X-Ray Goggles for the ISP: Improving in-Network Web and App QoE Monitoring with Deep Learning

Pedro Casas, Sarah Wassermann, Michael Seufert, Nikolas Wehner, Olivia Dinica, Tobias Hossfeld

X-Ray Goggles for the ISP: Improving in-Network Web and App QoE Monitoring with Deep Learning

Pedro Casas*, Sarah Wassermann*, Michael Seufert†, Nikolas Wehner†, Olivia Dinica*, Tobias Hoßfeld†
*AIT Austrian Institute of Technology, †University of Würzburg

***Abstract*—The wide adoption of end-to-end encryption is drastically limiting the visibility Internet Service Providers (ISPs) have on the performance of the services consumed by their customers. We present *DeepQoE*, a deep-learning based approach to infer the Quality of Experience (QoE) of web services and mobile applications from the ISP perspective, relying exclusively on the analysis of encrypted network traffic. Using raw features derived from the encrypted stream of bytes as input to deep Convolutional Neural Networks (CNNs), *DeepQoE* infers the Speed Index of web browsing sessions and general mobile apps with unprecedented accuracy, improving the state of the art absolute inference error by more than 25%, and reducing the QoE inference error in terms of mean opinion scores by nearly 40%. *DeepQoE* implements a web fingerprinting solution to identify individual web browsing sessions within concurrent web pages traffic, enabling highly detailed, per webpage QoE inference in practical deployments. Extensive evaluation across a large measurement dataset of popular webpages and mobile apps confirms the out-performance and generalization of *DeepQoE* to different devices, web pages, apps, and network setups.**

***Index Terms*—Deep Learning, Quality of Experience, Web Browsing, Mobile Apps, Network Monitoring, Traffic Encryption.**

I. Introduction

Quality of Experience (QoE) monitoring is a daunting yet critical task for Internet Service Providers (ISPs), who need to shed light on the performance of their networks as perceived by their customers, to avoid churn due to quality dissatisfaction. Web browsing QoE has attracted signification attention in recent years. While ISPs have traditionally relied on the usage of Deep Packet Inspection (DPI) techniques to understand the performance of web services from the network side, the wide adoption of end-to-end traffic encryption has drastically reduced their visibility. This has motivated a surge in the research and conception of Machine-Learning (ML) based approaches to infer application-level Web QoE metrics from the streams of encrypted bytes [3], [4], [6]. In these previous work, standard shallow-learning models have been used in the task. We take a step further into this direction, by conceiving approaches based on novel, Deep-Learning (DL) architectures, which provide better and deeper visibility into the QoE of web services. We conceive *DeepQoE*, a deep-learning based approach to infer QoE metrics capturing the user-perceived performance of web browsing and general web services accessed through apps, relying exclusively on the analysis of raw features, derived from the encrypted network traffic. *DeepQoE* targets the well known Speed Index (SI) metric as QoE proxy, and integrates novel approaches addressing the automatic identification and filtering of individual web pages and apps traffic flows, further improving the granularity of the monitoring, at the level of individual webpage load sessions. In the context of this study, a session corresponds to all the packets exchanged between the end-user's device IP and the IPs of the servers hosting the webpage embedded contents, for one specific webpage loading instance. Fig.1 depicts the end-to-end workflow of *DeepQoE*. It consists of three specific modules covering (A) the traffic identification and filtering step – which isolates the traffic belonging to each web browsing and app session, (B) the feature extraction from the isolated (encrypted) traffic, and (C) the DL-based SI inference. *DeepQoE* tackles important challenges partially addressed before in the state of the art, including: (i) the **automatic identification of web pages and apps** in the (encrypted) traffic streams, (ii) the **disentangling of concurrent web browsing and app sessions**, and (iii) the conception of **multi-device and multi-service DL-based models for Web and App QoE inference**. Previous work has also taken a similar direction for web traffic disentangling and Web QoE monitoring, introducing the PAIN indicator and Web performance monitoring system [1]. While *DeepQoE* has multiple similarities with PAIN, it also has important differences: to start with, *Deep-QoE* can directly infer the SI of a browser session – which is a direct proxy to Web QoE [7], whereas PAIN only approximates the SI; *Deep-QoE* traffic identification and disentangling are fully-supervised approaches, based on off-line active measurements and fingerprinting, whereas PAIN is meant for unsupervised operation, and requires passive measurements for webpages accessed by a large number of users. As stated in [1], PAIN cannot monitor the performance of a single visit to a specific webpage, whereas that is exactly the target of *Deep-QoE*. Finally, *DeepQoE* uses a double fingerprinting approach to identify webpages directly requested by the user, instead of simply tracking pre-defined webpage names defined by the operator. In this paper we separately evaluate the identification of websites, the disentangling of webpages, and the SI inference, which provides a more comprehensive overview on the functioning and performance of *Deep-QoE* as compared to PAIN. Finally, we also apply *Deep-QoE* to mobile web browsing and mobile apps, which is not done for PAIN.

Through extensive evaluation, we verify the out-performance of *DeepQoE* as compared to shallow-learning-based solutions. In particular, we evaluate *DeepQoE* using controlled measurements on: (i) mobile and desktop web

978-3-903176-47-8

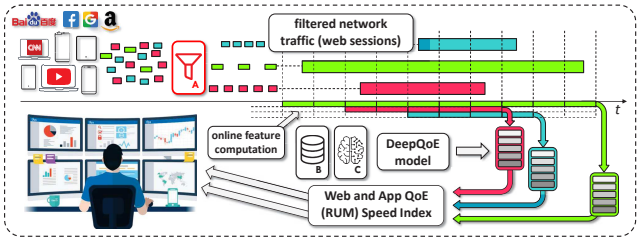


Fig. 1: *DeepQoE* monitoring workflow.

