU* in our case) and passes to the pooling layers.

The max-pooling layers separate the four convolutional layers. A pooling layer summarizes the result of the previous layer to its neighboring outputs, ultimately reducing the size of the data without losing the key features. This reduction of data generalizes the repeating patterns while simultaneously reducing memory requirements.

After convolutional and max-pooling layers, a dropout layer is added. The dropout layer randomly disables some of the inputs of the previous layer to prevent the model from learning only a few input values and restrain overfitting. The relative amount of input features are disabled by defining the probability value p in the dropout layer.

The dropout layer's output is passed to the flatten layer, which performs conversion of the multidimensional pooled feature map into a one-dimensional vector to make it compatible with forwarding into the dense layer. The dense layer is a fully connected layer having all of its neurons connected with the neurons of the previous layer. The output of the dense layer is passed to a final dense layer, also called Output layer [51].

The fingerprint of BPP is a one-dimensional array; therefore, the input of the proposed CNN model is a one-dimension series of BPP with the size of 20. The first convolutional layer has a kernel size equal to 5 with 300 filters and a stride of 1. A single neuron is connected to a cluster of five features of the input data. The output of the layer is 300 feature maps of size 19. Subsequently, the first convolutional layer contains 1800 trainable parameters (1500 weights and 300 bias parameters.) This layer is followed by a max-pooling layer that consists of a 300 feature map of size 50. Each feature map of this layer is connected to two feature maps of the previous convolutional layer. The max-pooling layer has no trainable parameters.

The second convolutional layer has 512 kernels, each of size 3, forming a total number of 461,312 parameters that yield 512 feature maps of size 3. The trailing max-pooling layer after the second convolutional layer generates 512 feature maps of size 11. The third convolutional layer containing 524,800 trainable parameters has 512 kernels of size 1, which output 512 feature maps of size 1. Its successive max-pooling layer generates 512 feature maps of size 1. The last convolutional layer has 300 kernels of size 1, having 307,500 trainable parameters.

Consequently, the last max-pooling layer produces 300 feature maps of size 1. After the last pooling layer, a dropout layer is added to disable arbitrary neurons from the previous layer with the probability of 0.8. The dropout layer is followed by a flatten layer that converts the pooled feature maps of the previous layer into a one-dimensional feature size of 3,300. The output layer, the last layer in the model, contains 141,743 trainable parameters for 43 labels in the dataset.

The activation function selected for all the convolutional layers in this model is the *ReLU* function. The softmax function is assigned as the activation function for the output layer. The *ReLU* function is quite simple as it outputs the input directly if it is a positive number. However, it outputs a zero in the case of a negative number. The softmax function

