Assessing the Limits of Privacy and Data Usage for Web Browsing Analytics

Daniel Perdices, Jorge E. López de Vergara, Iván González
Dept. Electronics and Communication Technologies, Escuela Politécnica Superior,
Universidad Autónoma de Madrid, Madrid, Spain
{daniel.perdices, jorge.lopez_vergara, ivan.gonzalez}@uam.es

***Abstract*—Web browsing analytics provides insights on the websites that users access, which affects their privacy. Although this analysis might be seen as an easy task, different problems, such as encryption, the tangled web, with several domains visited at the same time in a single web page, or IP addresses of a cloud provider shared by several sites, make it a though job. However, despite these issues, users' privacy is still unaccomplished, as we show in this work. We provide a novel approach that only takes into account the IP addresses that the user has connected to without performing any reverse DNS lookup. We use this sequence of addresses as an input of a neural network, which is able to identify accurately which was the website actually visited among Alexa's World Top 500 most visited domains. Moreover, we have also studied other factors, such as the dependence on the DNS server used to resolve the visited IP addresses, the accuracy for the top domains (e.g., Google, YouTube, Facebook, etc.), data augmentation to improve our results, or the impact on packet sampling. In this last case, we conclude that, using only a 10% of the packets, we can identify the visited website with an accuracy of 93%, whereas it can be over 97% if there is no packet sampling and we use data augmentation.**

***Index Terms*—web browsing analytics, neural network, privacy.**

I. Introduction

In the last years, data has become one of the most valuable assets in the world. It can provide deep customer insights to business makers, which are really interested in knowing the interests of their customers or if they are visiting the competitors web pages. Nowadays, most of the top tech companies exploit such data or even sell it to others, making this a really profitable business. Thus, end users and even academics [1] have been paying attention to this matter since they are the ones who produce data, and they have no real knowledge of how their data is being employed.

Consequently, privacy and data usage have become main concerns of the Internet users. Answering questions such as which data is used by the companies, who they share it with, and how valuable it is; are major issues nowadays. Therefore, users try to protect themselves as much as possible, in particular, they limit the amount of data they share. However, this does not sometimes avoid the data being captured and used by many agents. Solutions such as encryption, both at HTTP level and at DNS level, have become default standards that will cover the majority of the traffic in the next years. Nevertheless, they can only encrypt end-to-end conversations, meaning that IP and TCP or UDP information is still available.

Another popular method used to protect the privacy and avoid unapproved data usage is using Virtual Private Networks (VPNs). Although VPNs have become increasingly popular and most of them may encrypt and tunnel IP traffic, it can still be monitored at the termination point of the VPN. Then, it is just a matter of who you are giving your data to. This means that actors between the VPN server network and the website server can see and use the data. The VPN provider can even go beyond that, since it knows client's identity.

Other alternatives, such as Tor, Brave browser or using chains of proxies suffer from a similar issue, where your identity might not be clear, but your navigation data can be used by the last server in the chain. For many purposes, such as marketing and trends studies, aggregated data is still valuable, so it does not matter whether the identity of the user is known. In fact, we checked that popular Tor-capable browsers such as Tor or Brave use the same endpoint within the web browsing activity of a tab, meaning that these Tor exit nodes can still use, give, and monetize your data. This proved to be more than a possibility, when a significant percentage of Tor exit nodes spied on their users' activity [2] and even use `sslstrip` for HTTP traffic to cryptocurrency exchange websites [3].

In this light, we want to answer the following: Can someone identify where you are navigating through in these cases? As we pointed out, this could be achieved as long as web browsing activity can be inferred from IP traffic. This means that this question is equivalent to: Can it be inferred what website you are visiting with only IP layer information? This is the research question we want to answer, since IP addresses can be easily obtained using NAT logs, Netflow [4] or IPFIX [5] records, or custom monitoring tools.

Although this task is supposedly simple, there are many complications that arise. The main issue is what other authors have defined as the tangled web [6]. When connecting to a website, the web browser has to open a cascade of connections to other websites due to ads, banners, JavaScript libraries, social media links, and many more. It is not only content from third-parties that developers include in their websites, but also mechanisms of the web browser such as prefetching, ad

This work has been partially supported by the Spanish State Research Agency under the project AgileMon (AEI PID2019-104451RB-C21) and by the Spanish Ministry of Science, Innovation and Universities under the program for the training of university lecturers (Grant number: FPU19/05678).

blockers, or caching that may cause trouble. This entails that discerning the web browsing behavior from these connections can be problematic and unstable, since some connections to Facebook or Google servers might appear in other different websites that have nothing to do with them. It is even possible that some connections are opened to prefetch other websites or, on the other hand, they are not opened due to the cache or an ad blocker. Moreover, the websites deployed on content delivery networks and cloud providers make this worse, as a single IP address can be shared by several domains [7], which causes DNS reverse queries to be useless. Furthermore, in our datasets, we observed that, for each pair of web browsing activities, an average of 25% of the IP addresses receiving HTTP/HTTPS connections were present in both traces. Additionally, the mean of the percentage of IP addresses of one domain that overlap with the IP addresses of other domains is around 80%.