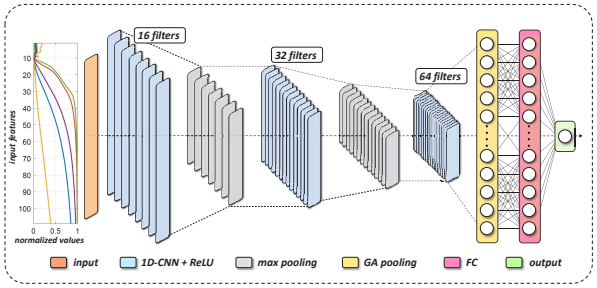


Fig. 2: *DeepQoE* deep learning architecture.

browsing, targeting the top-500 most popular websites in today's Internet, (ii) web services on popular native and web-based apps, and (iii) multiple network setups in terms of latency, packet-loss, and bandwidth. *DeepQoE* improves the state of the art by more than 25% with respect to SI inference, additionally reducing the QoE inference error in terms of Mean Opinion Scores (MOS) by nearly 40%. Evaluations also show the fine-granularity monitoring capabilities enabled by the conceived traffic identification approaches, being able to detect single web pages and to isolate all the traffic flows belonging to these with precision and recall above 95% (detection) and 80% (disentangling), for up to ten concurrent web pages visited by the same IP address at the same time.

The remainder of the paper is organized as follows. Sec. II briefly overviews related work. Sec. III presents the principles, raw features, and deep architecture behind *DeepQoE*, and describes the different datasets used for model training and evaluation. Sec. IV presents the techniques employed by *DeepQoE* for web traffic identification and filtering, analyzing their relevance in the corresponding web datasets. Extensive evaluation results are reported in Sec. V, including the inference of web and app SI and MOS scores, the detection and disentangling of web pages, and the evaluation of the end-to-end monitoring solution. Finally, Sec. VI concludes this work.

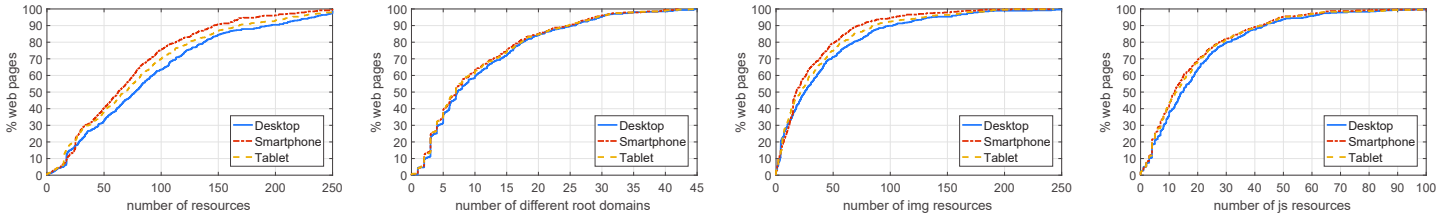
II. Related Work

There is a vast literature in Web QoE measurement and analysis [2], [7]–[10]; however, previous work have mostly focused on measurements at the application layer or assuming access to end devices, which is not applicable for ISPs to perform network-wide QoE monitoring. ISPs require approaches which can operate directly at the network traffic level, where massive data is available to their already deployed monitoring systems, yet with the additional complexity introduced by the wide adoption of TLS/HTTPS for end-to-end traffic encryption. Previous work [1], [2] have developed Web QoE-related metrics highly correlated to the SI metric, including the Byte and Object-Index [2] and the PAIN index [1], which can be computed directly from packet and flow level measurements, seamlessly operating with encrypted traffic. Still, such metrics are mostly informative, as they do not provide an absolute estimation of the actual user QoE. The SI metric is today widely accepted as one of the best metrics serving as proxy to Web QoE [7], as it takes into account the visual progress of the page loading. Recent work [3], [4], [6] have tackled the problem by conceiving (shallow) ML-driven models to directly infer the SI of web loading sessions, using inputs directly derived from the encrypted streams of traffic. These papers are the closest to our work, but have limited applicability in practice, as they only consider the idealized case of single web session analysis. As we confirm in our results, the application of deep learning models to the SI inference problem can better track the loading page phenomenon, significantly improving the inference accuracy, and therefore highly improving the visibility for the ISP. While the application of DL to the same inference task has been recently studied [5], results obtained with the proposed input features and network architecture (VGG16 [18]) provide limited improvements of just a few tens of milliseconds as compared to shallow learning models. *DeepQoE* reduces absolute inference errors by more than 25%, resulting in significant QoE performance gains. Explainability analysis through SHAP [15] suggests that session metrics such as total duration provide the most descriptive information behind such an improved accuracy, not used in [5].

III. *DeepQoE* Model and Datasets

A. Features and Deep Model

The first step in the *DeepQoE* workflow corresponds to the traffic identification and filtering (cf. Fig. 1). In a nutshell, we identify a session and its corresponding flows through DNS-based IP-addresses-to-domain-names mappings and static fingerprinting. In Sec. IV we present the specific solution to this problem. Once a session has been identified, different raw traffic features are computed on the corresponding traffic flows. *DeepQoE* considers two types of raw features: *global session features*, capturing the duration and volume of the identified set of flows (duration of the session, number of packets and number of bytes, separately computed for downlink and uplink), and *temporal progressive features*, capturing the progressive webpage download process, basically counting the fraction of downloaded bytes per a pre-defined unit of time (e.g., every 100 ms). The specific feature extraction process is out of the scope of this paper, but these features can be easily computed in an on-line manner; regarding scalability of the extraction process, *DeepQoE* is meant to be deployed in vantage points closer to the end-users (e.g., home routers or aggregation gateways) and not at core links. A total of **109 raw features** are extracted from the ongoing encrypted stream of bytes, including nine global session features, and 100 temporal progressive features. These progressive features are defined as the Cumulative Downloaded Bytes $CDB(i)_{\Delta T}$, and correspond to the (normalized) cumulative number of bytes



(a) Resources. (b) Root domains. (c) Images. (d) Java-script.

Fig. 3: Web pages characterization, per device type, including number of resources, number of root domains, and specific image and java-script contents.

dataset	# samples	browser/OS	# pages/app-actions
Web-Desk	25,000	chrome	alexa top-500
Web-Smart	25,000	chrome+android	alexa top-500
Web-Tab	25,000	chrome+android	alexa top-500
APP	13,500	android apps native/web/hybrid	4 apps/11 actions
Total	88,500		

TABLE I: Heterogeneous datasets for model training and evaluation.

QoS feature	EMU setup	QoS feature	EMU setup
–	no shaping	*latency (RTT)*	10, 100, 200, 400 ms
bandwidth	2, 5, 10 Mbps	*packet loss*	0.1, 0.5, 1, 2, 10 %

TABLE II: Network EMU QoS configurations.

downloaded from the first collected byte at time t_0 up to time $t = t_0 + i \times \Delta T$, with $i = 1, \ldots, m$, where m represents the total number of slots (measurements). The CDB features track the download progress of the page bytes, using a time resolution ΔT. We take $\Delta T = 100$ ms and $m = 100$ for *DeepQoE*.