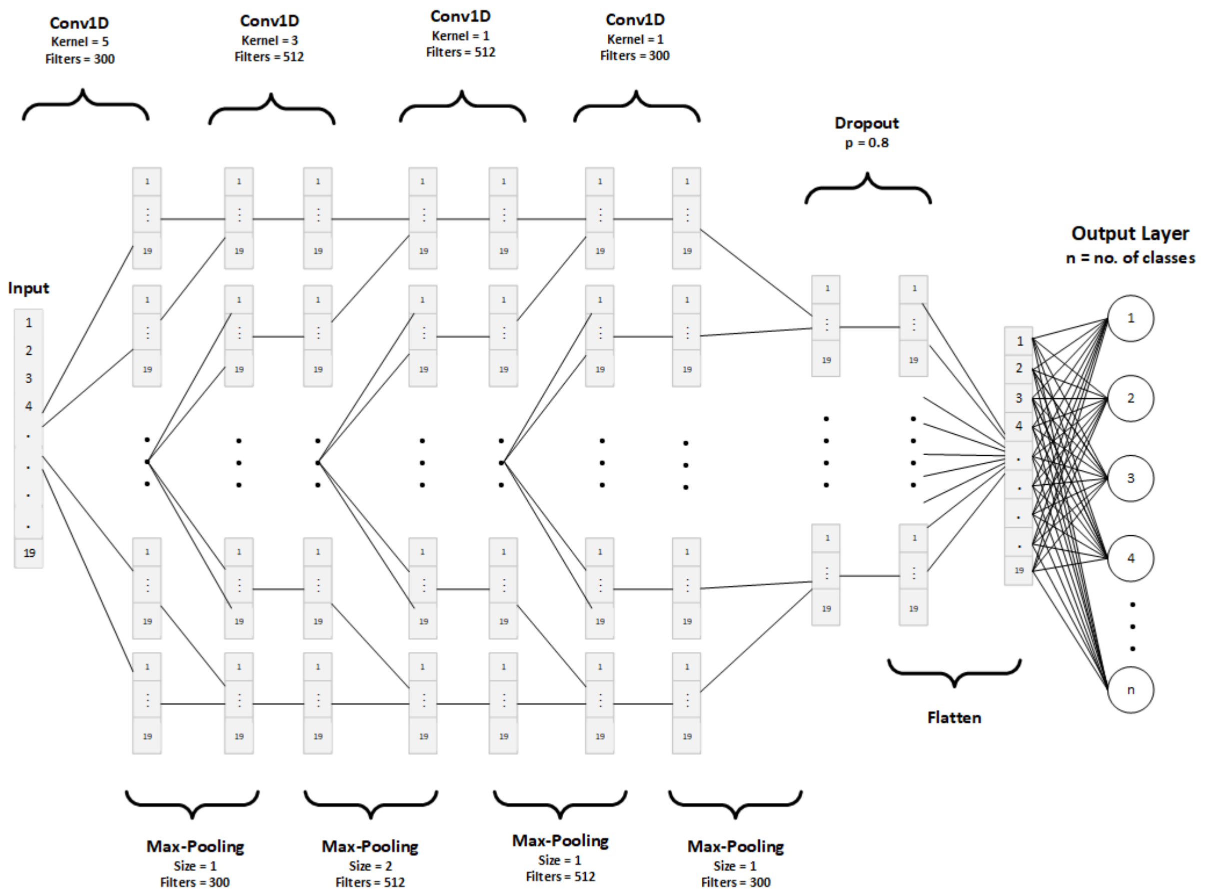

FIGURE 4. Arrangement of layers of convolutional neural network.

is a generalization of logistic regression to handle multiple classes. For N output classes, it normalizes an N-dimensional vector of actual values to an N-dimensional probability distribution vector of actual values in the range [0,1]. The N-dimensional output vector is the probabilistic score of each corresponding class.

The cost function measures the difference between the model's prediction with the actual output and returns an error value. This error rate helps the model determine how much more optimization is needed. The cost function selected for the proposed model is categorical cross-entropy. The optimization function, which assigns optimal weights to neurons in each layer, is the Adam optimizer. The complete architecture of the CNN model is shown in Figure 4

The softmax activation function is used for the dense layer and adam optimizer is used for model optimization. The model is trained on three types of datasets: SDF, ADF, and DF. The model summary is presented in the Table 2.

IV. EXPERIMENTAL SETUP AND MODEL FINE TUNING

The experiments performed in this paper are heavily based on a Graphics Processing Unit (GPU). Therefore, all the experiments are conducted on an Intel Core i7 processor @ 3.4GHz with 16GB RAM and Nvidia GeForce GTX 1060 with 6GB GPU memory. The experiment setup includes changing various hyperparameter values, including the number of filters, kernel sizes, pool size, adding another layer, dropout ratio, batch size, and the number of epochs. Table 3 illustrates the summary of each experiment.

A series of experiments are performed on each convolutional layer of the baseline model presented in [21]. In each experiment, several hyperparameters of the respective layer are changed. In the first experiment, we change the number

TABLE 2. Fine tuned CNN model summary.

Hyperparameter	Value
No. of Conv1D layers	(1,2,3,4)
No. of filters	(300, 512, 512, 300)
Kernel size	(5,3,1,1)
Activation function (all layers)	relu
Max-pooling – pool size	(1, 2, 1, 1)
Padding (all layers)	same
Dropout	0.8
Dense – neurons (output)	number of classes
Loss function	Categorical crossentropy
Optimizer	Adam
Batch size	50
Epochs	300

TABLE 3. Experiments summary.

Experiment number	Description	Accuracy
Experiment #1	Tuning the first convolutional layer	55.93%
Experiment #2	Tuning the second convolutional layer	56.16%
Experiment #3	Tuning the third convolutional layer	56.16%
Experiment #4	Tuning the dropout layer	61.86%
Experiment #5	Tuning pooling layers	65.58%
Experiment #6	Adding and tuning fourth convolutional layer	66.40%
Experiment #7	Adjusting the number of epochs	67.91%
Experiment #8	Setting the batch size	68.14%
Experiment #9	Changing Max Pool to Average Pool	68.14%

of filters of the first layer, and the rest of the values of hyperparameters remain unchanged. Once we get higher accuracy than the baseline model, we fix that value of the number of filters and change the value of kernel size. Again, on getting a higher accuracy, the value of kernel size is fixed for the next experiment. The same procedure is repeated for the second and third convolutional layers. After fixing the number of filters and kernel size of the convolutional layers, we repeat the same procedure for selecting the best dropout value, pool size of all layers, number of epochs, and batch size. We also add a fourth convolutional layer, and lastly, we change the max-pooling to average pooling to check its impact on accuracy. In this manner, we obtain a model with the most fine-tuned hyperparameters. These experiments are performed on VPN vs. Non-VPN with SDF applied dataset. The list of experiments is as follows. Each experiment is conducted to answer the following questions:

- What is the objective of the experiment?
- What are the outcomes of the experiment?
- What is the impact of the experiment on the results?

