A Dive into the Dark Web: Hierarchical Traffic Classification of Anonymity Tools

Antonio Montieri, *Graduate Student Member, IEEE*, Domenico Ciuonzo, *Senior Member, IEEE,* Giampaolo Bovenzi, Valerio Persico, and Antonio Pescapé, *Senior Member, IEEE*

Abstract—Privacy-preserving protocols and tools are increasingly adopted by Internet users nowadays. These mechanisms are challenged by the process of Traffic Classification (TC) which, other than being an important workhorse for several network management tasks, becomes a key factor in the assessment of their privacy level, both from offensive (malign) and defensive (benign) standpoints. In this paper we propose TC of anonymity tools (and deeper, of their running services and applications) via a *truly hierarchical approach*. Capitalizing a public dataset released in 2017 containing anonymity traffic, we provide an in-depth analysis of TC and we compare the proposed hierarchical approach with a flat counterpart. The proposed framework is investigated in both the usual TC setup and its "early" variant (i.e. only the first segments of traffic aggregate are used to take a decision). Results highlight a general improvement over the flat approach in terms of all the classification metrics. Further performance gains are also accomplished by tuning the thresholds ensuring progressive censoring. Finally, fine-grain performance investigation allows to (a) demonstrate lower severity of errors incurred by the hierarchical approach (as opposed to the flat case) and (b) highlight poorly-classifiable services/applications of each anonymity tool, gathering useful feedback on their privacy-level.

Index Terms—Dark web; Dark net; Tor; I2P; JonDonym; traffic classification; hierarchical classification; anonymity; privacy; security.

1 Introduction

In line with the growing criticality of the activities users perform online, privacy is today a key concern. In particular, preserving the *anonymity* of users is an aspect that has gathered the attention of the research community, that over last years put the effort in designing and developing tools able to achieve privacy at varying degrees. As a result, at present a number of Anonymity Tools (ATs) are freely available, able to hide—besides the content of the communication itself, via encryption—the identity of the parties (source and destination) involved in the communication.

The *most popular* ATs developed in recent years are The Onion Router (Tor)[1], the Invisible Internet Project (I2P)[2], and JonDonym[3] (formerly known as Java Anon Proxy or WebMix). These tools allow users to preserve their anonymity, e.g. encrypting data multiple times and routing it through multiple stations, providing each with just a piece of the information. From the user viewpoint, ATs allow browsing and running applications (also circumventing restrictions enforced at either providers or governmental level) keeping their identity and location secret to any intermediary entity observing the traffic. Accordingly, tracing users and their activities across these networks is a complex task.

ATs have been investigated in recent years by several studies from *different perspectives* including design improvement, AT delay and performance analysis, feasibility of effective attacks to ATs, users' behavior profiling and identity disclosure risk, and censoring policies enforced for ATs [1]. Among these crucial aspects, a cardinal issue is *understanding to what extent (encrypted) ATs traffic can be classified*, i.e. at which granularity ATs and related applications can be accurately recognized by external entities. Accordingly, in this paper, we focus on ATs Traffic Classification (TC).

Indeed, TC—consisting in associating traffic aggregates to specific applications or groups of applications (viz. *labeling* them)—is of utmost importance in Internet traffic engineering, along with related methodologies & tools, as they jointly support activities such as network monitoring, security assessment, application identification, anomaly detection, accounting, advertising, and service differentiation [2, 3]. For these reasons, TC has gained importance in recent years due to growing incentives to disguise certain applications [4], comprising those generating *anonymous traffic* [5].

On the one hand, investigating TC of ATs is useful to designers, as it puts their effectiveness to the test, reveals shortcomings, and leads the way to robustify them. On the other hand, these studies are of interest to providers as well as governmental entities, providing knowledge e.g. to enforce informed engineering policies or to prevent users performing unwanted actions. Hence, classifying ATs traffic is a particularly appealing and challenging research field whose state-of-art solutions leave room for improvement.

Because of the specific nature of ATs encrypted traffic, Machine Learning (ML) classifiers represent the natural

- *Antonio Montieri, Domenico Ciuonzo and Giampaolo Bovenzi are with the University of Napoli Federico II (Italy). E-mail: antonio.montieri@unina.it; domenico.ciuonzo@ieee.org; giampaolo.bovenzi@gmail.com.*
- *Valerio Persico and Antonio Pescapé are with the University of Napoli Federico II (Italy) and with NM2 s.r.l. (Italy). E-mail: valerio.persico@unina.it; pescape@unina.it.*

Manuscript received Oct. 2nd, 2018; revised Jan. 11th 2019; accepted Feb. 21th 2019. This work is partially funded by art. 11 DM 593/2000 for NM2 s.r.l. (Italy).

1. Tor Project. https://www.torproject.org.
2. The Invisible Internet Project. https://geti2p.net.
3. JonDonym. http://anonymous-proxy-servers.net.

enabler, able to provide decisions based on the sole traffic-flow features [5] and to overcome shortcomings due to the application of common solutions (e.g., those based on payload inspection or earlier port-based ones [6]).

On the top of that, hierarchical classification represents a perfect match for TC of ATs, as (i) it allows fine-grained tuning and design, potentially leading to classification performance gains; (ii) it also brings a number of *"practical" benefits by design*, at cost of moderate complexity increase. For example, re-training does not involve all the nodes in the hierarchy when new applications leveraging anonymity networks are released. Also, distributed deployment of TC tasks, thanks to the modularity of the framework, is enabled in the network (thus hierarchical classification could be achieved through chaining of *virtualized network functions*, each associated to a classifier). Albeit these benefits are granted by the hierarchical approach itself, research efforts are needed to deepen the aspects of design optimization (to obtain enhanced performance) and fine-grain evaluation to delve into privacy-level assessment of ATs.

1.1 Related Works

Up to our knowledge, there are no studies focused on classification of different anonymity services at various levels of granularity via *hierarchical learning*. Accordingly, this section first discusses the (few) works tackling standard TC (i.e. not focusing on traffic generated by ATs) via a hierarchical approach. Then, literature on TC of ATs via a "flat" (viz. non-hierarchical) approach is reviewed, including the conceptually-related Website Fingerprinting (WF) problem.

Hierarchical Traffic Classification

A first approach to hierarchical TC is described in [7], where a three-level system for Peer-to-Peer (P2P) biflow-based traffic categorization is proposed to discern among 11 P2P and 5 non-P2P apps. The first-level classifier adopts a Support Vector Machine (SVM) for P2P/non-P2P recognition, whereas Support Vector Data Descriptions are employed at the other two levels for P2P-type (i.e. file-sharing, messenger, and TV) and P2P plus non-P2P app TC, respectively. Results (pertaining to each single classifier in the hierarchy *separately*) show that the proposed approach achieves precision and recall $\geq 95\%$ in P2P/non-P2P recognition and $\geq 93\%$ in P2P-type classification, while a recall drop down to 71% is observed at last level. Similarly, Grimaudo et al. [8] propose the adoption of a hierarchical classifier to allow Internet TC (into ≥ 20 fine-grained classes), showing a comparison with flat-learning results. The proposed tree-structured (three-level) taxonomy introduces also a first level which identifies known/unseen traffic. Both (per-classifier) feature selection (from a set of 200, as proposed by past literature) and (coarse-grained) optimization w.r.t. different classifiers (such as Decision Trees, Neural Networks, and SVMs) are considered. Results show that the proposed hierarchical system outperforms off-the-shelf flat classification, with an average recall $> 95\%$ for most popular traffic classes.

In [9] a novel *multilateral* (viz. multi-view) and hierarchical approach for Internet TC is proposed, further refined into *FORMULA* framework [10], operating on biflow-segmented traffic. Specifically, the devised approach relies on a taxonomy that provides multilateral identification based on *four* different classification criteria (i.e. service, application, protocol, and function) via a hierarchical structure (whose working principle is however not detailed), supporting roll-up and drill-down operations on the classification results. Unfortunately, the collected traffic lacks a ground truth, precluding a verification of the reported results.