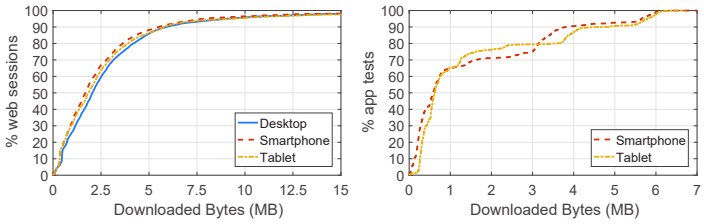
Fig. 2 describes the underlying deep architecture used by *DeepQoE*. While we want to make use of a sufficiently expressive architecture to learn useful representations out of the input data, we do not have a sufficiently large dataset (about 90,000 samples) to train very deep models with lots of parameters, and therefore decided for a simpler version of traditional deep models based on CNNs [16]–[18]. *DeepQoE*'s representation learning architecture is composed of a series of three consecutive 1D-CNN convolutional layers with 16, 32, and 64 filters respectively – using ReLU activation [19] – and intermediate max pooling layers – to compress the size of the representations. The last convolutional layer is connected to a Global Average (GA) pooling layer, which adds additional robustness against over-fitting. The regression stage is composed of a series of standard fully connected layers. The architecture additionally considers batch normalization and dropout layers, to regularize the model. We have tried deeper architectures without significant enhancement. The number and size of the filters were decided by standard grid search.

B. Datasets Characterization

We use a measurement testbed to generate fully-labeled datasets, relying on multiple private instances of WebPageTest (WPT) [20] – the default web-performance-analysis tool – and Appium-based controllers [21] for app testing – see [6] for further details on the platform. Network emulation (EMU) and monitoring capabilities enable the realization of performance and QoE degradation in the monitored traffic. Three different, non-emulated (i.e., real) types of devices are used in the testbed, including smartphones, tablets, and desktop laptops (Chrome is used as browser), using WPT agents for Android and Linux. Tab. I summarizes the four different datasets for web and app *DeepQoE* model training and validation purposes. The three web browsing datasets correspond to individual webpage loading sessions targeting the top 500 websites according to the well-known Alexa top-sites list (updated to Nov. 2020), and using desktop (*Web-Desk*), smartphone (*Web-Smart*), and tablet devices (*Web-Tab*). In the case of web browsing, we collect the so-called RUM Speed Index (RUMSI) metric [22], which is a passive approximation to the SI, computed from the analysis of webpage resource timings.

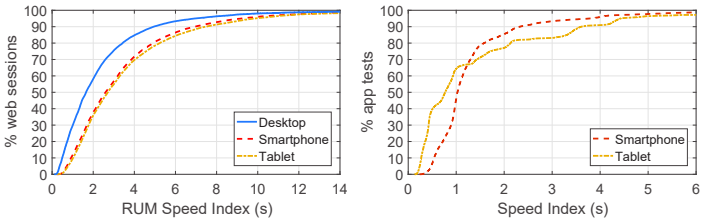
The Appium-based framework enables the analysis of independent *user interactions* with a particular app, generating for each a separate traffic capture and a screen capture from which the SI is computed [23]. Given that app measurement requires instrumentation for each specific app [11], we selected four popular apps for the study, including Amazon, YouTube, Facebook, and BBC News. The idea is to test different app technologies – e.g., Facebook and YouTube are both native apps, BBC News is a web app, and Amazon is a hybrid app, with different levels of interactivity in terms of user actions. We tested different kinds of interaction, including app startup (i.e., start the app, and wait for the main page to load), page scrolling, search, and menu items/links clicking. Note that no user accounts were used in the app tests. From these four apps, the only one requiring a user login is Facebook, so the only tested user-action for this app was the app start-up. The total data includes traffic captures for close to 90,000 individual webpage loading sessions and app interactions. Tab. II details the different network QoS configurations set at the EMU to induce Web QoE heterogeneity in the datasets.

Fig. 3 characterizes the 500 web pages in terms of contents, per device type. Their richness and complexity is reflected by the (a) number of embedded contents and (b) their location at different root domains, with more than 30% of the web pages consisting of more than 100 resources, and about 40% of the web pages fetching resources from more than 10 different root domains. (c) Images and (d) java-script contents make the most of the bytes in modern web pages, with 30% of the pages having more than 50 embedded image resources and more than 20 java-script ones. Fig. 4 depicts the distribution of downloaded bytes for (a) web pages and (b) app user-actions. Desktop-browsed web pages are bigger than those browsed



(a) Web Pages. (b) Apps.

Fig. 4: Downloaded bytes for web pages and app user-actions.



(a) (RUM)SI for web sessions. (b) SI for app actions.

Fig. 5: (RUM)SI for web sessions and app actions.

on smartphones or tablets, which are optimized for smaller screen sizes. The average page size is 2.7MB in desktop, 2.4MB in tablet, and 2.1MB in smartphone. Downloaded bytes for apps are significantly less for the instrumented user-actions as compared to the tested web pages; however, contents here are different, so a direct comparison is out of scope. Regarding performance, Fig. 5 depicts the distribution of (RUM)SI values for web sessions and app actions. (RUM)SI values are significantly higher for both smartphone and tablet devices as compared to desktop, pointing to a more complex rendering process in mobile devices, as well as to the specific hardware limitations of smartphones and tablets [12]. SI values for the specific instrumented apps are much lower, but again, they are not comparable to the values observed in web pages, as contents significantly differ.