A. EXPERIMENT #1 TUNING 1st CONVOLUTIONAL LAYER

This experiment aims to find the optimal settings for the number of filters and kernel size of the first convolutional layer of the baseline model. We start the experiment by setting the number of filters to 100 and increasing it by 100. The highest accuracy achieved during this experiment is 55.70% when the filters are equal to 300. After fixing the number of filters to 300, we start increasing the kernel size. However, the accuracy is decreased by 4%. Therefore, we decrease the kernel size to 5, 4, and 3. After kernel size 5, the accuracy starts to decrease. In the experiment, we get the optimal value for the kernel and the number of filters of the first convolutional layer with a 2.11% accuracy increase. The comparison of accuracy with different settings applied in this experiment is shown in Figure 5.

B. EXPERIMENT #2 TUNING 2nd CONVOLUTIONAL LAYER

After deducing the values of the first layer, this experiment is performed to tune the values of the second layer. The same procedure is followed as in Experiment #1. In this experiment, we use various filters and kernel sizes. However, in the case of filters, the settings of the baseline model provide higher accuracy; increasing or decreasing the number of filters results in a decrease in accuracy. On the contrary,

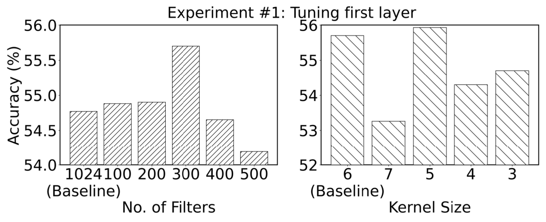

FIGURE 5. Accuracy comparison with various settings applied to first convolutional layer.

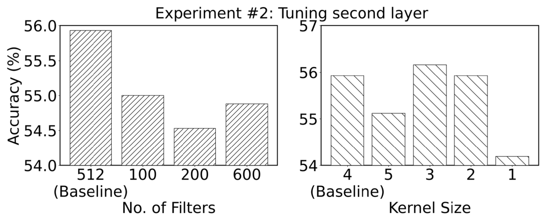

FIGURE 6. Accuracy comparison with various settings applied to second convolutional layer.

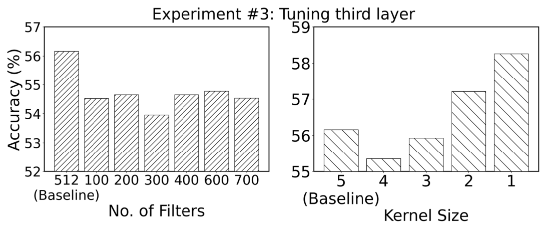

FIGURE 7. Accuracy comparison with various settings applied to third convolutional layer.

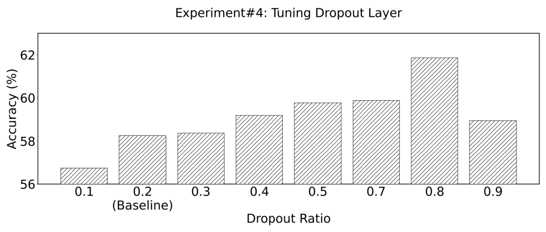

FIGURE 8. Accuracy comparison of various settings applied to dropout.

decreasing the kernel size shows an increase in accuracy. Experiments show that reducing the kernel size to 3 increases the accuracy to 56.16%. Figure 6 shows the accuracy comparison with various settings applied to the second convolutional layer.

C. EXPERIMENT #3 TUNING 3rd CONVOLUTIONAL LAYER

In the continuation of Experiment #1 and Experiment #2, this experiment is performed to fine-tune the third convolutional layer of the CNN. The experiment highlights that the value 512 of filters is the most suitable for this layer. Changing this value decreases the accuracy. The size of the kernel has a positive impact on the accuracy of the model. Reducing the kernel size to the minimum, that is, 1, increases the

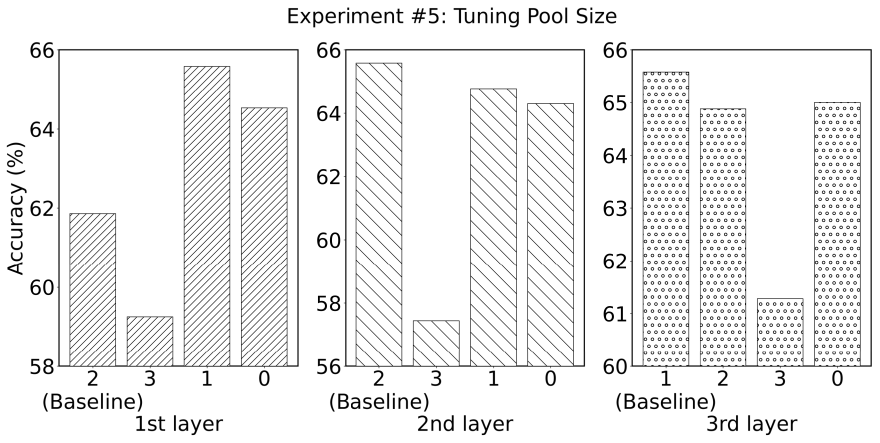

FIGURE 9. **Accuracy comparison with different pool size values.**

accuracy to 6% from the baseline model. Figure 7 shows the comparison of the accuracy of various settings applied to the third convolutional layer.