In this paper, we present a neural-network-based model that predicts the domain visited using just information of the observed IP addresses. For the sake of the evaluation, we have built a dataset of more than 340 GB of network traffic of automatic web browsing activities through the top 500 domains of Alexa [8]. This model achieves an accuracy higher than 97% on this dataset and proves that the answer to the previous question is affirmative. This allows us to see how many actors can really know about your web browsing behavior and, thus, use your data for any purpose.

The rest of the document follows this outline: section II provides a summary of the current state of the art, focusing of the novelty of this work. Next, section III explains the architecture of our proposal. Section IV benchmarks the performance of the model with several facts in mind, such as the DNS servers, the accuracy for the most visited domains, and the impact of packet sampling. After this, section V comments on the findings and outcomes of this work and section VI concludes the document summarizing the main results of this work.

II. State of the art

Traffic identification [9] has become a popular issue in the last few years. The reasons to do so, however, are quite varied. For instance, authors in [10] focused on monetization, in particular, they used DNS data to generate website fingerprints that can be later used for traffic identification or even traffic generation. Similarly, authors in [11] used a method called Bag of Domains, akin to Bag of Words for text processing. Although the objectives of these papers are quite similar, the data inputs are rather different. It is clear that DNS data is more reliable, since names might be the same over time or among different locations, whereas IP addresses usually change. Furthermore, both authors in [7], [10] tackled the issue of the tangled web that we have explained before, where it is hard to tell when some connections are caused by a particular website. On the other hand, DNS encryption through DNS-over-HTTPS (DoH) [12] is becoming increasingly popular, meaning that the only one that can monetize the data would be the DNS server that receives the encrypted requests. In this case, relying on the IP addresses solves this issue and makes our proposal more future-proof. Similarly, authors in [13] propose a system to determine the traffic generated by a site, but not with classification purposes in mind.

Flow features have also been used to identify web browsing activities. For instance, the work in [14] proposed to use the density estimation of the flow size and binary Bayes belief networks to distinguish between the top 50 most popular websites using anonymized NetFlow logs. This approach, although it is limited to 50 websites, can only detect correctly a 48% of the cases. Furthermore, authors of [15] designed a model to identify domain names using flow-level features such as the size of the second packet, the inter-delay between packets, or the number of packet retransmission. Although this approach may detect servers even if IP addresses have changed, the output of the model is much weaker than our proposal. While we predict the website the users are navigating through, they just predict the domain name of each individual connection.

Machine learning and deep learning have also helped the community to build better proposals. In [16], authors propose a deep learning approach based on Convolutional Neural Networks (CNN) to identify the users web connection. Despite the novelty, the approach is quite focused on the Tor network case, where security and privacy are a priority but data usage do not play a significant role. Also, the adversary that wants that information is in between the user and the input node of the Tor network. This is quite focused on security issues such as preventing users to access illegal sites, whereas we are focused on the other side—i.e. being near the exit node to use the produced data.

Coarse-grained traffic identification and classification is also useful to profile the users or clients of the network or to provide appropriate Quality of Service depending on the type of service—e.g. P2P, video streaming. In [17], a non-supervised algorithm is proposed to classify traffic into different categories. However, they use URLs for this sake, which entails limitations with the increasing presence of encrypted connections. Also, in [18], a collection of supervised algorithms is applied with a similar purpose, with raw captured traffic as input. In this case, the classes—for instance, WWW, telnet, or ftp—are very coarse for many purposes, such as web visit identification for monetization and website banning and filtering systems. Authors in [19] provided simple probabilistic signatures, obtained with first n bits of a flow, that can identify network applications, paying special attention to the amount of data they need for that purpose. Furthermore, the limitation of packet traces as input is a huge limit for many situations, where capturing a large amount of traffic can be overwhelming.

A similar topic that is growing in the community is mobile traffic identification, this is, identify if the traffic was produced by a mobile *app* and which *app* produced the traffic. In [20]–[25], different authors followed a similar approach using neural networks, CNN in particular, to identify the traffic. Authors in [26] provided a detailed overview of this kind

of approaches that use deep learning with encrypted data for mobile traffic identification. In contrast to our approach, they mainly focus on encryption and using raw data captures, which limits the final applicability and scalability to scenarios where full capture is feasible. It is worth noting that our proposal can also be applied to this case of mobile data, but it would require to re-train the model with an appropriate dataset. As it was pointed out in [20], most of the traffic they observed is HTTP or HTTPS and many mobile applications are just web views with native API connectors.