IV. Webpage Identification and Filtering

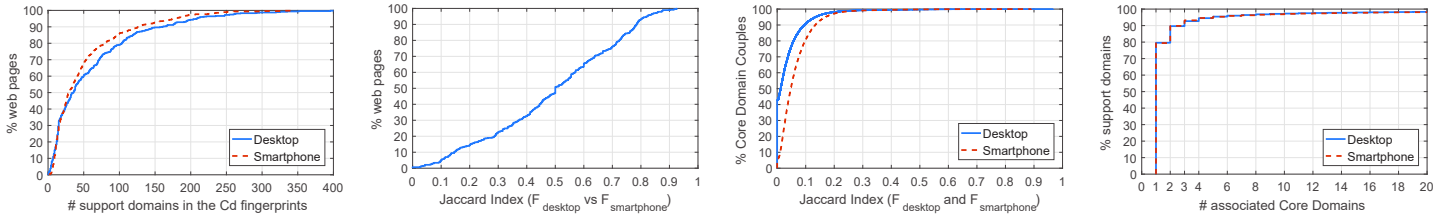
In Sec. III we assumed that the monitoring system takes as input network traffic from single webpage or app loading sessions for subsequent feature computation. However, in practice, sessions are concurrent, originating either at individual (e.g., multi-tab browsing) or multiple users, which might even share the same origin IP address if located behind a NAT gateway. Once a user visits a certain website, the browser opens multiple connections to the different servers, fetching all the contents which compose the corresponding webpage, including both local web contents as well as third-party embedded contents, such as advertisements and more. We refer to the domain associated with the webpage requested by the user as the *core domain* C_d, and the subsequently contacted domains as *support domains* S_{d_j}. Next, we introduce an approach to detect and isolate all the flows corresponding to an individual session. The solution tackles two specific challenges: (i) firstly, it is able to identify all the core domains C_d^i present in a group of mixed web traffic flows, and (ii) secondly, given a core domain C_d^i, it can identify all the support domains $S_{d_j}^i$ associated with it. We refer to the first problem as the *click-stream identification* – i.e., identifying web pages directly requested by the user – and to the second problem as *session disentangling* – i.e., splitting flows into their associated sessions. An additional complexity in these two problems is that a certain domain could be support for multiple different core domains, and that a core domain itself could actually be a support domain for a different webpage. Note that *DeepQoE* assumes traffic flows are tagged with their associated domain names, as observed in the DNS queries, which can additionally be used to build dynamic domain-to-flow mappings [14]. In such a scenario, correctly identifying core and support domains of a web session directly translates into a matching reconstruction of the network traffic flows. In this paper we do not address the usage of DNS encryption.

A. Click-stream Identification

We tackle the click-stream identification problem through a fully supervised, signatures-based approach, where signatures are derived from the sequence of core and associated support domains observed in multiple webpage loading sessions. Given a collection of m web pages – accessible through their core domains $C_d^i, i = 1..m$ – each visited (i.e., measured) n times during a signature-training period, we construct two types of signatures: (i) a core domain signature for each C_d^i, and (ii) a support domain signature for each $S_{d_j}^i$. A C_d^i signature corresponds to a list of all the associated $S_{d_j}^i$ domains observed during the n webpage loading sessions, along with an associated frequency of occurrence, reflecting the percentage of loading sessions where each $S_{d_j}^i$ is associated to the corresponding C_d^i. Each of these frequency values ranges from 0+ to 1, where 1 means that this $S_{d_j}^i$ is always present in loading sessions for webpage C_d^i, and 0+ that it appears occasionally. On the other hand, a signature for $S_{d_j}^i$ corresponds to a list of all C_d^i which this support domain is associated to through the complete set of $m \times n$ measurements, along with the corresponding frequency of occurrence of each C_d^i; in this case, frequency values represent the percentage of loading sessions each core domain is associated to this support domain, within all the loading sessions where $S_{d_j}^i$ appears as a support domain. Each frequency value ranges from 0+ to 1, but the sum of all values is always equal to one this time.

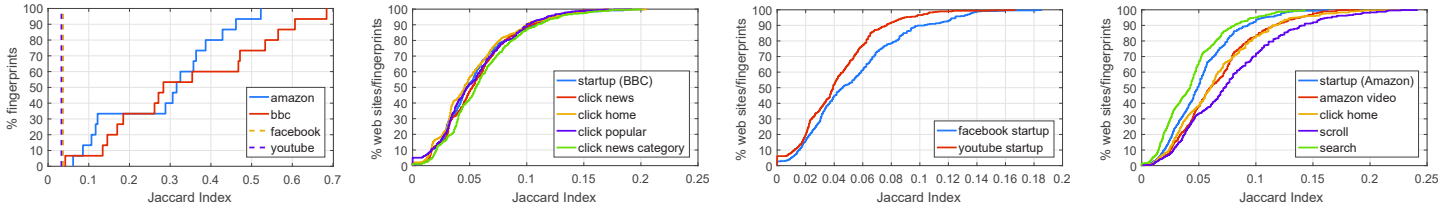
Let us consider webpage C_d = *instagram.com* as example. A potential signature for C_d could be the tuple $\{DC_d, FC_d\}$, where DC_d = {*connect.facebook.net, scontent.xx.fbcdn.net, s.10.facebook.com, static.facebook.com, www.facebook.com, www.instagram.com*} and FC_d = {1, 0.014, 0.014, 1, 1, 1}. For S_d = *outlook.live.com* we could have the tuple $\{DS_d, FS_d\}$, with DS_d = {*live.com*} and FS_d = {1}.

Using signatures for both core and support domains, we conceived four different click-stream identification approaches. In a nutshell, signatures are intersected with the sequence of observed domains in the traffic, and the associated frequencies are taken into account to rank potential core domains. The rationale is straightforward: regarding C_d



(a) C_d signatures. (b) Signatures overlap wrt device type. (c) C_d signatures overlap. (d) S_d signatures.

Fig. 6: Analysis of Web signatures in desktop and mobile browsing. High discriminative power is obtained through C_d and S_d signatures.



(a) Actions signatures vs. Web Pages. (b) BBC. (c) YT/FB. (d) Amazon.

Fig. 7: Analysis of App signatures. App user-action signatures have a small overlap with the top-500 Alexa web pages' signatures.