D. EXPERIMENT #4 TUNING DROPOUT LAYER

The dropout layer is used to mitigate overfitting in the model. This is achieved by randomly neutralizing neurons from the previous layer. Conventionally, the dropout ratio is recommended to be a small value. However, for this model, we set different values for dropout ranging from 0.1 to 0.9. The highest accuracy achieved in this experiment is 62% when setting the dropout value at 0.8. Figure 8 shows the accuracy comparison with various dropout values.

E. EXPERIMENT #5 TUNING POOLING LAYERS

The pooling layers play an important role in reducing the dimensions of the feature map, effectively reducing the number of parameters to learn. The baseline model consists of 3 pooling layers. The tuning of each pooling layer is done in the same manner as convolutional layers. We find the optimal setting for each pooling layer one by one. Changing the value of the pool size of the first pooling layer results in an increase in accuracy. However, changing the pool size of the second and third pooling layer decreases the accuracy. Therefore, in this experiment, we fix the pool size of the first layer to 1 and leave the size of the second and third layers to their default value, as mentioned in the baseline model. Figure 9 shows the summary of the experiment.

F. EXPERIMENT #6 ADDING A 4th CONVOLUTIONAL LAYER AND TUNING IT

After tuning the first three convolutional layers, this experiment is performed to check the impact of adding a new convolutional layer on accuracy. The new layer is tuned in the same manner as the previous three layers. To find the suitable number of filters for the fourth layer, we initially set the filters to 100 and then increased the size by 100. The experiments

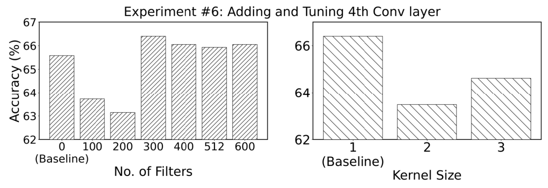

FIGURE 10. **Accuracy comparison with various hyperparameter settings on fourth layer.**

show that the highest accuracy is achieved when the number of filters equals 300. Similarly, we observe that the minimum kernel size produces the best results for the kernel size. After this experiment, the accuracy is improved from 65.58% to 66.40% with four convolutional layers, as shown in Figure 10.

G. EXPERIMENT #7 ADJUSTING THE NUMBER OF EPOCH

The epoch number is a hyperparameter used to define the number of times a model trains itself on the given dataset. An epoch is a single pass over the whole training set to the neural network. In general, increasing the number of epochs increases the accuracy with the trade-off of time taken to train the model. Therefore, considering the factors mentioned earlier, an acceptable value for the number of epochs is selected. In all the previous experiments, the training is done on 100 epochs. In this experiment, we check the impact of the number of epochs on accuracy. For this purpose, we increase the number of epochs by 50 in each experiment. However, maximum accuracy is obtained when the number of epochs is set to 300, as shown in Figure 11.

H. EXPERIMENT #8 SETTING BATCH SIZE

The number of training samples passed to the neural network at one time is called batch size. Increasing the batch size increases the GPU memory requirements for training the model. Therefore, the batch size should be set according to the resources at disposal. The batch size in the baseline

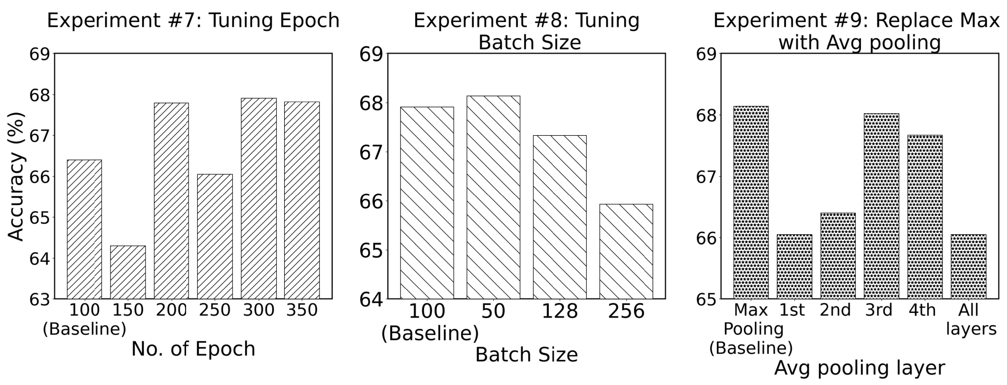

FIGURE 11. Accuracy comparison of various settings applied to third convolutional layer.

model is set to 100. However, decreasing the batch size to 50 increased the accuracy to 68.14% as shown in Figure 11.