In [11], the authors propose a fine-grained scheme for hierarchical TC of video traffic based on clustering. A hierarchical version of k-Nearest Neighbor (k-NN) is fed with the most discriminative statistical features selected among 40 extracted from downstream/upstream data, ranked by the information gain ratio. The dataset, including six types of (*symmetric* and *asymmetric*) traffic, is collected on the campus network of Nanjing University. Experimental results report $\geq 97\%$ F-measure in discriminating among all the considered (video) traffic and superior performance w.r.t. existing alternatives, while providing only a slight time-complexity increase. Recently, in [12] a two-level hierarchical classification framework is presented to identify the services running within HTTPS connections leveraging a set of features robust to alteration (e.g., statistics of inter-arrival times and packet/payload sizes). The proposed evaluation method, based on real traffic traces, achieves a recall within $[95, 100]\%$ in 50 out of the 68 HTTPS services considered.

(Flat) Traffic Classification of Anonymity Tools

The analysis of ATs has been initially carried in *private networks*, e.g. with the aim of discriminating between HTTPS and Tor traffic [13]. In detail, by leveraging a dataset made of (i) regular HTTPS traffic, (ii) HTTP and (iii) HTTPS over a private Tor network, authors show that HTTP/HTTPS traffic over Tor can be detected with $\geq 93\%$ accuracy, employing Random Forest (RF), C4.5, and AdaBoost classifiers.

On the other hand, a few works focus on TC analyzing *real traffic* from anonymity networks, as most of the "experimental" literature explores anonymous WF, whose aim is to identify a web-page accessed by a client of encrypted and anonymized connections by observing patterns of data flows (e.g., packet size and direction). Herrmann et al. [14] propose a Multinomial Naïve Bayes (MNB) that relies on the normalized frequency distribution of IP packet sizes to tackle the WF problem in the context of different privacy-enhancing technologies (in a closed-world scenario) including Tor and JonDonym. Although Tor and JonDonym guarantee a better protection than other privacy-enhancing technologies (i.e. OpenSSL, OpenVPN), they prove to be not perfect (3% and 20% average accuracy, respectively). In [15] the same problem is tackled via a Support Vector Classifier, obtaining a gain of detection rate over [14] from 3% to 55% (resp. from 20% to 80%) in Tor (resp. JonDonym) network. On the other hand, in an open-world case, the detection rate drops with a maximum of 73% and 0.05% false-positive rate. More recently, in [16] a WF approach aimed to overcome limitation of previously-devised alternatives is proposed and tested on a huge real-world representative dataset, exploring the limits of WF at Internet scale. Specifically, these are highlighted by a precision/recall drop with the size of the background sites which the monitored pages need to be distinguished from. Finally, we mention that the adoption of Deep Learning (DL) to WF is also a currently-investigated topic. A novel DL-based method to

deanonymize Tor traffic is proposed in [17] and tested on a dataset made of $\geq 3 \cdot 10^6$ network traces, with the best-performing DL model being $+2\%$ accurate than state-of-the-art attacks. In [18] a WF attack against Tor is developed, leveraging a Convolutional Neural Network and evaluated against state-of-the-art defenses (i.e. WTF-PAD and Walkie-Talkie). Results report an accuracy $> 98\%$ on undefended Tor traffic, while reaching $> 90\%$ accuracy (resp. 49.7%) when WTF-PAD (resp. Walkie-Talkie) is employed, with the attack remaining effective also in an open-world setting.

Moving to pure TC of ATs (based on real-data), He et al. [19] devise an approach based on Hidden Markov Models (HMMs) to classify four categories of Tor traffic (P2P, FTP, IM, and Web). HMMs are employed to build inbound and output models of the application types considered, and are fed with features based on burst volumes and directions of Tor flows, obtaining an accuracy up to 92%. AlSabah et al. [20] present a ML-based method employing Naive Bayes (NB), Bayesian Network (BN), functional and logistic model trees to recognize applications used by Tor users. Both *circuit-level* and *cell-level* information is leveraged for offline and offline/online classification, respectively. The highest accuracy achieved in online (resp. offline) case equals 97.8% (resp. 91%). A similar setup is proposed in [21] for user activity recognition by means of four classifiers (NB, BN, RF, and C4.5) fed with *traffic-flow* and *circuit-level* features. Both approaches reach $\approx 100\%$ accuracy, with flow-based TC being *less demanding* and based on data that could be captured anywhere between the user and the Tor's relay. Along the same lines, Shahbar and Zincir-Heywood [22] employ flow-based (statistical) traffic analysis to prove whether Tor Pluggable Transports (PTs) can evade censorship systems. Adopting a C4.5 classifier (and based on a thorough analysis), authors show that Tor PTs usage is recognizable, as PT-based obfuscation changes the content shape in a distinct way w.r.t. Tor (i.e. conferring to flows distinctive fingerprints). The effects of bandwidth sharing on I2P is analyzed by the same authors in [23] considering both application and user profiling achievable by an attacker. Using a C4.5 classifier fed with flow-based features, results show that users and applications on I2P *can* be profiled, with a harmful (resp. beneficial) effect of the shared bandwidth increase on applications (resp. users) profiling accuracy.

Recently, Shahbar and Zincir-Heywood [24] describe Anon17 public dataset comprising directional traffic-flows obtained by collecting data from three ATs (i.e. Tor, I2P, and JonDonym). Besides, detailed information about the traffic types and applications running on Tor and I2P is provided in the form of *three-level labels* for each flow. Up to our knowledge, the sole public dataset similar to Anon17 is that described in [25], containing however *only* Tor traffic of eight applications (browsing, audio, chat, mail, P2P, FT, VoIP, and video). Anon17 dataset is leveraged in [5], where three *flat classifiers* (that is, one per level, i.e. Anonymity Network, Traffic Type, and Application), are employed to perform TC, also in its "early" variant [26]. In detail, at each level the performance of five ML classifiers (NB, MNB, BN, C4.5, and RF) is compared, highlighting insights about the number and nature of relevant features needed for an accurate TC. Results show that the anonymity networks (i.e. Tor, I2P, and JonDonym) can be easily distinguished (up to 99% F-measure). Differently, the specific application generating the traffic can be classified with up to 69% F-measure. At both levels, early TC achieves worse results.

These findings highlight the need for delving into the problem via successive splits and devising a *more sophisticated classification framework* for harder classification tasks.

1.2 Summary of Contributions

The main contribution of this paper is a detailed study on TC of ATs by proposing a *general Hierarchical Classification (HC) framework* (see Fig. 1). The novelty of this work has a *two-fold* significance: it (i) investigates the unexplored adoption of hierarchical approaches to TC of ATs (as opposed to existing flat learning approaches) and (ii) overcomes the limitations of earlier hierarchical proposals dealing with standard TC. Such framework *uniquely suits* to encrypted traffic and *naturally allows* for both coarsening & narrowing of classification results. Our study includes its design, implementation, optimization and evaluation.

Specifically, the hierarchical approach resorts to a structure of (potentially different) classifiers arranged in a *tree* fashion, each specialized in labeling a subset of classes. Such framework exploits the "divide-et-impera" principle, enabling the partition of workload among several classifiers, almost-naturally enabling a distributed implementation. Hence, the obtained HC structure grants scalability, enables both per-node tuning and performance analysis, and supports roll-up and drill-down operations of the results, which are provided at different levels of detail. Per-node performance figures also allow to accurately evaluate per-node behaviors, and allow to identify potential causes of performance degradation, and thus prove useful in guiding feedback-driven design improvement. Besided, HC design supports incremental-update (e.g., only a minor additional training is required when a new part of the application traffic is needed, instead of re-training the whole system). Finally, the proposed architecture allows progressive censoring of "unsure" instances, implementing a per-classifier *reject option* via (a set of) independently-tunable thresholds.

The key issue researchers encounter in the collection of AT traffic datasets and the resulting lack of data publicly available in this field is herein overcome by leveraging the recently released Anon17 dataset [24], representing an important shared workbench for research studies on anonymity and consisting of a collection of traces gathered by different ATs, at different granularities: *anonymous network*, *traffic type*, and *application*. Although our HC strategy is encouraged by the nature of the classification problem defined on this dataset, the underlying HC principle *generally suits* to the varying degrees of privacy ensured by a certain AT, including its identification within "normal" traffic.