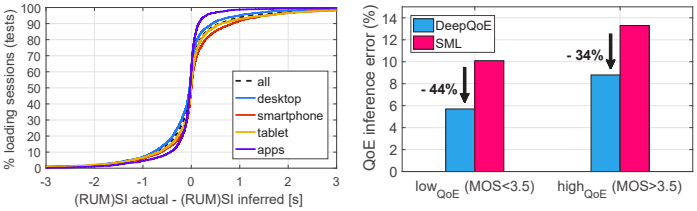
signatures, given a certain webpage with core domain C_d^i, we expect a higher overlap with the list of support domains associated to its signature DC_d^i; regarding S_d signatures, if a higher number of support domains have a common C_d associated, then this C_d is likely a real core domain. **Core and Support Frequency (CSF)** is the most complete approach and represents the selected solution, considering both C_d and S_d signatures. Firstly, all potential C_d are selected, based on the list of known core domains for which signatures were constructed; in our study, this corresponds to the top 500 Alexa websites core domains. A score s_{C_d} is computed for each of these potential C_d, based on the frequencies associated to the overlapping S_d between the analyzed domains and the domains in the corresponding fingerprint for C_d, DC_d. In addition, using the S_d signatures, potential C_d are also ranked by their association to the list of analyzed S_d, computing a second score s_{S_d}. The final selection of C_d consists of the intersection between the set of C_d which have a score s_{C_d} higher than a certain threshold Th_{C_d} (in the solution we take $Th_{C_d} = 0.5$), and the top ranked C_d according to the s_{S_d} score. The other three approaches are simpler variations of CSF, including (i) **Support Frequency (SF)** – selects top ranked C_d according to S_d signatures, limiting the s_{S_d} score to a threshold $Th_{S_d} = \text{mean}(s_{S_d})$; (ii) **Core Frequency (CF)** – selects top ranked C_d according to C_d signatures, using score s_{C_d} and $Th_{C_d} = 0.5$; and (iii) **Core Näive (CN)** – selects C_d based exclusively on the list of known core domains.

B. Session Disentangling

Once the core domains requested by the user have been detected, the next step consists of identifying the components (i.e., traffic flows) of each of the websites present in the mix of traffic. This corresponds to the identification of the corresponding support domains, for each of the previously detected C_d. We implemented two different approaches to realize this web session disentangling task: (1) **Domains in C_d Signatures (DC)** – this is the most basic approach, and consists of identifying all the corresponding S_d, using C_d signatures. In a nutshell, this is achieved by intersecting the set of S_d in the corresponding signature DC_d, with the list of domains which are observed *temporally after* the specific C_d we are trying to reconstruct; (2) **Domains in C_d Signatures with Frequencies (SCF)** – this is a more robust approach and the one implemented in the solution, using the same approach as DC, but additionally using C_d and S_d signature frequencies FC_d and FS_d to only keep the S_d with higher probability of belonging to the corresponding C_d. Note that the web traffic disentangling additionally filters all the background traffic which is not associated to the reconstructed web pages.

C. Analysis of Signatures

Given the differences between web browsing in desktop and mobile devices, we construct the corresponding C_d and S_d signatures separately for desktop and smartphone webpage loading sessions – results for tablet are basically the same as for smartphone. Fig. 6(a) shows how big can be the C_d fingerprints for both desktop and smartphone devices. More than 60% of the top 500 Alexa websites have signatures with less than 50 S_d, but for about 20% of the pages, a C_d signature consists of more than 100 S_d. In general, smartphone signatures are smaller than those for desktop web pages, and as we show next, signatures are markedly different and provide good discrimination power. Fig. 6(b) shows the similarities between signatures of the same webpage from smartphone and desktop devices, using the well-know Jaccard-index as similarity metric. The Jaccard-index represents the similarity between finite sample sets, and it is defined as the size of the intersection divided by the size of the union of the sample sets. A Jaccard-index equals to 1 means that both signatures are exactly the same, whereas a Jaccard-index equals to 0 means that both signatures are totally different. Signatures are rather different between smartphone and desktop, with about 50% of



(a) Inference performance. (b) MOS prediction error.

Fig. 8: DeepQoE (RUM)SI and MOS inference performance.

		SML	DeepQoE	
dataset	**content**	**MAE-mAE (ms)**	**MAE-mAE (ms)**	gain (%)
all	**web/app**	**605 – 247**	**501 – 182**	**17.2 – 26.3**
web – desk	web	591 – 287	489 – 203	17.3 – 29.3
web – smart	web	757 – 351	607 – 250	19.8 – 28.8
web – tab	web	767 – 333	615 – 252	19.8 – 24.3
APP	app	309 – 110	238 – 82	23.0 – 25.5

TABLE III: Performance of multi-device/multi-app Web QoE Inference.

the pages showing a Jaccard-index below 0.5. For about 20% of the pages, signatures significantly overlap, with a Jaccard-index above 0.75. We now focus on the discriminative power of the signatures. For the specific identification task we are targeting, good signatures are those which do not overlap, and therefore can be mapped to exclusive web pages. Fig. 6(c) depicts the distribution of Jaccard-index values among all couples of C_d signatures, both for smartphone and desktop. In both cases values are small, with more than 97% of the couples with a Jaccard-index below 0.2. This confirms the a-priori good discriminative power provided by the C_d signatures. Regarding S_d signatures, Fig. 6(d) shows the number of C_d associated to each of the S_d in the top 500 Alexa websites, for both smartphone and desktop. Within the top 500 Alexa websites, there is a total of 9.601 unique support domains in the case of smartphone, and 12.384 in the case of desktop. About 80% of the S_d have only one C_d associated to, which is highly beneficial to also associate a S_d to the corresponding C_d. For 10% of the S_d, there are two associated C_d, and for the remaining 10%, there are three or more associated C_d. Within this category of S_d, we find support domains such as *www.google.com* or *www.facebook.com* with a strong presence on all visited web pages, in about 60% and 40% respectively.

We also analyzed the signatures generated for mobile apps; in particular, we analyzed the signatures generated for each different app action, for the four instrumented apps, and compare them to the signatures observed for web pages, thinking on a potential discrimination between app actions and general web browsing traffic. Fig. 7(a) reports de distribution of Jaccard-index values for signatures corresponding to app actions and smartphone web browsing, considering the webpage versions of the apps for intersection. Two main observations can be derived: firstly, there is a strong separation between the app startup signatures and the corresponding webpage loading signatures for Facebook and YouTube, with a Jaccard-index below 0.1. Secondly, for the case of BBC and Amazon, where there are multiple similar actions, there is a stronger overlap among signatures, but for about 60% of the couples, the values are below 0.4, which is still promising to identify individual app actions. Fig. 7(b-d) show the distribution of Jaccard-index values when considering, for each app and for each specific app action, the overlap with the full set of top-500 Alexa webpage signatures, in smartphone. Values are small, at most 0.25, and below 0.15 for more than 95% of the couples, suggesting that app-action signatures do not overlap with web-browsing signatures, and can therefore be properly used for clickstream identification and disentangling. We have only focused on the case of desktop and mobile web browsing fingerprinting, and leave apps identification for future work.