III. Architecture of our proposal

In order to precisely define the model, we must specify the inputs and outputs. The input will be a sequence of IP addresses observed by a monitoring system in connections of a particular client, i.e., $A = [a_1, \ldots, a_M]$, where M is the length of the sequence and each a_i is an IP address. It is clear that, in real conditions, M might depend on several factors. However, we will fix M as a global parameter that trims or pads the sequences to the desired length. The output is the vector of probabilities, this is,

$$\mathbf{P} = [P(D = d|A = [a_1, \ldots, a_M])]_{d \in \mathbb{D}} \tag{1}$$

where $\mathbb{D}$ is the set of all domains we want to identify. From the point of view of parameters, there are two parameters that may impact the model: $|\mathbb{D}|$, the size of $\mathbb{D}$, and M, the length of the sequence. In both cases, the longer number of parameters, the more accurate the model is and, unfortunately, training is also going to take longer.

A. IP embeddings

One of the most important steps to build a neural network model that is fed with IP addresses is to convert the addresses to real-valued numbers or vectors. In fact, either an IPv4 or an IPv6 address are just integers that take values over a huge domain, the upper limits are $U = 2^{32}$ for IPv4 and $U = 2^{128}$ for IPv6. For that sake, we will be using the hashing trick [27] with a linear operator, as we explain below.

First, we need a hash function h that projects the large space $\{0, \ldots, U\}$ to a much smaller space $\{0, \ldots, v\}$, where U is the aforementioned upper limit and v is usually called the vocabulary size, i.e., the number of different elements given by the hash. Additionally, we expect this function to have the usual properties of a hash, such as distributing the original data in the project space with little collisions. Given that $v \ll U$, collisions are unavoidable, however, this is a price we must pay in order to build a model that fits in memory and that can be trained without major issues.

The next step is just as simple as mapping each index to a vector, this is, we train w_i, a vector in $\mathbb{R}^d$, for each $i \in \{0, \ldots, v\}$. This is a categorical embedding

$$\begin{aligned} L : \{0, \ldots, v\} &\to \mathbb{R}^d \\ i &\to L(i) = w_i \end{aligned} \tag{2}$$

where w_i are weights that must be trained. To sum up, both of these functions achieve the objective of projecting an IP address to a continuous domain. In particular,

$$\begin{aligned} \pi : \{0, \ldots, U\} &\to \mathbb{R}^d \\ a &\to \pi(a) = (L \circ h)(a) = L(h(a)) \end{aligned} \tag{3}$$

is a trainable neural network layer with $v \cdot d$ parameters. This means that the number of parameters increases linearly with the vocabulary size, forcing us to find a balance between using a v large enough to achieve the desired accuracy and not excessively large to cope with the constraints in training time and memory usage.

B. Completing the model

Once we have built a layer to project IP addresses to dense vectors, we have a sequence of real-valued vectors. This sequence should be processed by the model to obtain the prediction. In this matter, two approaches can be taken: first, one can use a Long Short-Term Memory (LSTM) layer, which captures the interaction between multiple elements and their order; or, second, one can just ignore the order and combine all the elements of the sequence using a global pooling layer, typically the global average pooling layer.

Choosing one option over the other is just a matter of deciding whether the order of the elements should be part of the model or not. Since packet disorder is quite common and this may result in unreliable results, in this case, LSTM suffers from this issue and thus, we chose to keep the model simple and ignore the order of the elements of the sequence, i.e., using a global average pooling layer. This is, we project the sequence A using the previous layer and take the mean

$$\Pi(A) = \frac{1}{M} \sum_{i=1}^{M} \pi(a_i) \tag{4}$$

and we have now that $\Pi(A)$ is a vector of $\mathbb{R}^d$ that can be used in simple models such as a Multi-Layer Perceptron (MLP), denoted hereinafter with f, to predict the output $\mathbf{P}$ using a soft max activation function. Once the network has produced a prediction $\hat{P}(d|A)$, the predicted domain, $\hat{d}$, can be computed just by taking the $\arg\max$, $\hat{d} = \arg\max_d \hat{P}(d|A)$.

Code for this model implemented using TensorFlow 2 [28], [29] has been made publicly available for reproducibility of the results[1].

Once the model is already defined, we need a particular set of values of the hyperparameters to build an instance of the model. Some of these parameters come from the data, such as the number of domains, $|\mathbb{D}|$, the length of sequences, M, or U. However, other parameters depend on the data but cannot be directly estimated. For them, we performed a grid search with different reasonable values and Table I shows the chosen ones for our dataset, described in the next section. Also, Figure 1 summarizes the whole model in a visual representation.

[1] https://github.com/hpcn-uam/ip-web-analytics/tree/main/code

TABLE I
PARAMETERS OF THE MODEL FOR OUR DATA

Name	Description	Value
M	Length of the sequence	250
U	Total number of IP addresses	2^{32}
h	Hash type	MD5
v	Hash output dimension	60 000
d	Embedding dimension	20
f	Layers of MLP Activation functions	$[75, 150, 250, 350, 400]$ ReLu [30]
Train	Optimizer Loss function	Adam [31] Cat. cross entropy

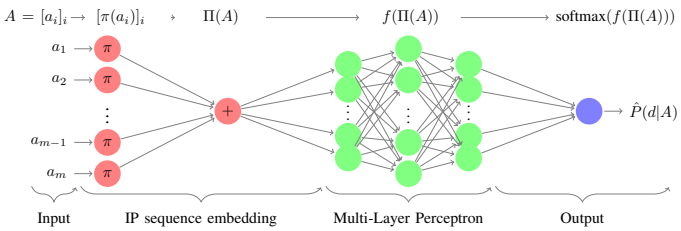


Fig. 1. Visual and mathematical description of the architecture of our model.