The results show that the proposed HC approach proves to be a very good fit to the problem of classification of ATs traffic. The HC approach is able to discern among ATs looking at their encrypted traffic, also when considering only the features extracted from the first K packets of each flow (i.e. implementing *early TC*). Compared to flat approaches we have recently proposed [5], the hierarchical framework also provides performance gains at application level. This result holds both when considering classic performance indexes at macroscopic level (typically adopted

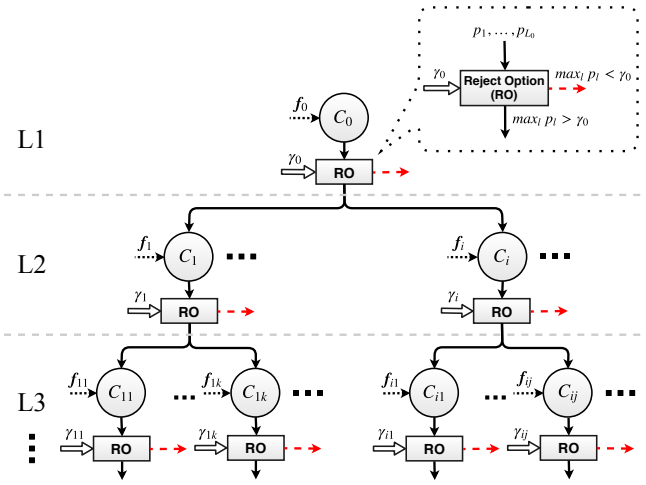


Figure 1: Sketch of the proposed HC framework.

in the TC literature) and by accurately breaking it down and evaluating it along multiple facets (i.e. by introducing metrics able to capture error severity).

The rest of the manuscript is organized as follows: Sec. 2 describes the considered hierarchical TC framework of ATs; the experimental environment and the corresponding classification results are reported in Secs. 3 and 4, respectively; finally, Sec. 5 provides conclusions and future directions.

2 Hierarchical Traffic Classification of Anonymity Tools

In this section we introduce the proposed framework for HC of AT traffic, as streamlined in Fig. 1, whose main components are discussed in the following. In detail, preliminaries on HC are discussed first, together with a general overview of the proposed approach (Sec. 2.1). Then, terms and concepts regarding traffic objects are introduced, along with a description of the feature sets (Sec. 2.2). The dissertation ends with the classification algorithms (Sec. 2.3) which could be adopted for AT (hierarchical) traffic analysis.

2.1 Preliminaries and Proposed HC Framework

In what follows we provide HC preliminaries needed to understand the design choices adopted for the TC framework here proposed and investigated in the context of TC of ATs.

First, we remark that our classification framework focuses on classes whose relationship can be summarized in the form of a tree (each class has one parent class, *at most*) with T classification *levels* and L_t classes to discriminate from at t^{th} level (whose number increases with the depth), organized in the corresponding set $\mathcal{L}_t \triangleq \{1, \ldots, L_t\}$.

The tree structure can be explored with three alternative approaches [27]: (*i*) top-down approach, i.e. the system employs a set of local classifiers each tackled to a specific sub-problem; (*ii*) big-bang (or global) approach, when a single classifier copes with the entire class hierarchy (i.e it is trained by considering the entire class hierarchy at once); (*iii*) flat classification approach, ignoring the class relationships at different levels (solving a classification task at each level of the hierarchy in an independent fashion), as adopted in [5].

In this paper, we adopt the *top-down* choice (similarly to [8]). In this case, for each instance to be classified, the HC framework first predicts its first-level (most generic) class, then it uses that predicted class to narrow the choices of classes to be predicted at the second level (i.e. allowed second-level predicted classes are the children of that predicted at the first level), and so on. Although errors at a certain class level could propagate downwards the hierarchy, this choice promotes the *architecture modularity*, which is crucial in TC, as opposed to big-bang and flat classifiers.

Further, *three different ways of using the local information can be employed*, mainly differing in their training phase [27]: (*i*) Local Classifier per Node (LCN); (*ii*) Local Classifier per Level (LCL); (*iii*) Local Classifier per Parent Node (LCPN). LCN and LCL consist in training one binary classifier for each node and a multi-class classifier for each level of the class hierarchy, respectively. Unluckily, since these two approaches suffer from class-membership inconsistency and have higher complexity, in this work we adopt *LCPN*, that is widely used in the literature [8, 11] and requires a *multi-class classifier* for each parent node in the class hierarchy, trained to distinguish among its children nodes (usually less than L_t, with t being node classification level).

Fig. 1 reports a sketch of the proposed HC framework. We highlight that the indexing of a node reflects the ordered list of its ancestors (except the *root* C_0), e.g. C_{ij} denotes the j^{th} L3 classifier having C_0 and C_i, as grandparent and parent, respectively. Classifier C_{ij} is in charge of discriminating from $\bar{L}_{ij} < L_3$ classes, grouped within the set Ω_{ij}, based on the associated prediction probabilities $p_{1,} \ldots p_{\bar{L}_{ij}}$.

The proposed HC framework is trained by *recursively* splitting the training set according to the tree structure. Specifically, the procedure starts from the root classifier C_0 trained using the whole set. On the other hand, each node concurring to $t > 1$ level classification uses a training set corresponding to a subset of $\mathcal{L}_t$, the elements all belonging to the same class at $(t-1)$. Reducing the number of classes per classifier hopefully simplifies the resulting problem and reduces the error scope of flat classification.

In addition, the proposed framework adopts a *progressive-censoring* (viz. non-mandatory leaf node prediction [27]) policy, accomplished by equipping each classifier node with a "reject option" that censors "unsure" classification outcomes at intermediate layers (RO blocks in Fig. 1). In other words, the reject option forces the classification process to stop for a given instance when a classifier node (e.g. $C_{i,j}$) does not reach a clear verdict, i.e. when the highest class prediction probability (e.g. $\max_{\ell=1,\ldots\bar{L}_{i,j}} p_\ell$) is below a threshold (e.g. γ_{ij}). This design choice—already introduced and justified in the mobile context in a flat scenario [28]—here avoids that misclassifications are propagated downwards, at the expense of *coarser-grained* predictions.

By looking at the hierarchy of classifiers reported in Fig. 1, it is apparent that its naivest implementation resorts to the *same* classification algorithm and feature set throughout all the hierarchy. Nonetheless, although HC can achieve a potential performance gain w.r.t. a flat approach even in this case (due to the decomposition of the classification task into sub-problems), the proposed framework allows for further *optimization* [27]. Indeed, classification performance is expected to improve with more refined

implementations, leveraging a *specific selection of features for classification*, and/or using *different classification algorithms at different nodes of the class hierarchy* (e.g., chosen from a pool of classifiers available). This work investigates both these optimization degrees-of-freedom. In case both the classifier and the feature set are optimized at each node, the combinatorial explosion of the resulting optimization is herein circumvented by a decoupled design resorting to *per-node performance*, e.g. selecting the pair corresponding to the classifier and the number of features ensuring the highest score for the sub-classification problem the classifier node is in charge to solve. Nonetheless, we remark that an optimization based on complete enumeration or alternative heuristics does not contrast with HC architecture in Fig. 1.

2.2 Traffic Object and Feature Design

The definition of a specific traffic object (also known as traffic view) determines how raw traffic is segmented into multiple discrete traffic units [2] which are then labeled by the classifier. We remark that, although several proposals exist and have been considered in the (ATs) TC literature, *flows* and *biflows* are the most commonly used traffic objects [5, 24]. In more detail, a *flow* is a set of packets with the same 5-tuple (i.e. source IP, source port, destination IP, destination port, and transport-level protocol). On the other hand, a *biflow* includes both directions of traffic. In other words, in the latter case, `(IP address, port)` pairs of source and destination are *interchangeable*. Referring to the hierarchical approach illustrated in Fig. 1, we assume that all the classifiers in the hierarchy shall *operate on a common TC object*, although not restricted to a specific one.