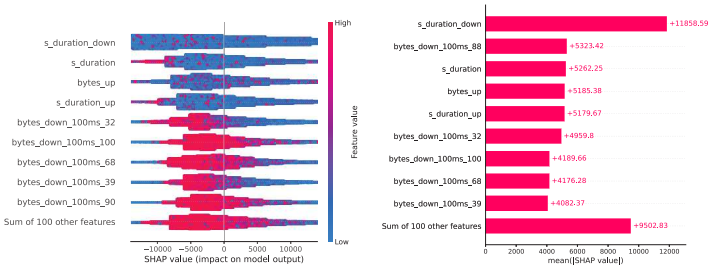
V. *DeepQoE* Evaluation

Next, we thoroughly evaluate *DeepQoE* modules for (i) SI inference and (ii) traffic filtering, both independently, as well (iii) as an end-to-end system. In Sec. V-A we evaluate the DL model inference performance for web browsing and apps, assuming an ideal scenario where single sessions are already isolated from each other. Sec. V-B evaluates the identification and disentangling of webpages for (a) desktop and (b) smartphone devices, without considering mobile apps – we did not construct app fingerprints in this study. Finally, Sec. V-C considers the end-to-end inference performance results for web browsing in (a) desktop and (b) smartphone.

A. (RUM)SI Inference and QoE Impact

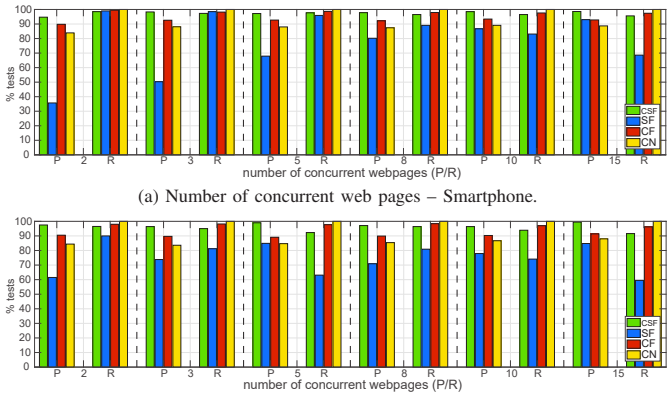
Using the generated data and network traffic features, we train *DeepQoE* for (RUM)SI inference. We train a **single model** for all web and app measurements, and then evaluate its performance for each independent dataset – web browsing per device and apps. **Unless otherwise stated, results presented next correspond to 5-fold cross validation**. To show evidence of the outperformance introduced by *DeepQoE*, we compare results against those obtained through a Shallow Machine Learning model (SML), following the state of the art [3], [4], [6]. In particular, we take the model used in [4], corresponding to LightGBM – a gradient boosting approach that uses tree based learning algorithms.

Tab. III reports the (RUM)SI inference performance of *DeepQoE* and the SML shallow model, for all data, and tested on each individual dataset. Fig. 8(a) additionally depicts the distribution of the inference errors obtained by *DeepQoE*. We assess performance using absolute errors (AE), taking both mean (M) and median (m) values, to filter out significantly large errors. The last column shows the performance gain of *DeepQoE* over SML – in MAE and mAE. The first observation is the consistency of generalization of both *DeepQoE* and SML models along the different datasets. Regarding web browsing, errors tend to be higher for web browsing in mobile devices (*web – smart* and *web – tab*) as compared to desktop, flagging the additional complexity introduced by the combination of OS (Android) and mobile hardware in the mapping between network traffic and visual display. Inference errors are significantly lower for the instrumented app user-interactions, but recall that the heterogeneity of contents is much smaller for this dataset – 11 different user actions as compared to 500



(a) SHAP-value distributions. (b) Mean absolute SHAP-values.

Fig. 9: Explainability of *DeepQoE*, for web browsing and apps.



(a) Number of concurrent web pages – Smartphone.

(b) Number of concurrent web pages – Desktop.

Fig. 10: Clickstream identification performance. *DeepQoE* uses CSF.

different web pages. Regarding the performance improvement achieved by *DeepQoE*, we see a significant gain above 25% (in the median) in all datasets, evidencing that a deep model can better track the underlying phenomenon and statistics.

A natural question which poses based on the presented results, is how relevant are the (RUM)SI inference errors and the performance gain achieved by *DeepQoE* in terms of end-user experience? To answer this question, we rely on well-known models for QoE prediction (MOS scores) based on waiting times, which assume that the relationship between waiting time and its QoE perception on a linear MOS scale is logarithmic. This model has been recently applied to SI in [7], using subjective lab studies for calibration. Using the models in [7], we are able to translate SI values into a MOS score between 1 (bad QoE) and 5 (excellent QoE). For the sake of simplified analysis, we map SI values into low (MOS = 1, 2, or 3) and high QoE (MOS = 4 or 5) categories. This is a standard cut used in the QoE community for binary user-experience analysis [3]. This threshold also results in a balanced split (53/47) in terms of number of sessions in each category. Fig. 8(b) depicts the corresponding QoE MOS inference errors obtained when mapping actual and inferred (RUM)SI values, for both *DeepQoE* and SML models. SML inference errors result in about 10% of the sessions being wrongly classified as low QoE, and above 13% wrongly classified as high QoE. *DeepQoE* is able to reduce these errors by about 40%, improving the classification of low and high QoE sessions by 44% and 34% respectively.

Another aspect we evaluated is the generalization of the trained model to websites that are not used in the training process. As we have recently shown, training inference models for specific content-based groups of webpages results in significant performance improvement as compared to training a single model for a broad set of webpages [13]. This directly suggests that a model trained on a set of webpages would underperform in a different, randomly selected set of unseen pages. To test this hypothesis, we randomly split the full dataset in a 80/20 proportion webpage-wise, training the model in the first 80% of the webpages, and testing in the remaining, non-overlapping 20% of webpages. As expected, inference performance degrades when testing on measurements coming from unseen webpages, but this degradation is rather limited, around 10% when considering the median absolute error as performance metric. Training a model capable to properly perform on the full span of webpages in the open Internet is certainly challenging, but focusing on the most popular websites already provides very good and useful results.