IV. EVALUATION

To assess the performance of the model, we have built a dataset consisting of a total of more than 100 000 samples of web browsing of sites in Alexa's World Top 500 most visited domains, performed with five different DNS servers. Although DNS server might seem not important, it is quite relevant since observed IP addresses can highly depend on the DNS resolutions of the domains.

These samples were collected using an automatic system previously developed by authors of [10] that was modified to save capture files. Dataset naming conventions and information about DNS servers are shown in Table II. Also, these datasets are publicly available in aggregated format[2].

We decided to evaluate the data in three different ways: first, we will assess averaged metrics for all the datasets with different DNS servers. Second, we will see the accuracy of our model for the most popular websites and, thirdly, we will cover how packet sampling impacts performance of the model.

A. Effect of name resolution

As we said before, DNS resolution can impact directly the observed IP addresses during the web browsing. Thus, we compare briefly the performance of the model as a function of the data the model has been trained with and the data the model has been tested with.

For this sake, we split each dataset into three pieces: train (65%), validation (15%) and test (20%). We used the train dataset to train all the models, the validation dataset to select the size of the model and control early stopping policies to avoid overfitting, and the test dataset to compute the next results. Figure 2 shows this comparison for the accuracy, i.e., the percentage of correctly predicted domains.

[2]https://github.com/hpcn-uam/ip-web-analytics/tree/main/dataset

TABLE II
DESCRIPTION OF THE DATASETS

Name	Desc.	DNS server	# samples
DNS_1	Campus DNS	150.244.X.Y	25 000
DNS_2	Google	8.8.8.8	25 000
DNS_3	Cloudfare	1.1.1.1	25 000
DNS_4	Quad9	9.9.9.9	25 000
DNS_5	OpenDNS	208.67.222.222	25 000
DNS_{all}	All datasets together	All	125 000

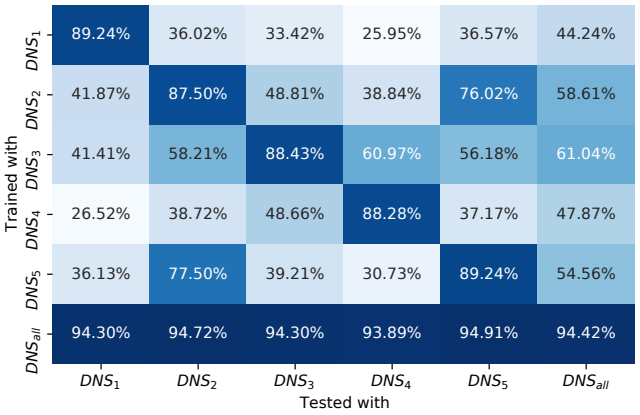


Fig. 2. Accuracy obtained training and testing with each of the different datasets

Looking at the results, we observed that datasets from DNS_1 to DNS_5 have a similar behavior: they score really well when they are tested with samples from the same dataset, but performance drops significantly with other datasets. The only pair that seems to resist this is (DNS_2, DNS_5), i.e., Google DNS server and OpenDNS server.

On the other hand, the DNS_{all} dataset is clearly more consistent, meaning that we must have a wide variety of data to achieve our objective. In order to be compared in fair conditions, all models were trained with the same number of samples, no matter DNS_{all} is five times larger than the rest of the datasets. This means that DNS_{all} was down sampled to have 25 000 samples just for this experiment.

Apart from DNS resolution, other factors such as the web browser—e.g. Google Chrome or Mozilla Firefox—or the device—e.g. mobile phone, tablet or computer—play a significant role. Authors of [10] had already discussed some of these matters with DNS queries in the past, which entails that the same applies to IP addresses.

Given the results, hereinafter we will consider just the model trained with DNS_{all} and we will use 65% of it for training, 15% for validation and model selection and 20% for test.

B. Detailed view of the most important domains

Next, we want to cover the performance of the model for the most popular domains. In real-world datasets, it is very frequent that the most frequent domains are responsible for 90% of the network traffic, the so-called Pareto rule, whereas the 10% remaining is the rest of the web browsing activities. This means that, for many purposes, as long as we are accurate with the top domains, our model will identify successfully a huge amount of the network traffic volume.