I. EXPERIMENT #9 - CHANGING MAX POOL TO AVERAGE POOL

Max-pooling is a technique used to detect the most significant values on the feature map. Similarly, the average pooling technique is used to calculate average values on the feature map. In this experiment, we change each pooling layer from max to average individually to check its impact on accuracy. For example, we change the first max-pooling layer to average pooling and check the accuracy. After that, we change the first pooling layer to the max and replace the second pooling layer to average, and so on. We also replace all the max-pooling layers with average pooling layers in the model. However, the results indicate that no increase in accuracy is achieved by changing the pooling type, as shown in Figure 11.

V. COMPARISON OF DIFFERENT FINGERPRINTING TECHNIQUES

After fine-tuning the model, the same model is applied to different datasets to check the accuracy. This section compares the accuracy with the fingerprint techniques, i.e., SDF, ADF, and DF. For this purpose, four types of datasets are generated by following the method mentioned in Section III. The first dataset is prepared by capturing the 43 video streams in normal traffic mode (Non-VPN mode). Similarly, the same videos are captured using an encryption technique (VPN) for the second dataset. We combine the first two datasets for the third dataset and obtain a dataset containing 86 videos, 43 in normal traffic mode and 43 in encrypted mode. For the fourth dataset, we label all videos captured in normal traffic mode as Non-VPN, and videos captured using a VPN are labeled as VPN. In this case, we get a dataset containing two labels, VPN and Non-VPN, and call it a traffic dataset. The improved model is trained on the aforementioned datasets. Our proposed method SDF, outperforms the other two techniques in all datasets, as shown in Figure 12. The results are summarized in Table 4.

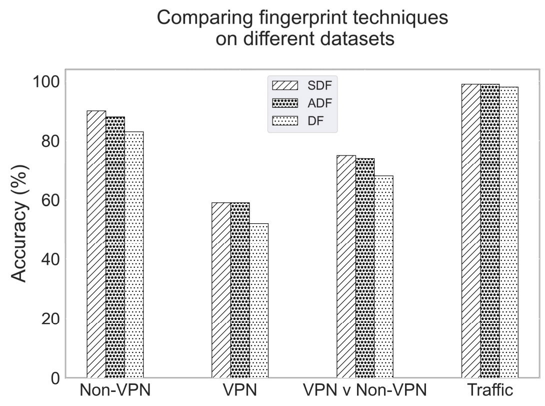

FIGURE 12. Accuracy comparison with different fingerprinting techniques on different dataset.

TABLE 4. Accuracy comparison of datasets on different fingerprinting methods.

Dataset	SDF	ADF	DF
Non-VPN	90%	83%	88%
VPN	59%	52%	59%
VPN v Non-VPN	75%	68%	74%
Traffic (VPN v Non-VPN)	99%	98%	99%

VI. DISCUSSION

From the results, the proposed framework for video and traffic identification is quite persistent. The performance of the baseline model is increased up to 12.21% with hyper-parameter tuning. The SDF technique outperforms other techniques in all datasets, as demonstrated in Section V. However, identifying video streams over VPN is relatively more challenging to detect compared to Non-VPN streams. Moreover, the accuracy of distinguishing between the VPN and Non-VPN traffic is 99%.

The proposed framework is quite feasible for detecting the known videos in the network as only 55 streams of a single video are required for training. However, in a real-world scenario, there are more unknown videos than known videos. Moreover, our proposed technique requires a huge storage capacity and computational requirement for video detection.

As the proposed framework works on the known videos, the model must be trained on a large dataset, limiting the scope of implementation.

Moreover, there are some shortcomings of this technique. The framework is set back by the phenomenon of *'concept drift'*. Hence, the proposed model requires a substantial amount of computational and space requirements at the observer's end, thus creating a challenge for large-scale deployment. Moreover, detection is only possible if the video is streamed exactly from the start to the first 120 seconds. Changing the video runtime between the first 120 seconds can lead to abnormal predictions

VII. CONCLUSION AND FUTURE WORK

Irregularities and inconsistencies due to the VBR encoding of the video make it challenging to identify the videos in the network traffic. To address the aforementioned problem, this paper converts BPSs into BPPs and presents a stable fingerprinting method, SDF. The SDF works on the difference between the BPPs to identify the VBR video streamed in encrypted network traffic. The created SDFs are used to train the CNN model. After tuning the model's hyperparameters, the model achieves an accuracy of 90% and 99% in predicting videos and classifying traffic, respectively. Additionally, the effects of variable period length on the model's prediction accuracy are yet to be analyzed. We aim to modify the technique to cope with the concept drift problem in the future. Observing the effect of variable period length and finding the optimal value will make this technique more foolproof and increase the practical deployment applications.

