Communications of the ACM

Research and Advances

Artificial Intelligence and Machine Learning

Research highlights

Traffic Classification in an Increasingly Encrypted Web

By Iman Akbari, Mohammad A. Salahuddin, Noura Limam, Raouf Boutaba, Bertrand Mathieu, Stephanie Moteau, Stephane Tuffin, and Leni Aniva

Posted Oct 1 2022

Abstract

Traffic classification is essential in network management for a wide range of operations. Recently, it has become increasingly challenging with the widespread adoption of encryption in the Internet, for example, as a *de facto* in HTTP/2 and QUIC protocols. In the current state of encrypted traffic classification using deep learning (DL), we identify fundamental issues in the way it is typically approached. For instance, although complex DL models with millions of parameters are being used, these models implement a relatively simple logic based on certain header fields of the TLS handshake, limiting model robustness to future versions of encrypted protocols. Furthermore, encrypted traffic is often treated as any other raw input for DL, while crucial domain-specific considerations are commonly ignored. In this paper, we design a novel feature engineering approach used for encrypted Web protocols, and develop a neural network architecture based on stacked long short-term memory layers and convolutional neural networks. We evaluate our approach on a real-world Web traffic dataset from a major Internet service provider and mobile network operator. We achieve an accuracy of 95% in service classification with less raw traffic and a smaller number of parameters, outperforming a state-of-the-art method by nearly 50% fewer false classifications. We show that our DL model generalizes for different classification objectives and encrypted Web protocols. We also evaluate our approach on a public QUIC dataset with finer application-level granularity in labeling, achieving an overall accuracy of 99%.

Back to Top

Introduction

Traffic classification is quintessential for network operators to perform a wide range of network operation and management activities. This includes capacity planning, security and intrusion detection, quality of service (QoS) assurance, performance monitoring, volumetry, and resource provisioning, to name a few. For example, an enterprise network administrator or Internet service provider (ISP) may want to prioritize traffic for business critical services, identify unknown traffic for anomaly detection, or perform workload characterization for designing efficient resource management schemes to satisfy performance and resource requirements of diverse applications. Depending on the context, misclassification on a large scale may result in failure to deliver QoS guarantees, high operational expenses, security breaches, or even disruption in services.

Encrypted communication between clients and servers has now become the norm. Most prominent Web-based services are now running over hypertext transfer protocol secure (HTTPS). On the other hand, to improve security and quality of experience (QoE) for end users, new Web protocols (e.g., HTTP/2 and QUIC) have emerged, which overcome various limitations of HTTP/1.1. Using a real-world mobile traffic, we estimate that around 32% of all HTTPS sessions already use HTTP/2 as their underlying protocol. However, HTTP/2 features, such as payload encryption, multiplexing and concurrency, resource prioritization, and server push, add to the complexity of traffic classification. While a large body of literature harnesses the power of machine learning (ML) for different traffic classification objectives (e.g., service- and application-level, QoE prediction, security), there exist various limitations that must be addressed for its practical usage

For instance, the particular nature of encrypted traffic is not taken into account in many state-of-the-art approaches, which affects their performance and efficiency when applied to encrypted protocols. Due to a lack of standard framework for traffic classification, numerous works in traff classification (e.g., Lopez-Martin,[9] Yao[16]) pick their classes somehow arbitrarily, which are ofte inconsistent in granularity. Furthermore, many approaches (e.g., Lotfollahi,[10] Rezaei,[11] Wang,[14] Zou[17]) use datasets with a mixed set of protocols that are often easily distinguishable using header signatures, making it unrealistic to justify the use of computationally expensive ML models. In some cases (e.g., Brissaud[6]), traffic classification approaches rely on clever techniques to guide the models based on expert domain knowledge that can be jeopardized by small variations in the protocol.

Another important issue is that some protocol extensions, such as the server name indication (SNI) in transport layer security (TLS), can essentially reveal the server's identity, allowing for trivial classification of many traffic flows based on the server name. In this case, it can be argued that expensive and complex models are being used to learn a relatively simple logic, similar to those of a server name to label look-up table, which can be implemented deterministically. All of these issues call for a more comprehensive study of how deep traffic classification models behave on encrypted traffic, especially emerging Web protocols due to their ubiquity. They also underline the importance of developing general frameworks and guidelines for how encrypted Web traffic should be treated as a data type for future research in traffic classification.

In this paper, we leverage deep learning (DL) for service classification (e.g., video streaming, social media, Web mail) with a focus on new encrypted Web protocols, that is, HTTP/2 and QUIC, and overcome the abovementioned limitations. Unlike many works in this area, we focus exclusively on encrypted Web traffic and explore the challenges of unleashing the full potential of find complex patterns that are innate to each traffic class. We occlude parts of the input that DL model can use to learn a lazy and unsophisticated logic, and instigate how encrypted traffic

should be treated differently from general raw ML input, for example, images. We also place emphasis on a feature set that generalizes the applicability of the model for varied encrypted Web traffic classification objectives.

We propose a novel feature engineering approach for encrypted traffic classification that focuses on protocol-agnostic aspects of the encrypted Web traffic. In our approach, we make use of standard flow statistics, the traffic shape with respect to packet sizes, inter-arrival times (IATs), and direction, along with raw bytes from the TLS handshake packets. This is in contrast to most DL approaches for traffic classification, where the full raw traffic is fed to the DL model. We show the proposed feature set to be a better fit for the classification of encrypted traffic. We also develop a neural network architecture based on convolutional neural network (CNN) and stacked long short-term memory (LSTM) layers that are highly effective in leveraging the extracted features for distinguishing between different traffic classes. Our DL model identifies and correlates useful traffic traits, while being lighter in the number of trainable parameters and less likely to overfit, compared with the existing methods.