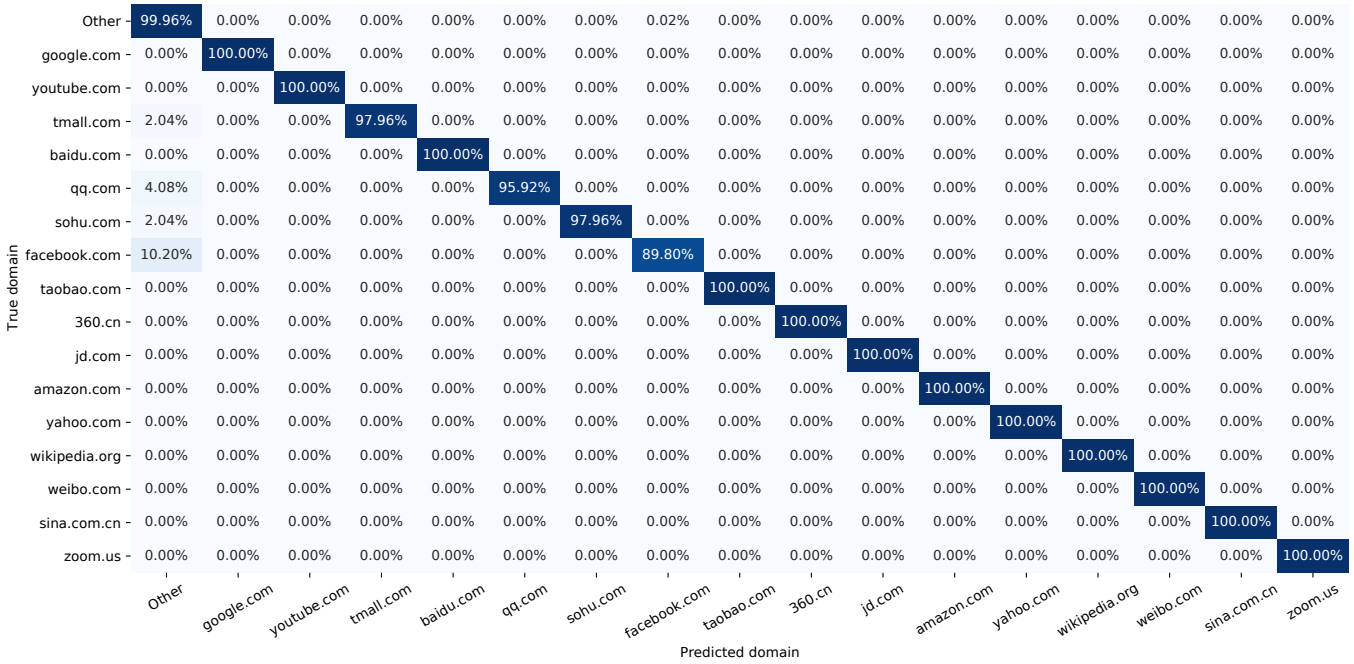
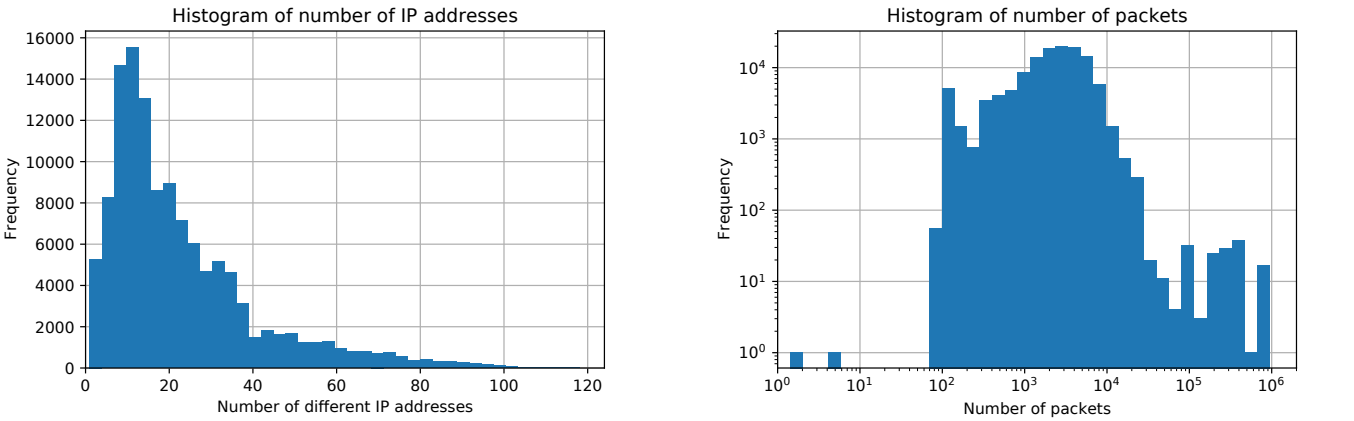


Fig. 3. Confusion matrix for most visited domains in Alexa's World Top 500.



(a) IP addresses.

(b) Packets.