As previously explained, different sets of features (of different sizes) can be considered to feed the classifiers in the hierarchical architecture (to achieve accurate TC), as shown in Fig. 1. For instance, referring to the example in Sec. 2.1, classifier C_{ij} is assumed to rely on M_{ij} features, collected in the vector $\boldsymbol{f}_{ij}$. Finally, given a set of features, *feature selection/extraction techniques* are adopted to extract only a subset among them, with the aim of improving further TC performance, while reducing the computational complexity.

2.3 Classification Algorithms

As in the case of set of features, the proposed HC approach allows for a different classifier to be employed at each node (see Fig. 1). Therefore, any ML/DL-based (as ATs traffic is encrypted) supervised classifier can be adopted. For example, referring to classifier C_{ij} in the example of Sec. 2.1, any ML/DL-based classifiers could be used to discriminate from $\bar{L}_{ij}$ classes within the set Ω_{ij}. The sole requirement for each classifier is to be able to provide its soft-output vector, required by the censoring mechanism described in Sec. 2.1.

3 Experimental Environment

Hereinafter, we first report the main details of Anon17 dataset and the pre-processing operations carried out on it (Sec. 3.1); then we discuss the design choices made to implement the HC system, matching the characteristics of the dataset (Sec. 3.2); finally, we introduce the performance metrics employed to evaluate our solution (Sec. 3.3).

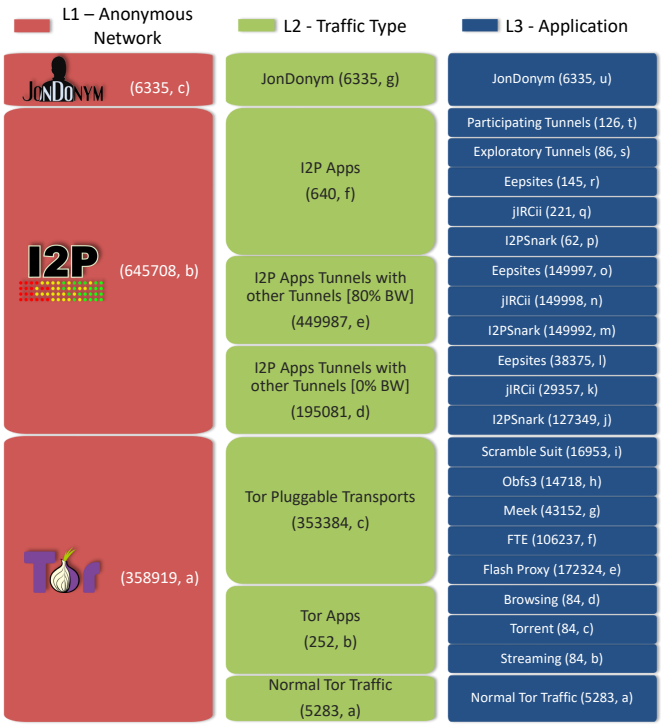


Figure 2: Anon17 Classification Levels: Anonymous network (L1), Traffic Type (L2) and Application (L3), with total no. of samples per class and class label at each level.

3.1 Dataset Description

Anon17 was collected in a *real-network environment* at the NIMS Lab [24] during 2014–17 and gathers traffic from Tor, I2P, and JonDonym. The dataset has been labeled leveraging the information provided by the anonymity services themselves. Indeed, it provides labels at different levels: (*i*) *Anonymous Network Level* ($L_1 = 3$ classes); (*ii*) *Traffic Type Level* ($L_2 = 7$ classes); (*iii*) and *Application Level* ($L_3 = 21$ classes). This label organization promotes analyses of ATs traffic at different levels of granularity, as well as the implementation of hierarchical approaches, as remarked in Fig. 2. We point to [24] for obtaining exhaustive information on Anon17 dataset. In this study, we down-sample to 5% the most-populated traffic-type classes of the original dataset, adopting a pre-processing strategy similar to [5], so as to mitigate class imbalance. This represents a safe choice due to the high number of samples of "majority" classes.[4]

According to the above dataset description, we tackle the problem of classification of ATs traffic assuming that we are in the presence of AT traffic only, similarly to [5]. The depicted scenario refers to a context where an *upstream classifier* has been able to separate AT traffic from clear or standard-encrypted one (e.g., as shown by [13] for Tor network). Hence, the aim of the proposed approach is to assess discrimination of anonymity services and related applications *once this AT traffic has been separated from other traffic*. More generally, the results of our analysis can be intended as an upper bound on the ATs classification performance in the case of an *open-world* assumption. Indeed, the

4. Experimental results, not shown, have highlighted negligible difference with the (additional) use of oversampling of "minority" classes.

use of an upstream classifier would perfectly fit within the hierarchical approach proposed in this paper, with design and evaluation of the whole AT-TC system (truly operating in an open-world scenario) left as future work.

3.2 HC Implementation Choices

Hereinafter, we briefly discuss how the general HC framework described in Sec. 2 is specialized in the case of its adoption on Anon17 dataset, with the following paragraphs covering all the different aspects needing specification. HC framework specialized for classification of Anon17 traffic-flows has been implemented in *Python*, leveraging *Weka* wrapper and *Scikit-learn* utilities. A Virtual Machine (OS Ubuntu Server 16.04) equipped with 32 VCPUs and 64 GB RAM, running on private *OpenStack* cloud platform, has been employed for evaluating the HC framework. Finally, the code used for the analyses has been made available on GitHub[5] to foster reproducibility of the experiments.

HC Architecture

Based on the nature of the traffic in Anon17 dataset, the TC here considered is arranged in three levels, corresponding to anonymity networks, traffic types, and specific applications. As a whole, the HC framework resorts on *one* classifier at L1, *two* classifiers at L2 and *five* classifiers at L3.

Traffic Object and Feature Sets

Anonymous traffic contained in Anon17 is split into different *flows* (see Sec. 2.2), by means of the flow-exporting tool *Tranalyzer2* [29], constituting the TC object here employed. We highlight that the direction of each flow (considered as a feature) is indicated as "A" or "B" for client-to-server and vice versa, respectively. We note that the proposed HC approach could even operate at the *packet-level*, in principle (viz. the TC object could be the *single packet*[6]). For each traffic flow, Anon17 provides different sets of features extracted via Tranalyzer2 [29]. In this work, we consider *two* different feature sets, referred hereinafter to as `TC_set` and `EarlyTC_set`. In brief, `TC_set` capitalizes complete traffic flows, while `EarlyTC_set` only relies on the first K packets of each flow, thus enabling early TC.[7]

In detail, `TC_set` originally refers to 81 per-flow statistical features. Repeated fields (such as those related to packet length and packet size) and (initial/final) timestamps have been removed to avoid overestimated results.[8] As a result, the employed feature set consists of 74 statistics comprising: (i) flow direction and duration, (ii) *packet length* (`PL`) and *inter-arrival time* (`IAT`) statistics (mean, min, max, median, quartiles, etc.), (iii) TCP[9] and IP header-related features (window size, sequence number, TCP and IP options, etc.), (iv) number of Tx/Rx bytes and packets, (v) number of distinct hosts connected to flow source or destination IP (during its lifetime), (vi) number of concurrent flows sharing the same (source IP, destination IP) pair (regardless of source & destination ports).[10] Differently, `EarlyTC_set` is made of the sequence of pairs `(PL, IAT)` of the first K packets of each flow. In the rest of the paper, we will employ `TC_set` when referring to standard TC, whereas adoption of `EarlyTC_set` will be assumed just for early TC.

Finally, for `TC_set` features, we consider feature selection, based on a *filtering approach*, ranking the elements of the set based on the relative importance of each feature (so as to skim the more informative ones), in terms of *mutual information* with the class (random) variable, whereas for `EarlyTC_set` features, ranking is performed according to a time-constraint (i.e. only the first K packets are employed). We remark that, for the `TC_set`, feature extraction techniques, such as PCA, could be easily adopted in the proposed HC framework without any substantial change.