We use a real-world mobile traffic dataset from an ISP and demonstrate that our approach has an edge over the state-of-the-art in service classification over encrypted Web traffic. Using our model based on stacked LSTM layers, we achieve an accuracy of over 95% for classification exclusively over HTTPS (i.e., HTTP/1.1 and HTTP/2 over TLS), outperforming Rezaei[11] by a significant margin of nearly 50% fewer false classifications. We also show that our approach generally achieves higher accuracies as it is less prone to over-fitting. Furthermore, the variation of our model that uses CNN layers instead of stacked LSTM requires lower training time while still achieving a higher accuracy compared with the state-of-the-art one. We also showcase that our DL model generalizes for a finer classification granularity, that is, application-level classes. Furthermore, we show that our model adapts to a different encrypted Web protocol, that is, QUIC, by simply changing the training data. We achieve an accuracy of 97% in application-level classification and an accuracy of 99% on a public QUIC dataset.[12]

Back to Top

2. Related Works

Traffic classification using ML started in the early 2000s to distinguish between protocols (e.g., DNS, SMTP, and HTTP) in a network trace. Soon, the attention shifted toward more challenging traffic classification tasks, such as classification of encrypted Skype traffic (e.g., Bonfiglio[4]). This is particularly interesting as Skype operates on non-standard port numbers. DL introduced new opportunities in traffic classification by making it possible to feed large fine-grained feature vectors such as raw traffic to models, as opposed to aggregated statistics over entire sessions that required manual feature extraction efforts.

We can broadly categorize the typical features employed in the traffic classification literature for modeling traffic into the following groups: (i) *Flow statistics*: A standard flowmeter, such as CICFlowMeter,[8] yields the mean, standard deviation, minimum, maximum, of packet lengths, IATs, TCP flag counts, flow durations, number of packets, number of bytes, etc. These statistics constitute a feature vector for each flow and have been employed since the early traffic classification literature. (ii) *Raw bytes*: The actual flow bytes from packet headers and payloads have grown in popularity with the advent of DL. Their appeal is the leveraging of data in the

rawest form, as done in more conventional applications of DL such as computer vision. (iii) *Time series*: Following a fixed-size, a packet-level feature through all packets in a flow can yield a dynamic-sized time series feature representing the flow. For example, the sizes of the packets in a flow are a valid time series feature.

Aceto et al.[1] evaluate numerous DL approaches for the classification of mobile application traffics, using a proprietary dataset. They argue that there is no silver bullet, when it comes to the choice of a neural network for traffic classification. However, one-dimensional CNN and LSTM networks typically perform well due to the sequential nature of network traffic. Lopez-Martin et al.[9] combine LSTM and CNN layers for service classification on a time series (i.e., feature type iii). However, their classes are inconsistent in granularity and the model is essentially classifying protocols for some labels. The authors also show that traditional methods (i.e., based on lightweight ML models) are inferior to DL models in accuracy, by a significant margin. The high accuracies reported for traditional models in other works often pertain to different classification tasks (e.g., QoS) or mixed-protocol datasets (e.g., Williams[15]).

Bronzino et al.[7] explore classification and regression models for inferring important QoS metrics of encrypted video traffic. The authors make use of traditional ML models such as linear regression, support vector regression, decision tree (DT), and random forest (RF) regressors, as well as RF classifiers. They leverage a carefully crafted set of statistical features (i.e., feature type i) to effectively predict the target metrics (i.e., playback startup delay and resolution) in detecting video resolution. However, their feature engineering is tailored for a specific task and does not generalize to various traffic classification objectives.

Rezaei et al.[11] leverage CNN and CNN-LSTM architectures with certain adjustments to achieve high classification performance. Their focus, like ours, is on encrypted Web traffic such as HTTPS. However, their dataset also contains non-encrypted traffic. For their CNN-LSTM model, the authors model the traffic sessions as a series of flows. From each flow, the first six packets are fed raw to the flow-level model (i.e., feature type ii). Their dataset is comprised of real-world mobile traffic, including SSL and TLS flows, with application labels. The authors report high classification accuracy, which drops for exclusively HTTPS traffic. In their *post hoc* analysis, the authors identify the importance of different parts of the TLS headers to their model by masking different portions of the input and evaluating its impact on the model accuracy. They uncover that the model does in fact heavily fit to cipher info and the SNI field, to the point that the accuracy of the model significantly drops when SNI records are occluded. Due to its high relevance, we use Rezaei[11] for comparison of our model to the state-of-the-art one.

Back to Top

3. Methodology

3.1. Feature engineering

Numerous works in DL-based traffic classification feed raw traffic bytes to a neural network model. Indeed, DL models are powerful enough to extract meaningful features from raw input on their own, provided a sufficiently large dataset. The notion of leveraging raw traffic bytes as model input is inspired from more conventional domains of DL, such as computer vision.

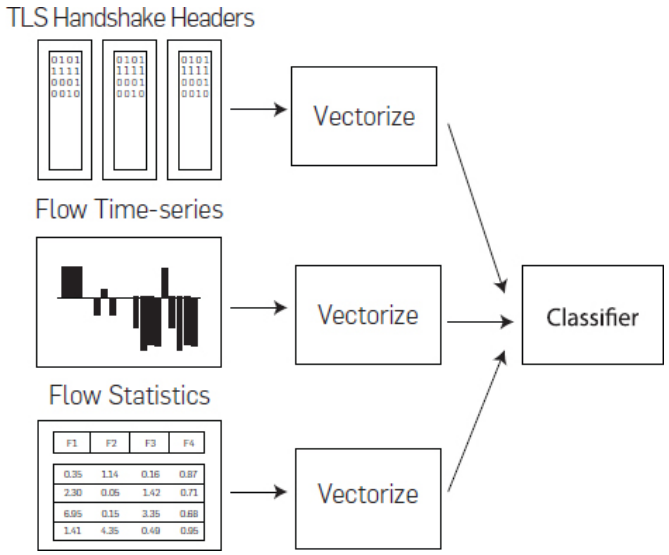
However, as with the adoption of DL in any new domain, there are important considerations in traffic classification based on domain-specific knowledge of the task and the nature of data.