Finally, to shed some light on the underlying functioning of *DeepQoE*, we resort to explainable AI techniques describing which input features have a stronger influence in the predictions, and in which sense. We apply SHAP [15], a game theoretic approach to explain the output of any machine learning model, by calculating the contribution of each feature to the corresponding predictions. In a nutshell, SHAP shows the relative impact of each independent input feature on the model output, by comparing the relative effect of the inputs against the average model output. Fig. 9 depicts the top-10-sorted most relevant input features and their corresponding impact on the model output according to SHAP, for all web browsing measurements and all app user-actions. For each feature and for each individual input sample, the plot shows how much it pushes the model output from the base value to the actual output for this sample. The base value corresponds to the average model output over the training dataset. Higher feature values are shown in red, and lower in blue. Session-related features have a clearer stronger impact, as they are among the top-ranked. For example, session_duration (in downlink and overall) has the strongest impact in *DeepQoE*, with shorter sessions pushing the inferred SI values lower. Interestingly, and as expected, the most relevant progressive traffic download features are the ones corresponding to the end of the loading progress, e.g., bytes_down at the middle or end of the 100 time slots (100, 90, 88, etc.). Both observations suggest that the loading progress itself is less relevant than the overall duration. The high relevance of session-related features might be one of the reasons for the outperformance of *DeepQoE* as compared to previous CNN-based architectures taken in the state of the art for this inference task [5].

B. Web Pages Identification and Filtering

We evaluate the different clickstream identification approaches on top of synthetically generated multi-concurrent web pages sessions, considering a different number of concurrent web pages, from two to 15 concurrent web pages. For each number of concurrent web pages, the evaluation corresponds to a set of 10.000 samples synthetically gener-

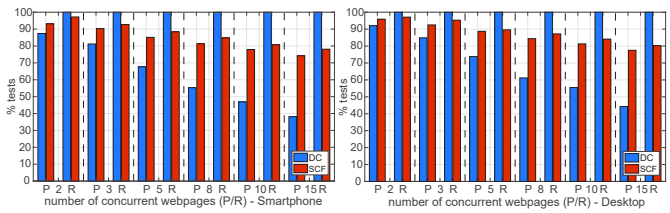
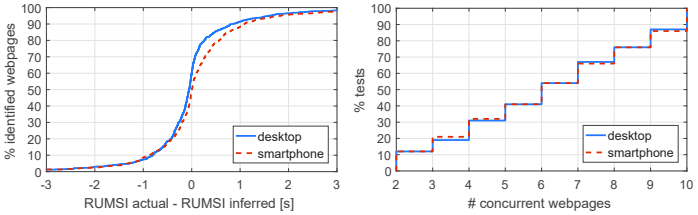


Fig. 11: Webpage disentangling performance. *DeepQoE* uses SCF.



(a) Inference performance. (b) Concurrent web pages.

Fig. 12: DeepQoE end-to-end performance.

	concurrent	individual (all sessions, cf. Tab. III)
device	**MAE-mAE (ms)**	**MAE-mAE (ms)**
desktop	580 – 263	489 – 203
smartphone	733 – 330	607 – 250

TABLE IV: Performance of end-to-end Web QoE Inference.

ated. These synthetic concurrent sessions are generated by randomly selecting traffic captures (pcap files) of individual loading sessions, and mixing them together, after properly aligning time references to actually make them concurrent. In a nutshell, this is achieved by setting the beginning of the mixed captures at the same start time (i.e., the time-stamp of the first packet in each capture), adding a random delay to each capture to emulate time precedence in the multi-session browsing. Fig. 10 reports the obtained results in terms or precision and recall, for desktop and smartphone devices, respectively. Recall and precision values are high for both device types, pointing to the good discrimination power of the signatures. The approach performing the worst is SF, which is expected, as it only relies on S_d signatures. Due to the nature of the approach and the evaluation, CN achieves 100% recall in all tests. As expected, precision drops for this approach, as some S_d are classified as C_d. As the number of concurrent pages grows, precision also increases, as the relative number of false positives decreases. CSF (the one used by *DeepQoE*) and CF perform similarly, but additionally using S_d signatures helps to increase precision of the identification approach. We follow a similar approach to evaluate the different web traffic disentangling approaches. In this evaluation, the list of C_d to reconstruct is given as input, and the task consists of identifying and correctly associating all the S_d to the correct C_d. Fig. 11 reports the obtained results in terms of precision and recall, again for desktop and smartphone mixed web loading sessions, respectively. The basic approach DC achieves 100% recall for all the traffic mixes, based on the signatures intersection. However, the approach is not precise, resulting in a significant fraction of misclassifications for S_d in the traffic mix. Precision drops as we add more concurrent pages, as overlapping S_d occur more often. Using both C_d and S_d signatures (SCF, the approach used by *DeepQoE*) significantly improves the precision of the disentangling approach, but naturally, sacrificing part of the identification performance in terms of recall. Still, disentangling and filtering performance is very high, even when dealing with 15 concurrent pages, both for smartphone and desktop devices.

C. End-to-End Inference Evaluation

The last evaluation targets the performance of the complete end-to-end monitoring solution, which uses the clickstream identification and the traffic disentangling steps to reconstruct isolated web pages for subsequent (RUM)SI inference. As before, we generate synthetic mixes of individual webpage loading sessions. We randomly generate two sets of 100 mixed traffic sessions each, one for smartphone and one for desktop. Within a set, the number of concurrent web pages is randomly selected for each new test, following a uniform distribution between two and ten consecutive web pages. Fig. 12(b) depicts the distribution of the number of concurrent web pages randomly selected. For about 70% of the tests, the mix of traffic consists of five or more concurrent web pages, resulting in a challenging test case. A total of 613 and 612 individual webpage loading sessions are selected by the random algorithm to generate the 100 tests, for desktop and smartphone, respectively. Clickstream detection performance is highly accurate, with a Precision/Recall = 97.2%/95.6% for desktop, and Precision/Recall = 93.1%/97.3% for smartphone. Fig. 12(a) and Tab. IV report the (RUM)SI inference results, obtained by the end-to-end pipeline, after the identification and filtering of web pages traffic. Inference results are accurate, achieving results comparable to the application of the model on top of single, isolated web-traffic loading sessions. In particular, median absolute errors are in the order of 250 ms to 300 ms, very close to the inference results obtained from individual sessions – cf. Tab. III.