Fig. 4. Histograms of the number of IP addresses and packets observed per web browsing sample

For this sake, Figure 3 shows the confusion matrix for the top domains according to Alexa. A class called "Others" has also been added to represent the rest of the domains. Overall, the performance is outstanding and even perfect for some of the most important domains such as Google, YouTube, or Baidu. It is worth noting that Google and YouTube share some servers, which complicates our task since similar if not the same IP addresses may appear on sequences of both domains.

On the other hand, some domains predictions are less accurate. For instance, Facebook only scores a precision (percentage of correct predictions) of about 90%, meaning that there is a 10% of the times that a user is navigating Facebook, but the model predicts otherwise. This could be to several reasons: similar advertisement presence in other pages, embedded Facebook pages in other domains, or sign-in with Facebook plugins. Similar reasons may apply to Chinese websites, where content or advertising could be shared among different websites. However, precision is over 95%, making the model both accurate and precise.

C. Impact of packet sampling

This last part of the results section covers the impact of packet sampling. First, we want to assess the distributions in our dataset. This provides valuable insights of the dataset that can help us to assess the limits of packet sampling per user, i.e., the minimum number of packets that are required to predict accurately the domain the user is visiting.

Figure 4 displays both the histogram of the number of observed IP addresses and the total number of packets. The right-hand side of the figure depicts the distribution of the number of IP addresses in each sample, i.e., the distribution of the length of the sequence A. In fact, most of the samples

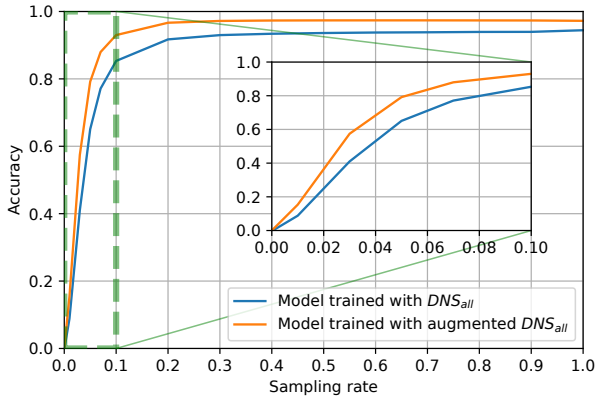


Fig. 5. Accuracy of the model with and without data augmentation against packet sampling rate per user. Zoomed-in version between 0 and 0.1 is included.

have a length around 15, but a significant percentage of the sample are composed of more than 50 different IP addresses.

On the other hand, the left-hand side shows the histogram of number of packets per sample in logarithmic scale for both axes. As we see, most of the samples are around 10^3, i.e., they are composed of hundreds to thousands of packets, while a few of them are composed of more than tens of thousands of packets, even reaching millions in a couple cases.

Since we want to know how the model behaves when we start losing information, we simulated packet sampling and look for solutions to mitigate it. In particular, data augmentation can be applied. It consists of extending the training dataset with additional samples which are just the original ones with small modifications or extra noise. In computer vision, it is quite useful to train the model to ignore scale or orientation.

Let $\mathcal{P} = [P_1, \ldots, P_M]$ be the vector of frequencies associated to each IP address, this is, a_1 has been observed in P_1 packets. It is simple to convert $\mathcal{P}$ into a vector of probabilities $\mathbb{P} = \frac{\mathcal{P}}{\sum_{i=1}^{M} P_i}$, which will be used for the packet sampling. Then, we can just sample from A using this distribution. If we call the frequencies of the sample $\mathcal{P}' = [P'_1, \ldots, P'_M]$, it is direct how to compute the augmented sample by just taking the elements of A whose element in P' is greater than zero—i.e., the elements that appeared in the sample: $A' = [a_i \;\; \forall a_i \in A : P'_i > 0]$.

Once we have A', we can add it to our extended dataset and train the model. Figure 5 shows the results for both the DNS_{all} dataset and the augmented version of DNS_{all} against the sampling rate of the test subset. For each sample of the dataset, we have added five extra samples for training. Each sample contains only 20% of the original information, i.e., we have used a sampling rate $r = 0.2$.

For the simple case, there is a slight decay of the non-augmented model with sampling rates from 0.2 to 1, followed by a more noticeable reduction between 0.2 and 0.1, ending in a complete drop at around 0.05. For the case of the augmented model, we see that it even improves with no sampling rate ($r = 1$) and the performance is always better than for the non-augmented model. Besides, zoomed-in subfigure shows the accuracy decay between 0 and 0.1. It becomes apparent that for sampling rates up to 5%, augmented model performance is around 80%. Since we have observed that the mode of the number of packets per sample is around 3 000 packets, it means that we need around 150 packets to have representatives samples that our model predicts with an accuracy of 80%. Besides, if we had 300 packets, accuracy would improve to more than 93%.

V. Discussion

During this whole work, we have observed several contributions and remarkable ideas of paramount importance for web browsing privacy:

1) ***Web browsing can be inferred from IP addresses***: this was the main objective of this work and the model achieved it with accuracy higher than 90%. It is worth mentioning that data plays a key role, so updated and varied data are required; as we observed with DNS servers. As long as the training data is representative of the users' behavior, web browsing activities can be easily inferred. This proves that, as long as an actor has data about where you are connecting to, web traffic identification can be easily achieved.