An important distinction between network traffic and images is encryption, which is becoming the norm in ordinary Web usage. A traffic flow or packet is often almost completely encrypted, except for the initial handshake and some of the header fields that are transmitted in plain text. Therefore, in the computer vision analogy, a traffic flow is like an image that is completely obfuscated except for a small area in it. Any effort to consume the encrypted portions of the traffic as the classification model input is essentially an attack on the established encryption algorithms, such as advanced encryption standard (AES), which is unrealistic.

Furthermore, it is crucial to consider what the DL model is exactly learning during training. For example, in an insightful *post hoc* analysis, Rezaei et al.[11] show that the accuracy of their DL model completely degrades when the SNI field or TLS cipher info is masked. This implies that typical neural network models trained on raw traffic basically implement a look-up table, which predicts a class based on the server's identity exposed by certain TLS extensions. We refer to the parts of the traffic that expose the server's identity as *canary features.*

There are three major drawbacks in relying on canary features: (i) An expensive deep neural network is used for implementing a relatively simple logic, which can be performed deterministically with a very low computational overhead. (ii) The performance of the DL mode is highly dependent on seeing large amounts of traffic from all relevant servers in a service category (e.g., traffic from all video streaming platforms) in training. In other words, the model i not really learning anything about the nature of video flows in general. (iii) The availability of these identifiers of the server (e.g., plain-text SNI field) in-the-clear is crucial for the utility of th DL approach. If the SNI field becomes outdated or encrypted in the future versions of TLS, which is not unlikely with the advent of encrypted SNI, the entire DL method can lose its effectiveness.

Our input to the DL model combines all three types of features, described in Section 2. As summarized in Figure 1, it is comprised of (i) TLS handshake header bytes, (ii) flow time series, and (iii) flow statistics.



re 1. TLS headers from the handshake, flow time series, and standard flow statistics as L model input.

First, we include raw bytes from the handshake in our input to the model. However, we remove the canary features such as SNI and cipher info in our preprocessing, to diminish the model's reliance on that information. Also, due to our focus on encrypted protocols, we assume that L5-7 payloads contain very little information as they are expected to be encrypted. Therefore, there is no utility in including entire packets in the DL model input and the aforementioned pay-loads only create more ways for the model to overfit. Besides, packets other than the handshake packets (i.e., ClientHello and ServerHello messages) are redundant and expose virtually no meaningful information to the model. Thus, the raw traffic data for our DL model input is truncated after the TLS headers of the handshake packets.

E' così?

ovvio

Second, we steer our DL model's focus on traffic aspects that are hardly affected by encryption. While the TLS records and extensions will evolve over time and new encrypted protocols will emerge with radically different characteristics, the traffic shape would always be available regardless of the underlying protocol. We hypothesize that the traffic flow time series of packet sizes, directions, and IATs contain useful information for service classification, as they are relatively independent from the implementation details of the protocol. Though it is possible to design strategies to obfuscate such information, it would have a negative impact on bandwidth, latency, and QoS, as it entails sending redundant traffic or delaying packets to manipulate the time series. Therefore, it is unrealistic that there would be enough motivation for introducing such measures in ubiquitous Web protocols. By combining these features with raw bytes, we can crea a powerful feature set that can be used for learning the nature of traffic, as well as identifying useful parts of the secure protocol's headers for identifying applications.

Feedback

Lastly, traditional flow statistics measured by standard flowmeters can also assist the model in traffic classification. Examples of traditional features include mean, standard-deviation, and median of packet sizes, number of different TCP flags, duration of the flow. These features have been used for a variety of traffic classification tasks for over two decades and continue to be a simple yet powerful tool for distinguishing between different classes of traffic. This also allows for our overarching methodology to generalize for works such as Bronzino,[7] where a set of features are picked by domain experts for a particular traffic classification or regression task.

The combination of handshake features and flow time series was first proposed by Anderson[2] to detect malicious traffic. Aside from the use of statistical features as a third input, a key distinction between our approach and Anderson[2] is the use of DL to extract useful features of the handshake, while they require a domain expert to cherry-pick them for the TLS protocol. Though our research is focused on encrypted Web protocols and mostly revolves around TLS, our feature engineering methodology is protocol-agnostic. Regardless of a protocol's implementation details, it is expected to have a negotiation or handshake segment, while the rest of the traffic would be fully encrypted. This segment will make up the only raw inputs to the model. The flow time series; that is, traffic shape and timing, as well as flow statistics, will always be available in IP. Therefore, the model will have to be retrained and specialized for new protocol versions as they evolve, but our overarching feature engineering methodology will still be applicable.

3.2. Model architecture

Our neural network architecture reflects the structure of the features presented earlier. As shown in Figure 2, our neural network model separately processes the flow time series, the TLS handshake headers, and the standard flow statistics as inputs. Each of the three inputs is fed to a

separate set of neural network layers, and the output of those layers is later concatenated and passed through additional fully connected layers to produce the final prediction.

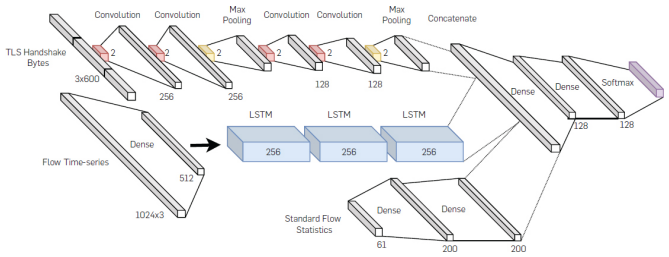


Figure 2. Tripartite neural network architecture.