Classification Algorithms

Herein, we consider as potential nodes *four* different ML-based classifiers, i.e. the C4.5, the RF, the NB, and the BN. Indeed, these classifiers have been successfully employed in several works tackling TC of anonymous traffic [20, 21, 23]. Nonetheless, as the proposed HC framework is general, other ML (e.g., SVM, Gradient Boosting, etc.) or even DL classifiers could be adopted with no substantial change.

Specifically, the *first two* belong to the family of *decision trees*, whereas the *second two* to the *Bayesian family*. In brief, C4.5 is an algorithm employed to generate a decision tree used for classification based on the distribution entropy concept and trained via a top-down recursive and greedy procedure, whereas RF is a classifier based on an ensemble of different decision trees built at training time exploiting the ideas of "bootstrap aggregating" and random-feature selection to mitigate overfitting. Differently, NB is a simple probabilistic classifier that assumes class conditional independence of the features, being not the case for real-world problems, but working well in practice and leading to low complexity. Such stringent assumption is mitigated by the BN classifier, modeling dependence relationships between features and classes usually via a *direct acyclic graph*. In this work, for the latter two algorithms we will adopt the variants shown to perform better in [5], i.e. NB_SD (supervised discretization is used for numerical features) and BN_TAN (the dependence model is reduced to a simpler tree).

3.3 Performance Metrics

Performance evaluation process is based on the following main performance metrics: accuracy, precision (prec), recall (rec), and specificity (spec). For conciseness, out of the latter three metrics, we consider the *F-measure*, $\mathrm{F} \triangleq (2 \cdot \mathrm{prec} \cdot \mathrm{rec})/(\mathrm{prec} + \mathrm{rec})$, and the *G-mean*, $\mathrm{G} \triangleq \sqrt{\mathrm{rec} \cdot \mathrm{spec}}$. Since these two arise from three metrics defined on a per-class basis, we employ their arithmetically averaged (viz. *macro*) versions. Also, *confusion matrices* (breaking the results down by class) are leveraged to provide a representation of the results of the investigated classification approaches, also

5. https://github.com/NM2/hierarchical-tc-at.

6. Up to our knowledge, there is no work in literature tackling anonymous TC at this granularity.

7. We highlight that other feature sets (e.g. histograms), leading to lower performance, have been considered in [5].

8. Timestamp values depend upon the process adopted for collecting traces, potentially introducing to classification artifacts.

9. TCP-related features have zero-value if the flow leverages UDP as transport protocol (e.g., I2P network works on both TCP and UDP).

10. We point to the user manual of Tranalyzer2 (https://tranalyzer.com) for further details about these features.

highlighting misclassification patterns at fine grain. To detect *performance bottlenecks* of our HC approach, we also provide *per-node metrics* (i.e. not considering classification errors introduced by upper levels), also deriving useful guidelines for system design and evaluation.

Considering that in our design each classifier implements a *reject option*, we also deepen the impact of this design choice on performance. In more detail, we investigate the impact of varying the thresholds (see Sec. 2.1), whose tuning can be effective to improve classification performance trading it off with the reduction of the classified instances (viz. the ratio of classified instances, CR).

For each considered analysis, our evaluation is based on a *(stratified) ten-fold cross-validation*, representing a stable performance evaluation setup. As a consequence, we report both the mean and the standard deviation (in the form of a $\pm 3\sigma$ interval, corresponding to 99.7% confidence under a Gaussian assumption) of each performance measure as a result of the evaluation on the ten different folds.

4 Experimental Evaluation

In this section, we show experimental results aimed at investigating anonymous TC performance via the proposed hierarchical framework, also when considering early TC. First, the performance of the proposed approach is analyzed and compared with the best flat classifier, showing that the usage of a "naive" hierarchical architecture is able to introduce non-negligible improvements, being the result of decomposition in *simpler TC sub-tasks*. Then, the results of design improvements are shown, deepening the impact of both rough- and fine-grained optimization choices—the former consisting in varying the number of features of the classifiers (keeping the classifier type fixed) in the hierarchy with the same increment, while the latter involving changes to both features and classifier types. Concerning the early-TC scenario, only final results pertaining to fine-grained optimization are reported, for brevity. Also, the evaluation contains finer-grained analyses of the error patterns of these two different classification "philosophies" along with the severity of errors (thus leading to interesting conclusions on the ATs considered). Finally, a first investigation of classification performance obtained by resorting to progressive censoring in the hierarchical case is reported, and compared with the effects of censoring on a flat classifier baseline.

Naive Hierarchical vs. Best Flat Classifier

In Tab. 1 we report the performance of the best flat classifiers (e.g., the configurations with an optimal number of features in terms of F-measure) as derived from [5] and resulting in a RF fed with 50, 35, and 65 features at L1, L2, and L3, respectively. Performance is reported in terms of accuracy, F-measure, and G-mean at each classification level.[11] Such optimal setup is compared with a first naive implementation of our HC approach, obtained by using the best L3 flat configuration (RF + 65 features) in *all* the classifier nodes of the hierarchy. Unsurprisingly, L1 performance metrics report a score $\geq 99.7\%$: traffic generated through different anonymity networks is easily distinguishable from each other, confirming that these tools are designed to provide anonymity but not to hide the usage of the tool itself. Also, results show that even a naive hierarchical solution improves L3 performance (up to $+2.68\%$ F-measure)[12] w.r.t. the best L3 flat classifier, and performs on a par in terms of L1–L2 levels when compared to level-optimized best flat classifiers. This is due to split of the original TC task (21 classes at L3) into smaller tasks (at most, 5 classes).

Table 1: Accuracy, F-measure, and G-mean (%) of the best flat classifier (RF) with {optimal number of features} at each level compared to naive hierarchical configuration. Results are in the format *avg. (± std.)* over 10-folds.
Legend: Best Accuracy ($\star$), F-measure (†), and G-mean ($\diamond$) per level.

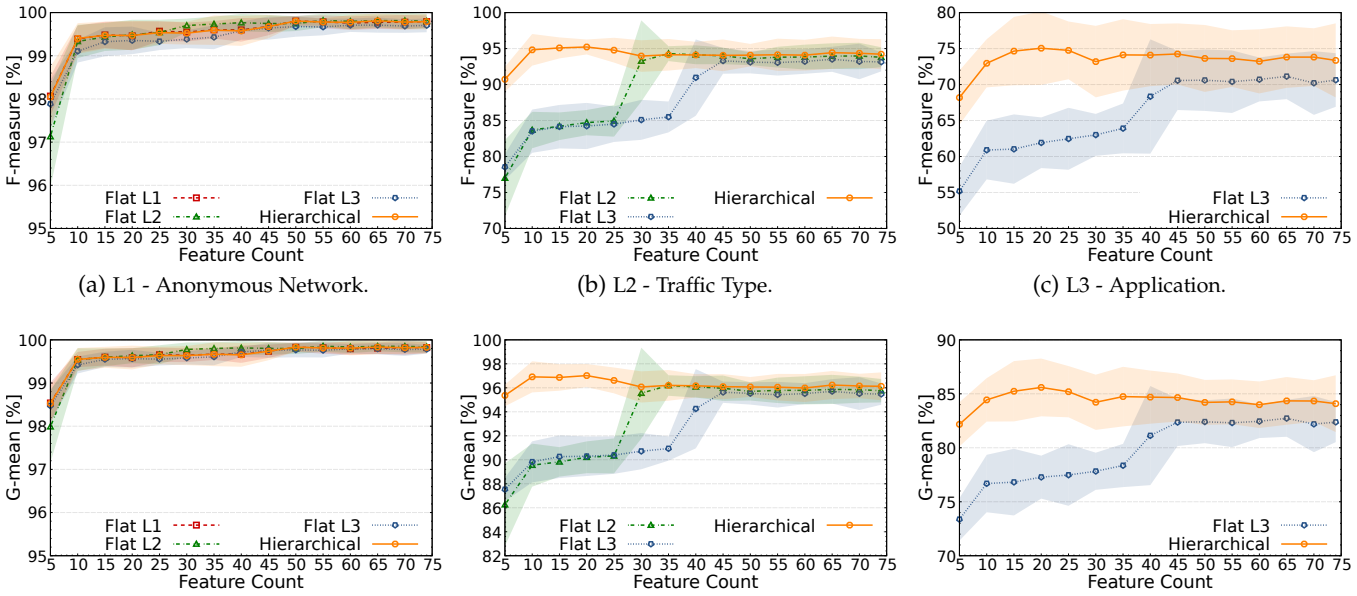
Classifier	Metric	L1	L2	L3
Best Flat L1 {50}	*Accuracy*	99.80±0.03%	n.d.	n.d.
	F-measure	99.80±0.04%	n.d.	n.d.
	G-mean	99.83±0.03% $\diamond$	n.d.	n.d.
Best Flat L2 {35}	*Accuracy*	99.75±0.06%	97.01±0.24%	n.d.
	F-measure	99.73±0.06%	94.30±0.35%	n.d.
	G-mean	99.80±0.05%	96.19±0.29%	n.d.
Best Flat L3 {65}	*Accuracy*	99.70±0.06%	96.77±0.24%	73.52±0.40%
	F-measure	99.71±0.06%	93.51±0.58%	71.14±1.05%
	G-mean	99.79±0.04%	95.71±0.34%	82.73±0.57%
Naive Hierarchical {65}	*Accuracy*	99.81±0.06% $\star$	97.17±0.24% $\star$	74.60±0.48% $\star$
	F-measure	99.81±0.06% †	94.43±0.75% †	73.82±1.42% †
	G-mean	99.83±0.05%	96.23±0.39% $\diamond$	84.35±0.74% $\diamond$