REFERENCES

[1] U. Cisco. (2020). *CISCO Annual Internet Report (2018–2023) White Paper*. Accessed: Dec. 15, 2021. [Online]. Available: https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/whitepaper-c11-741490html

[2] M. U. S. Khan, A. Abbas, A. Rehman, and R. Nawaz, "HateClassify: A service framework for hate speech identification on social media," *IEEE Internet Comput.*, vol. 25, no. 1, pp. 40–49, Jan. 2020.

[3] M. Khan, D. Baig, U. S. Khan, and A. Karim, "Malware classification framework using convolutional neural network," in *Proc. Int. Conf. Cyber Warfare Secur. (ICCWS)*, Oct. 2020, pp. 1–7.

[4] A. Reed and B. Klimkowski, "Leaky streams: Identifying variable bitrate DASH videos streamed over encrypted 802.11n connections," in *Proc. 13th IEEE Annu. Consum. Commun. Netw. Conf. (CCNC)*, Jan. 2016, pp. 1107–1112.

[5] R. Dubin, O. Hadar, I. Richman, O. Trabelsi, A. Dvir, and O. Pele, "Video quality representation classification of safari encrypted DASH streams," in *Proc. Digit. Media Ind. Acad. Forum (DMIAF)*, Jul. 2016, pp. 213–216.

[6] A. Bremler-Barr, Y. Harchol, D. Hay, and Y. Koral, "Deep packet inspection as a service," in *Proc. 10th ACM Int. Conf. Emerg. Netw. Exp. Technol.*, Dec. 2014, pp. 271–282.

[7] S. Miller, K. Curran, and T. Lunney, "Detection of virtual private network traffic using machine learning," *Int. J. Wireless Netw. Broadband Technol.*, vol. 9, no. 2, pp. 60–80, Jul. 2020.

[8] M. U. S. Khan, M. Jawad, and S. U. Khan, "Adadb: Adaptive diff-batch optimization technique for gradient descent," *IEEE Access*, vol. 9, pp. 99581–99588, 2021.

[9] K. S. Zaidi, S. Hina, M. Jawad, A. N. Khan, M. U. S. Khan, H. B. Pervaiz, and R. Nawaz, "Beyond the horizon, backhaul connectivity for offshore IoT devices," *Energies*, vol. 14, no. 21, p. 6918, Oct. 2021.

[10] M. U. S. Khan, A. Abbas, M. Ali, M. Jawad, and S. U. Khan, "Convolutional neural networks as means to identify apposite sensor combination for human activity recognition," in *Proc. IEEE/ACM Int. Conf. Connected Health, Appl., Syst. Eng. Technol.*, Sep. 2018, pp. 45–50.

[11] S. Mohammad, M. U. S. Khan, M. Ali, L. Liu, M. Shardlow, and R. Nawaz, "Bot detection using a single post on social media," in *Proc. 3rd World Conf. Smart Trends Syst. Secur. Sustainablity (WorldS)*, Jul. 2019, pp. 215–220.

[12] O. Jogunola, B. Adebisi, K. V. Hoang, Y. Tsado, S. I. Popoola, M. Hammoudeh, and R. Nawaz, "CBLSTM-AE: A hybrid deep learning framework for predicting energy consumption," *Energies*, vol. 15, no. 3, p. 810, Jan. 2022.

[13] S.-U. Hassan, M. Shabbir, S. Iqbal, A. Said, F. Kamiran, R. Nawaz, and U. Saif, "Leveraging deep learning and SNA approaches for smart city policing in the developing world," *Int. J. Inf. Manage.*, vol. 56, Feb. 2021, Art. no. 102045.

[14] H. Waheed, M. Anas, S.-U. Hassan, N. R. Aljohani, S. Alelyani, E. E. Edifor, and R. Nawaz, "Balancing sequential data to predict students at-risk using adversarial networks," *Comput. Electr. Eng.*, vol. 93, Jul. 2021, Art. no. 107274.

[15] F. Zaman, M. Shardlow, S.-U. Hassan, N. R. Aljohani, and R. Nawaz, "HTSS: A novel hybrid text summarisation and simplification architecture," *Inf. Process. Manage.*, vol. 57, no. 6, Nov. 2020, Art. no. 102351.

[16] A. Dvir, A. K. Marnerides, R. Dubin, and N. Golan, "Clustering the unknown—The YouTube case," in *Proc. Int. Conf. Comput., Netw. Commun. (ICNC)*, Feb. 2019, pp. 402–407.

[17] A. Dvir, A. K. Marnerides, R. Dubin, N. Golan, and C. Hajaj, "Encrypted video traffic clustering demystified," *Comput. Secur.*, vol. 96, Sep. 2020, Art. no. 101917.

[18] A. Reed and M. Kranch, "Identifying HTTPS-protected Netflix videos in real-time," in *Proc. 7th ACM Conf. Data Appl. Secur. Privacy*, Mar. 2017, pp. 361–368.

[19] R. Schuster, V. Shmatikov, and E. Tromer, "Beauty and the burst: Remote identification of encrypted video streams," in *Proc. 26th Secur. Symp.*, 2017, pp. 1357–1374.