The raw handshake bytes are fed to a deep one-dimensional CNN with max-pooling layers in between. The structure of these layers is quite standard and a one-dimensional equivalent of commonly used computer vision models, which has proven effective on network traffic.[10,11] We only feed the first C bytes of up to three ClientHello and ServerHello packets from the flow to the model. In our experiments, $C = 600$, which is picked through a hyper-parameter search. We did not notice a disadvantage in omitting the rest of the traffic, which has strong implications for future research in this area.

The flow time series has three channels: (i) IAT, (ii) size, and (iii) direction. In our experiments, we found that a stacked LSTM architecture preceded by a dense layer is extremely effective in processing the flow time-series features, while one-dimensional CNNs are also viable (cf. Sectio 4.2). In our implementation, we use a stack of three LSTM layers going through the flow time series in both directions. We also include flow statistics extracted using CICFlowMeter. Since these features do not have a natural ordering or sequentiality, a fully connected network is used to ingest them.

One of the major advantages of our feature engineering is the ability to include information about a large number of packets without substantially increasing the model size. In a classic raw input approach, it is normal to include the first b bytes of the first k packets of a flow to the model. The size of the input grows linearly by increasing k, which can create a super-linear increase in the number of model parameters and quickly lead to overfitting. In contrast, our model limits the raw traffic to the handshake packets and uses a lightweight representation with only three channels for the other packets of the flow. This allows the model's scope to grow and consider hundreds of packets without a significant impact on its complexity. The outputs from these three parts (i.e., flow time series and statistics, and TLS headers) are concatenated and passed through multiple additional dense layers, which yields the output of the network as a softmax layer.

Our early experiments showed that the models are highly prone to overfitting. This is not surprising considering the fact that the number of parameters in the model to be trained is in the order of millions. It is not uncommon for traffic classification models to be trained on datasets that have an order of magnitude fewer entries than the number of trainable parameters in the model. To overcome the problem of overfitting, we use very high dropouts (i.e., up to 50% at some layers), especially in the final dense layers. As show-cased in Section 4.2, our feature engineering itself has a tremendous effect in lowering the chance of overfitting when compared with the ntional raw traffic input.

3.3. Data preprocessing

We design our preprocessing to be performed in a distributed fashion using Apache Spark. We begin by extracting the flows (i.e., 5-tuples of src/dst IP/port and protocol) *via* standard flowmeter such as YAF. We then filter flows with TLS packets, as we are only interested in encrypted Web traffic. Basic flow information, such as the flow start and end times, packet count, byte count, flow time series, are then extracted along with the statistical features, which are computed using CICFlowMeter and stored as metadata. The SNI domain name is also stored to assign class labels based on a look-up table. Next, we group the flows having the same TLS session ID together. If TLS session ID does not exist, time proximity, and NAT-aware IP and port numbers are used. For each unlabeled flow *f*, we check other flows in the same session as *f* and use their label for *f*. Often in multi-flow TLS sessions, only the first flow contains the SNI record. The flows can then be vectorized into a time series of binary information as follows: (i) mask IP addresses by injecting zeroes even if they are already randomized, (ii) remove TLS cipher information, (iii) mask the SNI record, and (iv) truncate to MTU size or zero-pad the packet bytes—ensure fixed vector size. Finally, the raw traffic bytes are written to binary files, with each entry having an array of vectorized bytes from up to three handshake packets. We include flow statistics and a time series of maximum length 1024 with the three channels for packet sizes, directions (±1), and IATs for each entry as well.

Back to Top

4. Evaluation

4.1. Datasets

Orange'20 dataset. The primary dataset used in our work is provided by Orange S.A., a major ISP in Europe. The dataset was collected on July 11, 2019, for about 80 minutes, from the ISP's mobile network. For privacy concerns, the IP addresses are masked and the packet pay-loads are removed with the exception of TLS headers. The entire dataset has more than 800K unlabeled flows, where ~300K are TLS flows and of interest to us. We use the SNI field to label the TLS flows with the following service categories: (i) chat, (ii) download, (iii) games, (iv) mail, (v) search, (vi) social, (vii) streaming, and (viii) Web. Each domain name from the SNI field is matched against a set of regular expressions that are either carefully handpicked (i.e., by monitoring the traffic of prominent Web sites and mobile apps, e.g., Netix, YouTube, and AppStore), or gathered from a dataset of categorized domain names, such as the Blacklists UT1 dataset.[3] For certain providers (e.g., Google and Facebook), extra care must be taken, as similar sub-domains may be shared between multiple service categories. A total of 119,565 out of the 343,228 TLS flows are labeled, using our approximate labeling scheme, with both manually picked URLs and the UT1 dataset. Note that not all TLS sessions can be identified, as the application-layer protocol negotiation (ALPN) and next protocol negotiation (NPN) records may not be available. Therefore, we end up with a highly imbalanced distribution of service categories in the labeled dataset.

UCDavis QUIC dataset. The QUIC dataset has been recently released by Rezaei et al.[12] It consists of 3637 flows, classified into Google Docs, Google Drive, Google Music, Google Search, and YouTube. This is natural as Google is currently the primary advocate for the QUIC

protocol's adoption in the industry. The dataset is relatively balanced, with no class being twice as large as the others. Furthermore, it is partly generated by human users and partly *via* automated agents.

For comparison, we implement the CNN and CNN-LSTM models proposed by Rezaei et al.,[11] which have shown good performance in application-level traffic classification. The authors model the input as a series of flows, which can be thought of as a *user session.* The CNN model operates on the *flow* level; that is, the first 256 bytes of the first six packets of a single flow in the series are fed to a deep CNN. The CNN model is very similar to the header part of our model (cf. Figure 2). On the other hand, the CNN-LSTM model receives the session (i.e., a time series of flows) as its input, and essentially makes the CNN model time-distributed over the flows of each session and labels the entire session. Similar architectures have been employed over the years for encrypted traffic classification,[1,14] making it an ideal baseline to compare our model against.