VI. Concluding Remarks

DeepQoE is a deep-learning based approach to infer the QoE of web services and mobile applications from the ISP perspective, relying exclusively on the analysis of encrypted network traffic. It implements a web fingerprinting solution to identify individual web browsing sessions within concurrent web pages traffic, enabling highly detailed, per webpage QoE inference in practical deployments. Through extensive evaluations over a large and heterogeneous dataset composed of web and app measurements, we have shown that *DeepQoE* can infer the Speed Index of web browsing sessions and general mobile apps with 25% of gain when compared to the state of the art – based on shallow learning models, and reducing the QoE inference error in terms of mean opinion scores by nearly 40%. In addition, we have demonstrated the feasibility of the proposed web traffic identification and disentangling approaches, showing that the low error rates incurred in the traffic filtering process have limited impact on the final SI estimations.

REFERENCES

[1] Martino Trevisan, Idilio Drago, Marco Mellia, "PAIN: A Passive Web Performance Indicator for ISPs," *Computer Networks*, vol. 149, pp. 115-126, 2019.

[2] Enrico Bocchi, Luca De Cicco, Dario Rossi, "Measuring the Quality of Experience of Web Users," *ACM SIGCOMM Computer Communication Review*, vol. 46(4), 2016.

[3] Sarah Wassermann, Pedro Casas, Zied Ben Houidi, Alexis Huet, Michael Seufert, Nikolas Wehner, Joshua Schuler, Shengming Cai, Hao Shi, Jinchun Xu, Tobias Hoßfeld, Dario Rossi, "Are you on Mobile or Desktop? On the Impact of End-user Device on Web QoE Inference from Encrypted Traffic," in *16th IEEE International Conference on Network and Service Management (CNSM)*, 2020.

[4] Alexis Huet, Antoine Saverimoutou, Zied Ben Houidi, Hao Shi, Shengming Cai, Jinchun Xu, Bertrand Mathieu, Dario Rossi, "Revealing QoE of Web Users from Encrypted Network Traffic," in *IFIP Networking Conference*, 2020.

[5] Alexis Huet, Antoine Saverimoutou, Zied Ben Houidi, Hao Shi, Shengming Cai, Jinchun Xu, Bertrand Mathieu, Dario Rossi, "Deployable Models for Approximating Web QoE Metrics From Encrypted Traffic," in *IEEE Transactions on Network and Service Management*, vol. 18(3), 2021.

[6] Pedro Casas, Sarah Wassermann, Nikolas Wehner, Michael Seufert, Joshua Schüler, Tobias Hoßfeld, "Mobile Web and App QoE Monitoring for ISPs - from Encrypted Traffic to Speed Index through Machine Learning," in *13th IFIP Wireless and Mobile Networking Conference (WMNC)*, 2021.

[7] Tobias Hoßfeld, Florian Metzger, Dario Rossi, "Speed Index: Relating the Industrial Standard for User Perceived Web Performance to Web QoE," in *10th International Conference on Quality of Multimedia Experience (QoMEX)*, 2018.

[8] Diego Neves da Hora, Alemnew Sheferaw Asrese, Vassilis Christophides, Renata Teixeira, Dario Rossi, "Narrowing the Gap between QoS Metrics and Web QoE using Above-the-Fold Metrics," in *Passive and Active Measurement Conference (PAM)*, 2018.

[9] Ravi Netravali, Vikram Nathan, James Mickens, Hari Balakrishnan, "Vesper: Measuring Time-to-Interactivity for Web Pages," in *15th USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, 2018.

[10] Qingzhu Gao, Prasenjit Dey, Parvez Ahammad, "Perceived Performance of Top Retail Webpages in the Wild: Insights from Large-scale Crowdsourcing of Above-the-fold QoE," in *Workshop on QoE-based Analysis and Management of Data Communication Networks (Internet-QoE)*, 2017.

[11] Ashkan Nikravesh, Qi Alfred Chen, Scott Haseley, Xiao Zhu, Geoffrey Challen, Z. Morley Mao, "QoE Inference and Improvement without End-host Control," in *IEEE/ACM Symposium on Edge Computing (SEC)*, 2018.

[12] Javad Nejati, Meng Luo, Nick Nikiforakis, Aruna Balasubramanian, "Need for Mobile Speed: A Historical Study of Mobile Web Performance," in *4th Network Traffic Measurement and Analysis Conference (TMA)*, 2020.

[13] Pedro Casas, Sarah Wassermann, Nikolas Wehner, Michael Seufert, Tobias Hoßfeld, "Not all Web Pages are Born the Same. Content Tailored Learning for Web QoE Inference," in *6th IEEE International Symposium on Measurements & Networking (M&N)*, 2022.

[14] Ignacio N. Bermudez, Marco Mellia, Maurizio M. Munafo, Ram Keralapura, "DNS to the Rescue: Discerning Content and Services in a Tangled Web," in *ACM Internet Measurement Conference (IMC)*, 2012.

[15] Scott Lundberg, Su-In Lee, "A Unified Approach to Interpreting Model Predictions," in *31st International Conference on Neural Information Processing Systems (NIPS)*, 2017.

[16] Alex Krizhevsky, Ilya Sutskever, Geoffrey E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," in *26th International Conference on Neural Information Processing Systems (NIPS)*, 2012.

[17] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, Andrew Rabinovich, "Going Deeper with Convolutions," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[18] Karen Simonyan, Andrew Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," in *3rd International Conference on Learning Representations (ICLR)*, 2015.

[19] Abien Fred Agarap, "Deep Learning using Rectified Linear Units (ReLU)," *arXiv preprint arXiv:1803.08375*, 2018.

[20] Patrick Meenan, "Webpage Performance Testing," online available at https://github.com/WPO-Foundation/webpagetest, 2021.

[21] Dan Cuellar, "Appium, an Open-source, Cross-platform Test Automation Tool for Native, Hybrid, and Mobile Web and Desktop Apps," online available at https://github.com/appium/appium, 2022.

[22] Patrick Meenan, "RUM SpeedIndex – SpeedIndex Measurements from the Field using Resource Timings," online available at https://github.com/WPO-Foundation/RUM-SpeedIndex, 2020.

[23] Patrick Meenan, "Command-line Port of the WebPagetest Mobile Video Processing and Metrics Code," online available at https://github.com/WPO-Foundation/visualmetrics, 2021.