[20] J. Gu, J. Wang, Z. Yu, and K. Shen, "Walls have ears: Traffic-based side-channel attack in video streaming," in *Proc. IEEE Conf. Comput. Commun.*, Apr. 2018, pp. 1538–1546.

[21] M. U. S. Khan, S. M. A. H. Bukhari, S. A. Khan, and T. Maqsood, "ISP can identify YouTube videos that you just watched," in *Proc. Int. Conf. Frontiers Inf. Technol. (FIT)*, Dec. 2021, pp. 1–6.

[22] S. Chen, R. Wang, X. Wang, and K. Zhang, "Side-channel leaks in web applications: A reality today, a challenge tomorrow," in *Proc. IEEE Symp. Secur. Privacy*, Jan. 2010, pp. 191–206.

[23] J. Han, C. Qian, P. Yang, D. Ma, Z. Jiang, W. Xi, and J. Zhao, "GenePrint: Generic and accurate physical-layer identification for UHF RFID tags," *IEEE/ACM Trans. Netw.*, vol. 24, no. 2, pp. 846–858, Apr. 2016.

[24] M. Korczynski and A. Duda, "Classifying service flows in the encrypted skype traffic," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Jun. 2012, pp. 1064–1068.

[25] W. Wang and D. N. Cheng, "Skype traffic identification based on trends-aware protocol fingerprints," in *Vehicle, Mechatronics and Information Technologies II* (Applied Mechanics and Materials), vol. 543. Zurich, Switzerland: Trans Tech Publications, 2014, pp. 2249–2254.

[26] X. Gong, N. Kiyavash, and N. Borisov, "Fingerprinting websites using remote traffic analysis," in *Proc. 17th ACM Conf. Comput. Commun. Secur.*, 2010, pp. 684–686.

[27] X. Cai, X. C. Zhang, B. Joshi, and R. Johnson, "Touching from a distance: Website fingerprinting attacks and defenses," in *Proc. ACM Conf. Comput. Commun. Secur.*, 2012, pp. 605–616.

[28] T. Wang, X. Cai, R. Nithyanand, R. Johnson, and I. Goldberg, "Effective attacks and provable defenses for website fingerprinting," in *Proc. 23rd Secur. Symp.*, 2014, pp. 143–157.

[29] F. Zhang, W. He, X. Liu, and P. G. Bridges, "Inferring users' online activities through traffic analysis," in *Proc. 4th ACM Conf. Wireless Netw. Secur.*, 2011, pp. 59–70.

[30] M. Conti, L. V. Mancini, R. Spolaor, and N. V. Verde, "Analyzing Android encrypted network traffic to identify user actions," *IEEE Trans. Inf. Forensics Security*, vol. 11, no. 1, pp. 114–125, Jan. 2015.

[31] R. Irfan, O. Khalid, M. U. S. Khan, F. Rehman, A. U. R. Khan, and R. Nawaz, "SocialRec: A context-aware recommendation framework with explicit sentiment analysis," *IEEE Access*, vol. 7, pp. 116295–116308, 2019.

[32] Z. Zhou, Z. Yang, C. Wu, W. Sun, and Y. Liu, "LiFi: Line-of-sight identification with WiFi," in *Proc. IEEE Conf. Comput. Commun.*, Apr. 2014, pp. 2688–2696.

[33] X. Chen, X. Wu, X.-Y. Li, X. Ji, Y. He, and Y. Liu, "Privacy-aware high-quality map generation with participatory sensing," *IEEE Trans. Mobile Comput.*, vol. 15, no. 3, pp. 719–732, Mar. 2015.

[34] Y. Guo, L. Yang, B. Li, T. Liu, and Y. Liu, "RollCaller: User-friendly indoor navigation system using human-item spatial relation," in *Proc. IEEE Conf. Comput. Commun.*, Apr. 2014, pp. 2840–2848.

[35] Q. Ma, S. Zhang, T. Zhu, K. Liu, L. Zhang, W. He, and Y. Liu, "PLP: Protecting location privacy against correlation analyze attack in crowd-sensing," *IEEE Trans. Mobile Comput.*, vol. 16, no. 9, pp. 2588–2598, Sep. 2016.

[36] M. Conti, N. Dragoni, and V. Lesyk, "A survey of man in the middle attacks," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 3, pp. 2027–2051, 3rd Quart., 2016.

[37] T. Mangla, E. Halepovic, M. Ammar, and E. Zegura, "Using session modeling to estimate HTTP-based video QoE metrics from encrypted network traffic," *IEEE Trans. Netw. Service Manage.*, vol. 16, no. 3, pp. 1086–1099, Sep. 2019.

[38] S. Wassermann, M. Seufert, P. Casas, L. Gang, and K. Li, "Let me decrypt your beauty: Real-time prediction of video resolution and bitrate for encrypted video streaming," in *Proc. Netw. Traffic Meas. Anal. Conf. (TMA)*, Jun. 2019, pp. 199–200.