4.2. TLS classification at service level

We start by evaluating the performance of our feature engineering approach and DL model architecture on the Orange'20 dataset. The data is pre-processed according to Section 3.3 and comprises of TLS flows only. We compare our DL model against the state-of-the-art CNN and CNN-LSTM architectures, showing a clear advantage and asserting our contributions.

Our model is trained using the Adam optimizer for 40 epochs, with 20% of the dataset used for validation. The learning rate is set to 0.001 at first and reduced every 10 epochs. The results of th experiment are shown in Figure 3, with an overall accuracy and weighted average F1-score of 95.56% and 95.58%, respectively. Figure 3b shows the per-class precision, recall, and F1-score of our model. The F1-score is over 94% for all classes, which implies very good stability despite the highly imbalanced classes in the dataset. This can be attributed to the upsampling strategy employed during training. As a baseline, a C4.5 model is trained on statistical flow features. C4.5, among other DT-based algorithms, is a popular choice in traditional traffic classification[5,13] but only achieves an accuracy of 81.39%, as shown in the table here. We attribute the disadvantage of the traditional approach in part to its sole reliance on high-level statistics and not being able to make distinctions based on more fine-grained details of the traffic shape. Furthermore, the advantage of our model is clear when compared against the UCDavis CNN model, as shown in the table here. When evaluated on the Orange'20 dataset, the UCDavis CNN model in Rezaei[11] achieves an accuracy of 91.09% after 20 epochs, which is 4.5% lower than our tripartite model in Figure 2. This is a significant gain in performance with 50.39% reduction in false classifications.

	W Avg precision (%)	W Avg recall (%)	W Avg F1-score (%)	Accuracy (%)	Epoch time (s)
Full model (SLSTM)	95.62	95.56	95.57	95.56	2584
Full model (CNN)	94.54	94.42	94.37	94.43	232
Flow-only model (SLSTM)	86.71	86.51	86.56	86.51	1814
Flow-only model (CNN)	76.77	73.17	73.76	73.17	211
UCDavis CNN[11]	91.09	91.06	91.04	91.05	168
UCDavis CNN-LSTM[11]	89.74	89.72	89.73	89.72	245
Traditional baseline (C4.5)	81.56	81.39	81.41	81.39	18*

Table. Performance comparison of TLS flow classification models (*C4.5 time is reported for entire training).

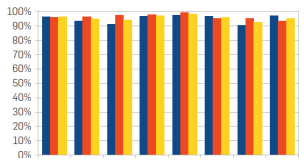
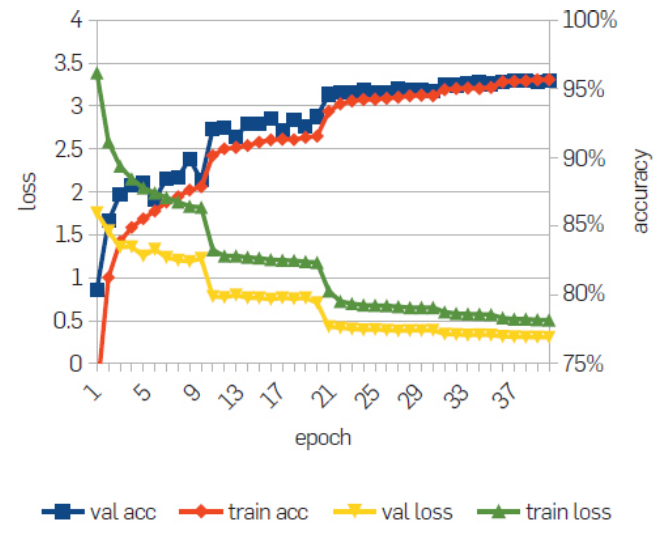
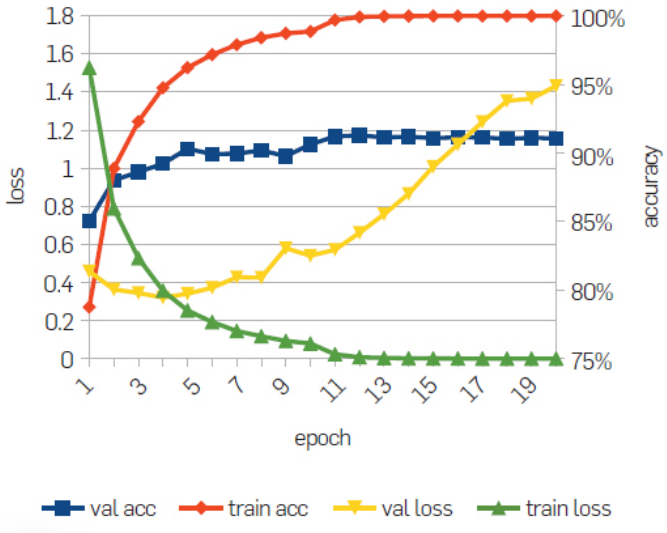


Figure 3. Service-level confusion matrix and per-class precision, recall, and F1-score for our model's evaluation on the Orange'20 dataset. Our classifier consistently achieves +94% F1-score across all classes.

In Figure 4, we highlight the training progress to reason about this performance gain. The figure shows validation loss, training loss, and accuracy at the end of each training epoch. Evidently, the UCDavis CNN model quickly overfits to the dataset. This is primarily due to a larger raw traffic input, only a part of which is actually useful to the model. After 12 epochs, the training accuracy is perfect, while the validation performance fails to improve. In contrast, the validation and training accuracies converge very well in the case of our model, as our feature engineering only provides the useful information for encrypted flows (i.e., handshake and flow shape). The implications of these results are significant. Despite having access to twice the number of packets, the competitor UCDavis CNN model is far less effective, as all meaningful information lies in the handshake. Exposing a larger chunk of the raw traffic to the UCDavis CNN model simply confuses the model and provides more ways to overfit.