Impact of Feature Selection

As a first step towards the optimization of the proposed approach, we investigate here the performance of the framework when varying the number of features used by each classifier but *considering a common number of features (here denoted M) for each node in the tree*, and *keeping the classification algorithm fixed to RF* (that is, the best-performing one in the flat case [5]). We remark that although there is a common M for all nodes, the specific set of features *may differ*.

Fig. 3 summarizes the obtained results—with a view to finding the optimal value *M for the entire hierarchy*—also providing a comparison against the three flat classifier counterparts (in terms of F-measure and G-mean, as accuracy showed similar trends). First, results highlight that there is no appreciable performance difference among L1–L3 flat classifiers and the hierarchical approach by looking at L1 metrics, and only a slight performance improvement is observed with a high number of features (performance saturation is observed with at least 10 features) is achieved. On the other hand, at L2 slightly higher performance are obtained by the hierarchical approach with a lower number of employed features per node (approximately 10–20), whereas at L3 (being the harder task) the following key observations can be made: (i) the hierarchical approach provides a non-negligible improvement over the L3 flat classifier for all the range considered, (ii) the best performance of the HC is attained with a smaller number of features. These considerations apply to both F-measure and G-mean.

11. "n.d." points out unavailable performance for flat classifiers at levels deeper than that considered for classification.



(a) L1 - Anonymous Network. (b) L2 - Traffic Type. (c) L3 - Application.

(d) L1 - Anonymous Network. (e) L2 - Traffic Type. (f) L3 - Application.

Figure 3: F-measure (a–c) and G-mean (d–f) [%] of hierarchical and flat classifiers: RF fed with different subsets of features in `TC_set` (from 5 to 74 with increments of 5). Average on 10-folds and corresponding $\pm 3\sigma$ confidence interval are shown.

Fine-grained Optimization

Herein the fine-grained framework optimization, in case of features from `TC_set`, is discussed. In detail, with the aim of trading off design complexity with performance, *we remove the constraints previously introduced, and allow each node in the hierarchy to be optimized in terms of both the number of features and the classifier type*. As remarked in Sec. 3.2, we consider *four* different ML-based classifiers (C4.5, RF, NB_SD and BN_TAN). We remark that, to avoid a combinatorial explosion of the optimization problem, we resort to the *per-node* optimization rationale described in Sec. 2.1. Based on the above rationale, in Fig. 4 we report the classification performance in terms of accuracy (Fig. 4a), F-measure (Fig. 4b) and G-mean (Fig. 4c), by comparing the flat classification approaches with the per-node optimized hierarchical classifier. The proposed (optimized) hierarchical classifier is able, at least, to perform at t^{th} ($t = 1, 2, 3$) level as well as the corresponding flat classifier *explicitly designed* to solve the classification task of the same level. In detail, such optimized hierarchical approach is able to achieve 99.81% (resp. 99.83%), 95.81% (resp. 97.44%) and 75.56% (resp. 85.89%) F-measure (resp. G-mean) score at L1, L2, and L3, respectively. These results (almost) represent a tie at L1, whereas $+1.51\%$ (resp. $+1.73\%$) and $+4.42\%$ (resp. $+3.16\%$) gains are experienced at L2 and L3, respectively.[13] The details of optimized HC are reported in Fig. 5a, where for each classifier node the employed classifier and the number of features are reported. Remarkably, RF denotes the best classifier for each node-specific classification task, while only for the classifier of *Tor App* applications BN_TAN provides higher performance. Instead, the variability of the optimal number of features underlines no clear trend, except that usually I2P-related node classifiers require a lower number of features, at least when leveraging those in `TC_set`.

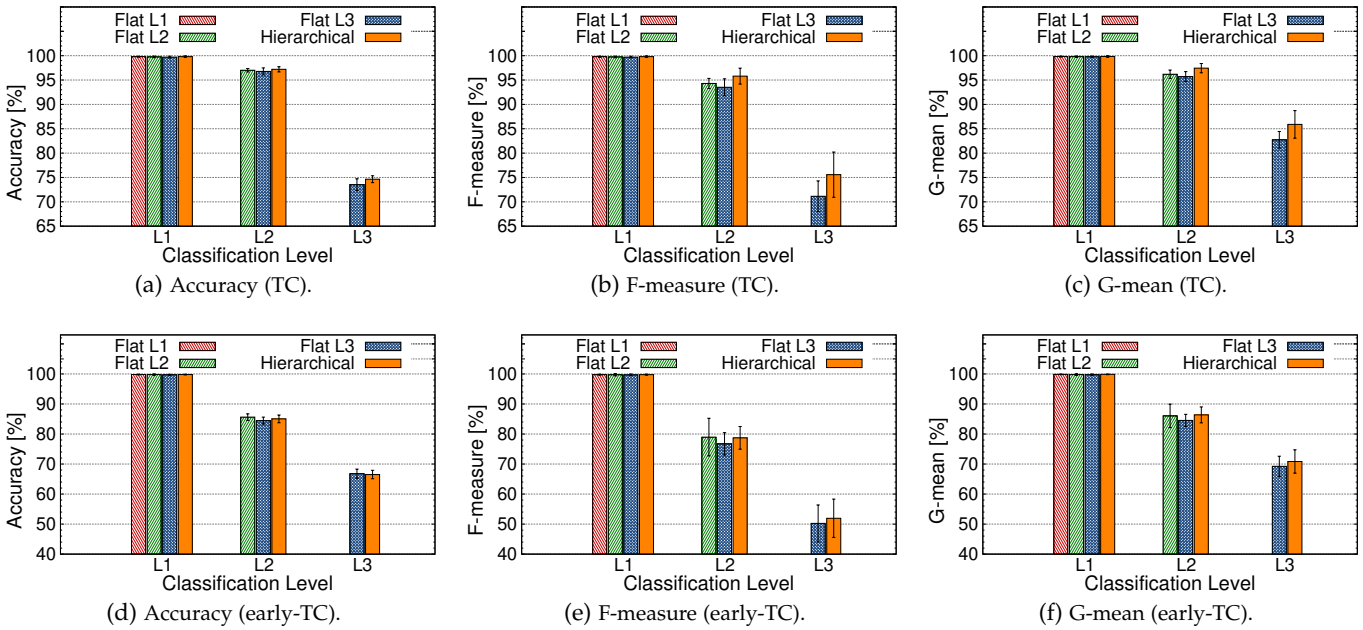
Optimization for Early TC

Herein we evaluate the hierarchical framework when the classifiers are fed with features in `EarlyTC_set` (see Sec. 3.2), i.e. considering `PL`s and `IAT`s of the first $K = 1, \ldots, 16$ (non-zero payload) packets. This analysis helps assessing the framework capability in supporting early TC, i.e. to evaluate how soon and to which degree ATs and related services can be identified. To this end, we have paralleled the previous investigations in the early-TC scenario. Nonetheless, herein we omit the details for brevity, and comment only the final results. We remark that the *main difference* with the previous analysis concerns the *feature selection* process, herein performed on a *time-basis* (i.e. only the features drawn from the first K packets are considered).