2) ***Pay attention to the most popular websites***: because dataset imbalance can be problematic for training, it is useful to use a balanced dataset. However, top domains must be taken into consideration separately, since they are responsible for the majority of the traffic. It does not matter if the model is accurate at 99% if it does not work for these domains that represent, in most cases, 90% of the traffic. Providing a detailed analysis is advisable to know better how the model works and which popular domains can be confused.

3) ***Data augmentation and sampling improve the model***: as we mentioned before, data augmentation is a powerful way of extending training datasets and teaching the model not to pay attention to some facts. In this case, we simulate traffic sampling so that we generate new samples to be incorporated into the training set. It makes the model more resilient to sampling and accurate in all cases, since it allows the learning to be focused on features that are more likely to happen, even under packet sampling. We also did study how many packets the model would need to provide an accurate prediction, which specifies when the model result is trustworthy.

However, there are few drawbacks about this approach. Mainly, the necessity of re-training periodically—due to the presence of dynamic addresses that may change over time—and the necessity of varied datasets—for instance, different DNS servers, or user agents.

VI. Conclusion

Along this work, we have presented a model that effectively identifies web browsing activities just using IP addresses. As we motivated, this means that there are potentially many actors that are able to use your data without your consent. In fact, even some of the best alternatives presumably to protect your data and your privacy cannot totally bypass this, which means that we must be aware of this possibility. The results proved

indeed that performance was promising enough to be deployed and to use it to gather web browsing data from users.

In order to overcome this problem and avoid traffic identification, we foresee several solutions: If the user navigates through Tor or a proxy chain, the browser should use a different exit node for each connection. In this way, it becomes tougher to guess which website the user is visiting. On the server side, servers should change IP addresses every short period of time so that it becomes nearly impossible to train the model before the IP address changes again. Finally, users could also take advantage of noise, this is, generate fake web browsing activities so that data is less valuable because actors have to distinguish between real and fake.

Yet, there are still some open future lines of work. First, we plan to study the lifespan—validity over time—of the datasets. One of the issues that potentially may face these approaches is to determine when the training data is very old to be representative of the real-world data. For DNS domains web fingerprints, some authors [10] have determined that lifespan of web fingerprints is around a week, meaning that we expect this lifespan to be at most a week. Besides, we intend to analyze all the possible data sources, especially, we would like to see how SDN can enrich this model. In particular, we would like to see how network equipment can be programmed to send us the first packet of each connection, so that data reduction is optimal, and it is horizontally scalable. Finally, we also plan to apply this methodology to mobile apps, in order to identify which applications are used in a network based on their interactions at the IP level.

REFERENCES

[1] I. Castell-Uroz, T. Poissonnier, P. Manneback, and P. Barlet-Ros, "URL-based Web Tracking Detection Using Deep Learning," in *2020 16th International Conference on Network and Service Management (CNSM)*. IEEE, 11 2020.

[2] The Hacker News, "Over 25% Of Tor Exit Relays Spied On Users' Dark Web Activities," https://thehackernews.com/2021/05/over-25-of-tor-exit-relays-are-spying.html, accessed: 2021-05-19.

[3] The Tor Project, "Tor security advisory: exit relays running sslstrip in May and June 2020," https://blog.torproject.org/bad-exit-relays-may-june-2020, accessed: 2021-05-19.

[4] B. Claise, "Cisco Systems NetFlow Services Export Version 9," RFC 3954, 2004.

[5] P. Aitken, B. Claise, and B. Trammell, "Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of Flow Information," RFC 7011, Sep. 2013.

[6] X. Hu and N. Sastry, "What a Tangled Web We Weave: Understanding the Interconnectedness of the Third Party Cookie Ecosystem," in *WebSci 2020 - Proceedings of the 12th ACM Conference on Web Science*. Association for Computing Machinery, Inc, 7 2020, pp. 76–85.

[7] I. N. Bermudez, M. Mellia, M. M. Munafò, R. Keralapura, and A. Nucci, "DNS to the Rescue: Discerning Content and Services in a Tangled Web," in *Proceedings of the 2012 ACM conference on Internet measurement conference - IMC '12*. New York, New York, USA: ACM Press, 2012.

[8] Amazon Web Services, "Alexa - Top sites," 2020.

[9] A. Callado, C. Kamienski, G. Szabó, B. P. Gero, J. Kelner, S. Fernandes, and D. Sadok, "A survey on internet traffic identification," *IEEE Communications Surveys and Tutorials*, vol. 11, no. 3, pp. 37–52, 2009.

[10] J. L. García-Dorado, J. Ramos, M. Rodríguez, and J. Aracil, "DNS weighted footprints for web browsing analytics," *Journal of Network and Computer Applications*, vol. 111, pp. 35–48, Jun. 2018.