(a) Our model



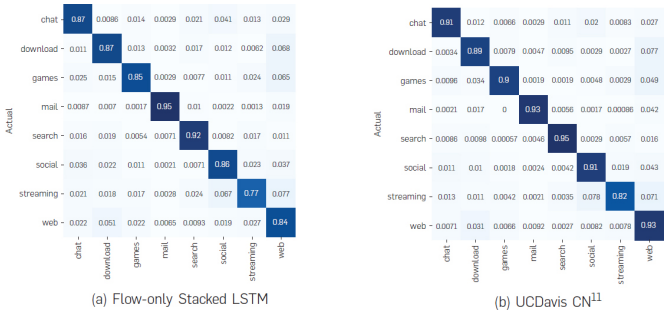
(b) UCDavis CNN[11]

Figure 4. Training progression, validation accuracy, and loss, while training our model and the UCDavis CNN.

The accompanying table depicts the performance of other variations of our model. We replace the Stacked LSTM (SLSTM) layers in the original model (cf. Section 3.2) with a deep one-dimensional CNN, which is also a reasonable network for consuming a one-dimensional time series. Though inferior to stacked LSTM, the accuracy and F1-score of the resultant model are 94.43% and 94.37%, respectively. Nevertheless, it has a clear advantage over the UCDavis CNN model, which only processes raw traffic, with 37.77% less false classifications. The advantage of employing CNNs on the flow time series side of the model, however, is in higher training speed (232 vs. 2584 seconds/epoch), which is close to our competitor model despite being much more accurate. LSTM networks are notorious for being computationally expensive to train, and stacked LSTM layers are even more so. Nevertheless, as model training is often a one-time investment and with the rapid advancement of computational hardware, this is a reasonable cost for higher classification accuracy.

We attribute the superior performance of our model to the flow aspect of our feature engineering and the stacked LSTM layers. In fact, the flow time series, despite being a simple feature set to model the traffic, is quite effective by itself. In the table here, we also showcase results of our model variation with all the other inputs and their corresponding layers removed, except the flow time series. We refer to these models as *Flow-only*. In this case, the stacked LSTM and deep one-dimensional CNN architectures achieve an accuracy of 86.51% and 73.17%, respectively. Although being inferior in performance to models that also include raw traffic as input, it is important to note that the flow time series features will always be available regardless of how the encrypted protocols evolve. These features enable a model to learn about the nature of traffic categories themselves, rather than fingerprinting a particular set of servers. Therefore, for all future research, we instigate the use of these flow features as a baseline for evaluations.

It is important to note that the UCDavis CNN model depicts a high misclassification between the streaming and social classes, where mutual providers such as Facebook and Twitter exist, as shown in Figure 5. This is a reoccurring issue with DL models for traffic classification that only rely on raw TLS bytes and are more tuned toward identifying a *server* rather than a *service*. The flow-only model, despite having access to very simplistic traffic features, makes fewer misclassifications between these classes and, when used together with the handshake bytes in our full model, is able to alleviate the difficulties in distinguishing between the two classes.



(a) Flow-only Stacked LSTM (b) UCDavis CNN[11]

Figure 5. Confusion matrices for our flow-only stacked LSTM model and the UCDavis CNN model.

A rather surprising result in the table here is the performance of the UCDavis CNN-LSTM

model.[11] The UCDavis CNN-LSTM model is time-distributed over the flows of a *session* and theoretically has access to more information in comparison with our model that processes flows individually. However, the model's access to full traffic bytes does not work in its interest, and though having more parameters and capacity than the UCDavis CNN model, it overfits more severely to the data achieving a slightly less accuracy of around 90%. In fact, similar to the UCDavis CNN model, it quickly rises and achieves perfect training accuracy at around the seventeenth epoch, but fails to increase the validation accuracy any further.

The final insight here pertains to the HTTP version and how it affects the performance of our model. In Section 1, we mentioned that HTTP/2 brings new features to the Web, but at the same time complicates traffic classification. We evaluated our model's performance on different subsets of the validation set, based on the HTTP version. Recall that not all flows captured in the Orange'20 dataset have the NPN or ALPN records available to identify the protocol used over TLS. The *known* and *unknown* in Figure 6 elude to this fact.

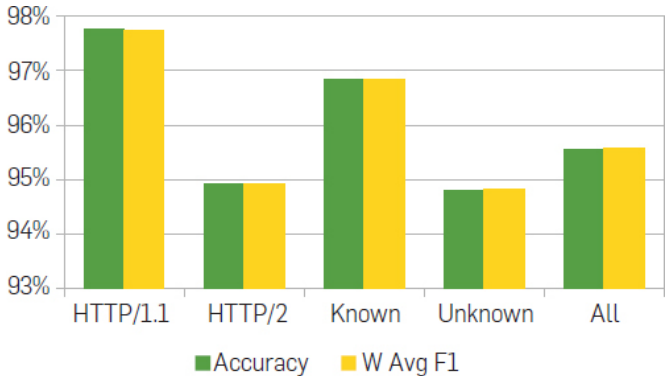


Figure 6. Impact of different HTTP versions on the performance of our model.

As expected, the performance of the model on HTTP/1.1 is higher than HTTP/2, that is, 97.75% vs. 94.91%, due to the latter being a more complex protocol with features such as multiplexing, which makes traffic classification more difficult. More importantly, the flows captured without the ALPN/NPN records are also generally harder to classify than the rest. This is due to the fact that there is a higher likelihood that these flows are captured from the middle of a session. Hence, they contain less information in their beginning for the model to leverage. It should also be noted that there are twice as many HTTP/1.1 flows as HTTP/2 flows in the training set. This reinforces the model's better performance on HTTP/1.1.