First, results show that a naive hierarchical "extension" of best flat L3 classifier (in this case a `BN_TAN` with $K = 11$ packets) is *not* able to provide improved performance (e.g. 48.80% F-measure at L3, as opposed to 50.23% in the flat case). This result highlights a key difference with respect to the scenario leveraging `TC_set` and emphasizes the *need for hierarchical-specific optimization* in such case. Secondly, we investigate (as in Fig. 3) how varying the features corresponding to the first K packets could improve the performance of the naive hierarchical extension, and compare it with the best flat counterpart. Our investigations reveal that a performance saturation is observed after ≈ 10 packets, and that the HC approach is able to improve best flat L3 approach in terms of G-mean, whereas an F-measure drop is observed for all values of K considered. Finally, fine-grained optimization (see Figs. 4d, 4e, and 4f) provides

12. Also, from a statistical significance viewpoint, we observed a gain of HC approach in 100% of the cases over the considered folds.

13. Also, from a statistical significance viewpoint, we observed a gain of HC approach in 97.5% of the cases over the considered folds.



(a) Accuracy (TC). (b) F-measure (TC). (c) G-mean (TC).

(d) Accuracy (early-TC). (e) F-measure (early-TC). (f) G-mean (early-TC).

Figure 4: Accuracy (a), F-measure (b) and G-mean (c) of the best classifiers and of their early-TC counterparts (d, e, and f). Average on 10-folds and corresponding $\pm 3\sigma$ confidence interval are shown.

higher performance with respect to the optimized flat case, e.g., $+1.71\%$ F-measure and $+1.59\%$ G-mean at L3.[14]

The corresponding optimized hierarchical structure is shown in Fig. 5b and—when compared to Fig. 5a—clearly shows that per-node optimized classifiers significantly differ in type and number of features, thus motivating the need for fine-tuned optimization of the proposed HC. Although the HC gain is not significant, its operating principle allows to delve into the "information structure" of the TC problem and highlight critical points, by excluding potential performance drops due to the size of the classification task, as shown by the following *per-node* performance analysis.

Per-node Detailed Classification Performance

For completeness, we report in Figs. 5c and 5d the detailed per-node classification performance, corresponding to `TC_set` and `EarlyTC_set`, respectively.

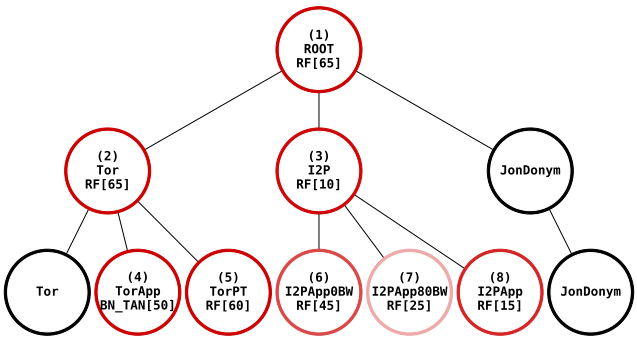
The following interesting observations can be made on the reported tree representation. First, with respect to *Anonymous Network Level* classification (L1) nodes present near-ideal performance both when relying on `TC_set` and `EarlyTC_set`, thus showing *(almost) no errors propagating from HC of ATs*. Secondly, at *Traffic Type Level* (L2) both approaches based on `TC_set` and `EarlyTC_set` present near-ideal performance in classifying *Tor* traffic types, whereas some performance degradation is observed in classification of I2P traffic types; this phenomenon is more penalizing in the early-TC case, e.g., with *I2P* F-measure dropping down to 60.61%. This represents one of the main causes of performance difference among the two scenarios, as all these *errors are propagated downwards*. Finally, at *Application Level* (L3), the behavior of the classifier nodes is more varied. Indeed, referring to approach based on `TC_set`, it is apparent that Tor nodes (`TorApp` and `TorPT`) work well, whereas on I2P nodes significant degradations (higher than those at L2) are observed, with *I2PApp80BW* the *most significant*. Differently, discrimination within each I2P traffic type and *TorApp* is only possible with $< 65\%$ F-measure with features in `EarlyTC_set`. Therefore, classification within *I2PApp80BW* cannot be accurately attained with neither of the considered feature sets (confirming the intuition that *I2PApp80BW* represents traffic obtained by mixing different apps), and the advantage of the classification task split into sub-problems cannot exceed a certain threshold in this case, due to the impossibility of discerning applications within this traffic type, resorting to the set of the available features.

14. Also, from a statistical significance viewpoint, we observed a gain of HC approach in 90.0% of the cases over the considered folds.

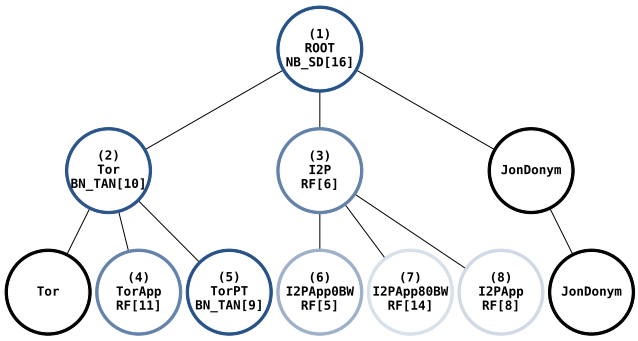
Per-Class Performance Breakdown

Since classification at L3 is a challenging task (but also the most interesting from a user's privacy perspective), we report in Fig. 6 the per-class performance breakdown of HC results in terms of confusion matrices at L3, comparing them against results obtained with flat approach. We recall that for these matrices the higher the concentration toward the main diagonal, the better the overall performance. Further, to highlight how errors at L3 which do not imply misclassifications at L1/L2 are *less severe* and should be *promoted* as opposed to the others, in the same figures we also highlight the error patterns corresponding to the same traffic type (solid boxes) and the same AT (dashed boxes).

First, comparing flat and hierarchical approaches (Figs. 6a and 6b, respectively), we can observe a *reduction of error-patterns* in the latter case, highlighting the beneficial "divide-et-impera" principle of HC. Secondly, confusion matrices at L1 (dashed boxes) show that classifiers based on both approaches (with different quantitative outcomes) present error patterns which almost entirely lead to a misclassification of the traffic type *within the same*



(a) Fine-grained optimized hierarchical structure **with `TC_set`**.



(b) Fine-grained optimized hierarchical structure **with `EarlyTC_set`**.