[39] Y. Liu, S. Li, C. Zhang, C. Zheng, Y. Sun, and Q. Liu, "ITP-KNN: Encrypted video flow identification based on the intermittent traffic pattern of video and K-nearest neighbors classification," in *Proc. Int. Conf. Comput. Sci.* Cham, Switzerland: Springer, 2020, pp. 279–293.

[40] C. Gutterman, K. Guo, S. Arora, X. Wang, L. Wu, E. Katz-Bassett, and G. Zussman, "Requet: Real-time QoE detection for encrypted YouTube traffic," in *Proc. 10th ACM Multimedia Syst. Conf.*, Jun. 2019, pp. 48–59.

[41] S. Xu, S. Sen, and Z. M. Mao, "CSI: Inferring mobile ABR video adaptation behavior under HTTPS and QUIC," in *Proc. 15th Eur. Conf. Comput. Syst.*, Apr. 2020, pp. 1–16.

[42] P. Ameigeiras, J. J. Ramos-Munoz, J. Navarro-Ortiz, and J. M. Lopez-Soler, "Analysis and modelling of YouTube traffic," *Trans. Emerg. Telecommun. Technol.*, vol. 23, no. 4, pp. 360–377, Jun. 2012.

[43] R. Ravattu and P. Balasetty, "Characterization of YouTube video streaming traffic," M.S. thesis, School Comput., Blekinge Inst. Technol., Sweden, 2013. Accessed: Jun. 1, 2022. [Online]. Available: https://www.diva-portal.org/smash/get/diva2:830691/FULLTEXT01.pdf

[44] A. Rao, A. Legout, Y.-S. Lim, D. Towsley, C. Barakat, and W. Dabbous, "Network characteristics of video streaming traffic," in *Proc. 7th Conf. Emerg. Netw. EXperiments Technol.*, 2011, pp. 1–12.

[45] Y. Liu, S. Li, C. Zhang, C. Zheng, Y. Sun, and Q. Liu, "DOOM: A training-free, real-time video flow identification method for encrypted traffic," in *Proc. 27th Int. Conf. Telecommun. (ICT)*, Oct. 2020, pp. 1–5.

[46] M. U. S. Khan, S. M. A. H. Bukhari, T. Maqsood, M. A. B. Fayyaz, D. Dancey, and R. Nawaz, "SCNN-attack: A side-channel attack to identify YouTube videos in a VPN and non-VPN network traffic," *Electronics*, vol. 11, no. 3, p. 350, Jan. 2022.

[47] A. Z. Broder, "Some applications of Rabin's fingerprinting method," in *Sequences II*. Cham, Switzerland: Springer, 1993, pp. 143–152.

[48] M. Korczynski and A. Duda, "Markov chain fingerprinting to classify encrypted traffic," in *Proc. IEEE Conf. Comput. Commun.*, Apr. 2014, pp. 781–789.

[49] M. Shen, Y. Liu, S. Chen, L. Zhu, and Y. Zhang, "Webpage fingerprinting using only packet length information," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2019, pp. 1–6.

[50] S. Miskovic, G. M. Lee, Y. Liao, and M. Baldi, "AppPrint: Automatic fingerprinting of mobile applications in network traffic," in *Proc. Int. Conf. Passive Act. Netw. Meas.* Cham, Switzerland: Springer, 2015, pp. 57–69.

[51] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, Sep. 2015.

WALEED AFANDI is a Research Assistant with the Department of Computer Science, COMSATS University Islamabad, Abbottabad Campus. His research interest includes computer security.

SYED MUHAMMAD AMMAR HASSAN BUKHARI is a Senior Research Assistant with the Department of Computer Science, COMSATS University Islamabad, Abbottabad Campus. His research interest includes computer security.

MUHAMMAD U. S. KHAN (Member, IEEE) received the Ph.D. degree in electrical and computer engineering at North Dakota State University, USA, in 2015. He is an Assistant Professor with COMSATS University Islamabad, Abbottabad Campus. His research interests include data science, artificial intelligence, and computer security.

TAHIR MAQSOOD is currently an Assistant Professor with COMSATS University Islamabad, Abbottabad, Pakistan. His research interests include resource allocation, multi/manycore systems, reliable systems, the Internet of Things, and mobile edge computing.

SAMEE U. KHAN (Senior Member, IEEE) received the Ph.D. degree from the University of Texas, in 2007. He was the Cluster Lead of the Computer Systems Research at the National Science Foundation, from 2016 to 2020, and the Walter B. Booth Professor at North Dakota State University. Currently, he is the James W. Bagley Chair Professor and the Head of the Department of Electrical and Computer Engineering with Mississippi State University (MSU). His work has appeared in over 400 publications. His research interests include optimization, robustness, and security of computer systems. He is an Associate Editor of IEEE Transactions on Cloud Computing and *Journal of Parallel and Distributed Computing*.