4.3. TLS classification at application level

Many works in traffic classification (e.g., Aceto,[1] Rezaei[11]) focus on application-level classification that is at a finer granularity than our labels in Section 4.2. While counter-intuitive, application-level classification is often an easier task, especially when the model is closed world (i.e., dataset entries strictly belong to one of the *n* known applications) or canary features are not occluded. If the DL model is trained with the objective of a look-up table for identifying servers themselves (cf. Section 3.1), application-level classification is generally easier for the model, as it does not need to learn what behaviors are *shared* between different applications of the same service category.

To evaluate our model in application-level classification, we identified 19 fine-grained labels that

have enough representative flows for the training to be consistent. These labels make up for 82,776 flow entries of the dataset (i.e., ~70%). Figure 7 depicts the result of employing our model by simply modifying the last softmax layer to accommodate 19 fine-grained classes instead of the 8 service categories. The overall accuracy of the model is 97.08%. Despite having more classes, the accuracy of the model is higher than service-level classification, due to the raw bytes part of the model being extremely effective in fingerprinting specific servers. One side effect is that the different services from the same provider (e.g., Facebook video and Facebook social) have higher cross-misclassifications, as evident in Figure 7.

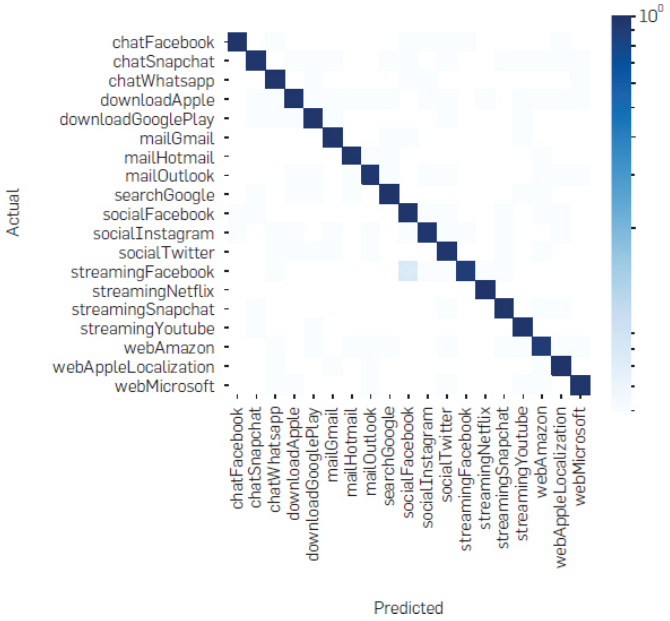


Figure 7. Confusion matrix of our model for application-level classification on the Orange'20 dataset.

A key takeaway from this experiment is that a good feature engineering approach for encrypted traffic classification can be adapted to different classification tasks, as it is good in capturing the nature of traffic. It will be an interesting part of future research to leverage the feature engineering presented in Section 3.1 in areas other than service- and application-level classification, such as QoS classification and security.

4.4. QUIC classification

We evaluated our model on the UCDavis QUIC Dataset (cf. Section 4.1.2), which only includes the traffic shape time series and not the actual network traces. To adapt to the dataset structure, we modified our approach by only activating the flow time-series part of the model and conducted the training for 20 epochs.

In evaluation, our model achieve a high validation accuracy (i.e., 99.37%), which is higher than the best one reported by Rezaei et al.[12] for their CNN model (i.e., ~98%), regardless of whether their semi-supervised scheme (i.e., pretraining on unlabeled data first) is carried out or not. Figure 8 shows the result of the classification, which re-affirms that our proposed feature engineering is indeed a good indicator of the traffic class, and can adapt well to different encrypted Web cols. Our model achieves high accuracies despite the fact that QUIC is a more challenging col with a larger encrypted portion. These results also validate the utility of the stacked

LSTM architecture used in the flow time series part of our model.

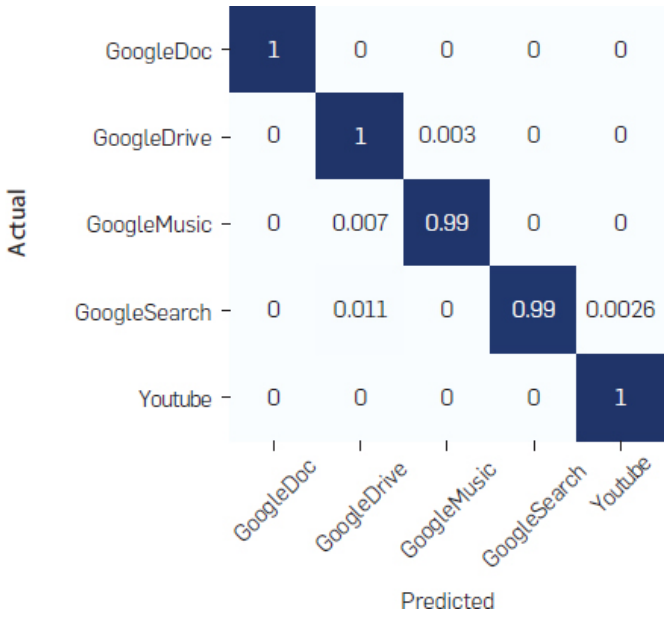


Figure 8. Confusion matrix of our model on the UCDavis QUIC traffic dataset.

Back to Top

5. Conclusion

Traffic classification has become increasingly challenging with the widespread adoption of encryption in the Internet. Moreover, encrypted protocols are bound to evolve, rendering protocol specific approaches futile in the future. In this paper, we propose a DL approach for encrypted traffic classification that focuses on protocol-agnostic aspects of the encrypted Web traffic. Our feature set comprises of a time series of packet size, direction and inter-arrival times, flow statistics, and raw bytes from only the TLS handshake, while the DL model is based on CNN and stacked LSTM layers. We show that raw traffic apart from the TLS handshake does not contribute to the DL model's performance, but rather adds to its complexity and increases overfitting. Therefore, our feature engineering method makes use of concepts that are applicable to most encrypted protocols.