[11] M. Trevisan, I. Drago, M. Mellia, and M. M. Munafo, "Towards web service classification using addresses and DNS," in *2016 International Wireless Communications and Mobile Computing Conference, IWCMC 2016*. IEEE, Sep. 2016, pp. 38–43.

[12] P. E. Hoffman and P. McManus, "DNS Queries over HTTPS (DoH)," RFC 8484, Oct. 2018.

[13] M. Trevisan, I. Drago, M. Mellia, H. H. Song, and M. Baldi, "WHAT: A big data approach for accounting of modern web services," in *Proceedings of the 2016 IEEE International Conference on Big Data, IEEE Big Data 2016*. IEEE, 2016, pp. 2740–2745.

[14] S. E. Coull, M. P. Collins, C. V. Wright, F. Monrose, and M. K. Reiter, "On web browsing privacy in anonymized netflows." in *USENIX Security Symposium*, 2007, pp. 339–352.

[15] M. Trevisan, F. Soro, M. Mellia, I. Drago, and R. Morla, "Does domain name encryption increase users' privacy?" *SIGCOMM Comput. Commun. Rev.*, vol. 50, no. 3, p. 16–22, Jul. 2020.

[16] S. Bhat, D. Lu, A. Kwon, and S. Devadas, "Var-CNN: A Data-Efficient Website Fingerprinting Attack Based on Deep Learning," *Proceedings on Privacy Enhancing Technologies*, no. 4, pp. 292–310, 2019.

[17] A. Morichetta and M. Mellia, "LENTA: Longitudinal exploration for network traffic analysis from passive data," *IEEE Transactions on Network and Service Management*, vol. 16, no. 3, pp. 814–827, 2019.

[18] M. Shafiq, X. Yu, A. A. Laghari, L. Yao, N. K. Karn, and F. Abdessamia, "Network Traffic Classification techniques and comparative analysis using Machine Learning algorithms," in *Proceedings of the 2016 2nd IEEE International Conference on Computer and Communications, ICCC 2016 - Proceedings*. IEEE, 5 2017, pp. 2451–2455.

[19] N. Hubballi, M. Swarnkar, and M. Conti, "BitProb: Probabilistic Bit Signatures for Accurate Application Identification," *IEEE Tran. Network and Service Management*, vol. 17, no. 3, pp. 1730–1741, Sep. 2020.

[20] S. Rezaei, B. Kroencke, and X. Liu, "Large-Scale Mobile App Identification Using Deep Learning," *IEEE Access*, vol. 8, pp. 348–362, 2020.

[21] X. Wang, S. Chen, and J. Su, "Automatic mobile app identification from encrypted traffic with hybrid neural networks," *IEEE Access*, vol. 8, pp. 182 065–182 077, 2020.

[22] T. Shapira and Y. Shavitt, "FlowPic: Encrypted Internet Traffic Classification is as Easy as Image Recognition," in *Proceedings of the 2019 IEEE Conference on Computer Communications Workshops, INFOCOM WKSHPS 2019*. IEEE, 4 2019, pp. 680–687.

[23] M. Wang, K. Zheng, D. Luo, Y. Yang, and X. Wang, "An Encrypted Traffic Classification Framework Based on Convolutional Neural Networks and Stacked Autoencoders," in *Proceedings of the 2020 IEEE 6th International Conference on Computer and Communications, ICCC 2020*. IEEE, 12 2020, pp. 634–641.

[24] B. Sun, W. Yang, M. Yan, D. Wu, Y. Zhu, and Z. Bai, "An Encrypted Traffic Classification Method Combining Graph Convolutional Network and Autoencoder," in *Proceedings of the 2020 IEEE 39th International Performance Computing and Communications Conference, IPCCC 2020*. IEEE, 11 2020.

[25] R. Moreira, L. F. Rodrigues, P. F. Rosa, R. L. Aguiar, and F. D. O. Silva, "Packet Vision: A convolutional neural network approach for network traffic classification," in *Proceedings of the 2020 33rd SIBGRAPI Conference on Graphics, Patterns and Images, SIBGRAPI 2020*. IEEE, 11 2020, pp. 256–263.

[26] G. Aceto, D. Ciuonzo, A. Montieri, and A. Pescapé, "Mobile encrypted traffic classification using deep learning: Experimental evaluation, lessons learned, and challenges," *IEEE Transactions on Network and Service Management*, vol. 16, no. 2, pp. 445–458, Jun. 2019.

[27] C. B. Freksen, L. Kamma, and K. Green Larsen, "Fully understanding the hashing trick," in *Advances in Neural Information Processing Systems*, vol. 31. Curran Associates, Inc., 2018.

[28] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard *et al.*, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org. [Online]. Available: https://www.tensorflow.org/

[29] F. Chollet, "Keras," 2015.

[30] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, J. Fürnkranz and T. Joachims, Eds., 2010, pp. 807–814.

[31] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," in *Proceedings of the 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track*, 2015.