We obfuscate parts of the TLS handshake (for example, SNI field and cipher info) that give away the server identity. Instead, we focus on the traffic shape and timing of packets, which show high potential in learning the complex nature of traffic among different classes. We show that our DL model generalizes for different classification objectives, that is, service- and application-level classification, and adapts to different encrypted Web protocols (such as, HTTP/2 and QUIC) by simply changing the training data. We evaluate our approach for service-level classification on a real-world mobile traffic dataset from an ISP, and show that by leveraging less raw traffic and a smaller number of parameters, our model outclasses a state-of-the-art approach.[11,12]

References

1. Aceto, G., Ciuonzo, D., Montieri, A., Pescapé, A. Mobile encrypted traffic classification using deep learning: Experimental evaluation, lessons learned, and challenges. *IEEE Trans. Netw. Serv. Manag 16*, 2 (2019), 445–458.

2. Anderson, B., Paul, S., McGrew, D. Deciphering malware's use of tls (without decryption). *Springer J. Comput. Virol. Hacking Tech 14*, 3 (2018), 195–211.

3. Blacklists UT1, 2020. http://dsi.ut-capitole.fr/blacklists/index_en.php. [Online; Accessed 01-November-2021].

4. Bonfiglio, D., Mellia, M., Meo, M., Rossi, D., Tofanelli, P. Revealing skype traffic: When randomness plays with you. In *ACM SIGCOMM Comput. Commun. Rev 37* (2007), 37–48.

5. Boutaba, R., Salahuddin, M.A., Limam, N., Ayoubi, S., Shahriar, N., Estrada-Solano, F., Caicedo, O.M. A comprehensive survey on machine learning for networking: Evolution, applications and research opportunities. *Springer J. Internet Serv. Appl 9*, 1 (2018), 16.

6. Brissaud, P.-O., Francçis, J., Chrisment, I., Cholez, T., Bettan, O. Transparent and service-agnostic monitoring of encrypted web traffic. *IEEE Trans. Netw. Serv. Manag 16*, 3 (2019), 842–856.

7. Bronzino, F., Schmitt, P., Ayoubi, S., Martins, G., Teixeira, R., Feamster, N. Inferring streaming video quality from encrypted traffic: Practical models and deployment experience. *ACM SIGMETRICS 3*, 3 (2019), 1–25.

8. Lashkari, A.H., Draper-Gil, G., Mamun, M.S.I., Ghorbani, A.A. Characterization of tor traffic using time based features. In *International Conference on Information Systems Security and Privacy (ICISSP)* (2017), 253–262.

9. Lopez-Martin, M., Carro, B., Sanchez-Esguevillas, A., Lloret, J. Network traffic classifier with convolutional and recurrent neural networks for internet of things. *IEEE Access 5* (2017), 18042–18050

10. Lotfollahi, M., Siavoshani, M. J., Zade, R.S.H., Saberian, M. Deep packet: A novel approach for encrypted traffic classification using deep learning. *Springer Soft Comput 24*, 3 (2020), 1999–2012.

11. Rezaei, S., Kroencke, B., Liu, X. Large-scale mobile app identification using deep learning. *IEEE Access 8* (2019), 348–362.

12. Rezaei, S., Liu, X. How to achieve high classification accuracy with just a few labels: Semi-supervised approach using sampled packets. *arXiv:1812.09761* (2018).

13. Velan, P., Čermák, M., Čeleda, P., Drašar, M. A survey of methods for encrypted traffic classification and analysis. *Int. J. Netw. Manag 25*, 5 (2015), 355–374.

14. Wang, W., Zhu, M., Wang, J., Zeng, X., Yang, Z. End-to-end encrypted traffic classification with one-dimensional convolution neural networks. In *IEEE International Conference on Intelligence and Security Informatics (ISI)* (2017), 43–48.

15. Williams, N., Zander, S., Armitage, G. A preliminary performance comparison of five machine learning algorithms for practical ip traffic flow classification. *ACM SIGCOMM Comput. Commun. Rev 36*, 5 (2006), 5–16.

16. Yao, H., Gao, P., Wang, J., Zhang, P., Jiang, C., Han, Z. Capsule network assisted iot traffic classification mechanism for smart cities. *IEEE IoT J. 6*, 5 (2019), 7515–7525.

17. Zou, Z., Ge, J., Zheng, H., Wu, Y., Han, C., Yao, Z. Encrypted traffic classification with a convolutional long short-term memory neural network. In *IEEE International Conference on High Performance Computing and Communications; IEEE International Conference on Smart City; IEEE International Conference on Data Science and Systems (HPCC/SmartCity/DSS)*, (2018), 329–334.

tnotes

To view the accompanying Technical Perspective, visit doi.acm.org/10.1145/3556640

The original version of this paper is entitled "A Look Behind the Curtain: Traffic Classification in an Increasingly Encrypted Web" and was published in *Proceedings of the ACM Meas. Anal. Comput. Syst.* (Mar. 2021).

About the Authors

Iman Akbari (iakbaria@uwaterloo.ca), University of Waterloo, Canada.

Mohammad A. Salahuddin (mohammad.salahuddin@uwaterloo.ca), University of Waterloo, Canada.

Leni Aniva (aniva@stanford.edu), Stanford University, Stanford, Calif.

Noura Limam (n2limam@uwaterloo.ca), University of Waterloo, Canada.

Raouf Boutaba (rboutaba@uwaterloo.ca), University of Waterloo, Canada.

Bertrand Mathieu (bertrand2.mathieu@orange.com), Orange Labs, France.

Stephanie Moteau (stephanie.moteau@orange.com), Orange Labs, France.

Stephane Tuffin (stephane.tuffin@orange.com), Orange Labs, France.

Join the Discussion (0)