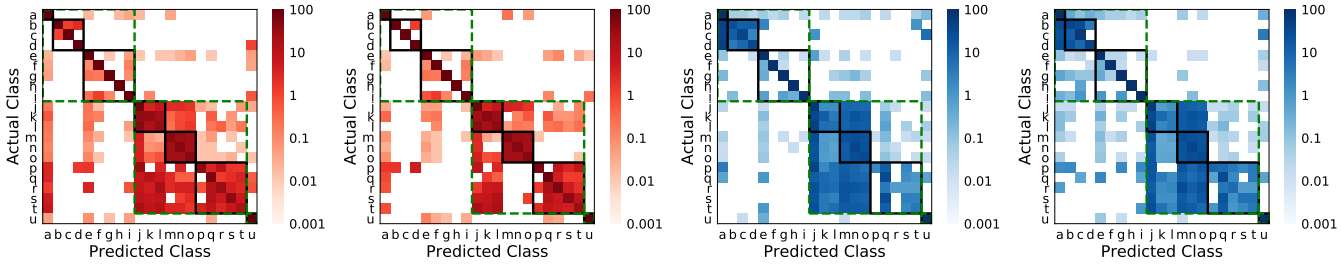
Classifier	#Classes	Accuracy	F-measure	G-mean
$C_0 \rightarrow ①$	$L_1 = 3$	99.81 ($\pm$ 0.06)	99.81 ($\pm$ 0.06)	99.83 ($\pm$ 0.05)
$C_1 \rightarrow ②$	$L_1 = 3$	99.97 ($\pm$ 0.04)	99.91 ($\pm$ 0.20)	99.97 ($\pm$ 0.04)
$C_2 \rightarrow ③$	$L_2 = 3$	95.05 ($\pm$ 0.29)	91.53 ($\pm$ 1.06)	93.29 ($\pm$ 0.79)
$C_{11} \rightarrow ④$	$L_{11} = 3$	99.58 ($\pm$ 1.25)	99.58 ($\pm$ 1.25)	99.69 ($\pm$ 0.94)
$C_{12} \rightarrow ⑤$	$L_{12} = 5$	99.72 ($\pm$ 0.12)	99.44 ($\pm$ 0.21)	99.63 ($\pm$ 0.14)
$C_{21} \rightarrow ⑥$	$L_{21} = 3$	72.42 ($\pm$ 1.32)	58.43 ($\pm$ 1.63)	69.00 ($\pm$ 1.26)
$C_{22} \rightarrow ⑦$	$L_{22} = 3$	48.94 ($\pm$ 0.51)	48.90 ($\pm$ 0.52)	60.37 ($\pm$ 0.42)
$C_{23} \rightarrow ⑧$	$L_{23} = 5$	75.12 ($\pm$ 5.95)	72.50 ($\pm$ 6.99)	81.68 ($\pm$ 4.58)

(c) Performance metrics **with `TC_set`**.

Classifier	#Classes	Accuracy	F-measure	G-mean
$C_0 \rightarrow ①$	$L_1 = 3$	99.80 ($\pm$ 0.05)	99.78 ($\pm$ 0.06)	99.87 ($\pm$ 0.04)
$C_1 \rightarrow ②$	$L_1 = 3$	99.20 ($\pm$ 0.12)	90.48 ($\pm$ 1.61)	94.82 ($\pm$ 1.27)
$C_2 \rightarrow ③$	$L_2 = 3$	72.20 ($\pm$ 0.81)	60.61 ($\pm$ 1.70)	66.85 ($\pm$ 1.02)
$C_{11} \rightarrow ④$	$L_{11} = 3$	63.42 ($\pm$ 4.92)	63.27 ($\pm$ 4.92)	72.01 ($\pm$ 3.85)
$C_{12} \rightarrow ⑤$	$L_{12} = 5$	99.69 ($\pm$ 0.07)	99.55 ($\pm$ 0.18)	99.63 ($\pm$ 0.13)
$C_{21} \rightarrow ⑥$	$L_{21} = 3$	67.37 ($\pm$ 0.81)	44.56 ($\pm$ 1.45)	56.38 ($\pm$ 0.97)
$C_{22} \rightarrow ⑦$	$L_{22} = 3$	43.47 ($\pm$ 0.75)	43.13 ($\pm$ 0.72)	55.76 ($\pm$ 0.63)
$C_{23} \rightarrow ⑧$	$L_{23} = 5$	50.67 ($\pm$ 9.21)	37.75 ($\pm$ 11.24)	58.04 ($\pm$ 7.81)

(d) Performance metrics with **with `EarlyTC_set`**.

Figure 5: Fine-grained optimized hierarchical structure with (a) `TC_set` and (b) `EarlyTC_set`. Optimal number of features for each classifier (node) is shown in square brackets. Lighter colors point to worse performance. Related per-node metrics are shown in (c) and (d), respectively (with $< 60\%$ F-measure nodes highlighted in gray).



(a) Flat TC applied to L3. (b) Hierarchical TC at L3. (c) Flat early TC applied to L3. (d) Hierarchical early TC at L3.

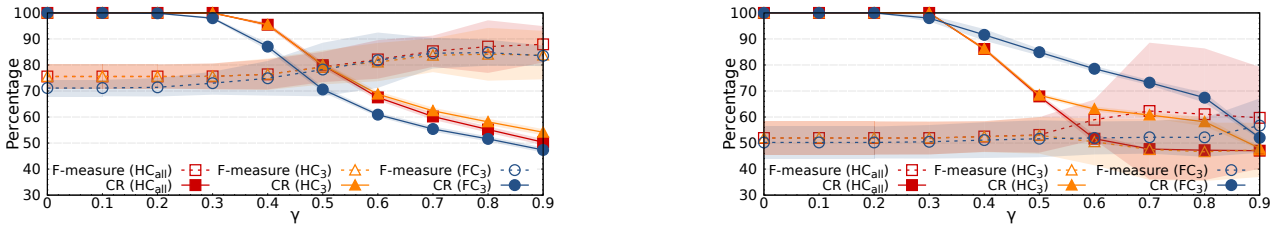
Figure 6: L3 Confusion matrices ([%] in log scale) of best flat and hierarchical classifiers in standard TC (red) and early TC (blue).

anonymous network. Differently, referring to L2 standpoint (solid boxes), HC provides *improved capabilities in confining errors to the same traffic type* (especially in the case of I2P traffic). A similar consideration applies to early TC results (Figs. 6c and 6d) and, in particular, to errors concerning the applications of *I2P Apps* and *Tor Apps*. Hence HC approach performance clearly highlights the inability, *not depending on the size of the classification task*, in satisfactorily discriminating (with an early TC setup in mind) among I2P traffic types and within their corresponding applications, along with those in `Tor Apps`. On the other hand, results confirm the outcomes of [30], witnessing that obfuscation implemented by `Tor Pluggable Transports` induces a class fingerprint easily distinguishable ($\geq 99\%$ accuracy, see C_2 classifier in Fig. 5) from both `Normal Tor Traffic` and `Tor Apps`.

Performance with Reject Option

Finally, in Fig. 7 we focus on the adoption of censoring threshold(s) for flat classification and HC of ATs. Specifically, we report L3 performance, being the hardest task considered. We recall that, although in the flat case there is only a single tunable γ (referred to as FC_3), HC allows to set a different threshold value at each node (e.g. for C_{ij} the threshold γ_{ij} can be adjusted independently from the others, see Sec. 2.1). Nonetheless, as a preliminary investigation toward the censored behavior of HC approach—to avoid cumbersome analyses—we consider two simplified options: considering only a common γ value shared by (a) by *all the nodes* in the hierarchy (HC_{all}) and (b) *solely* by the classifier nodes concurring to L3 classification (HC_3). Accordingly, we report the F-measure vs. γ (similar trends have been observed for other metrics) along with the corresponding trend of the ratio of classified samples (CR).

Although using such thresholds cannot be intended as a *panacea* able to cope with high-confidence wrong predictions, results pertaining to the adoption of `TC_set` show performance improvement with increasing γ. In detail, both hierarchical variants offer performance gain with higher CR (less discarded flows) w.r.t. FC_3 (e.g. $\approx 80\%$ F-measure is attained with $\approx 80\%$ CR), with HC_{all} having slightly



(a) TC. (b) Early TC.

Figure 7: F-measures and Classified Ratios (CR) of best classifiers vs. γ.

improved performance (while incurring in a slightly higher CR, due to the presence of non-zero thresholds for nodes at higher levels in the hierarchy). On the other hand, in the early-TC scenario, the CR is lower for hierarchical approaches, while HC_{all} provides slightly improved performance than FC_3, whose performance do not benefit from a γ increase. Nonetheless, such results underline how progressive censoring via per-node thresholds may be a viable option for further performance improvement.

5 Conclusions and Further Directions

In this paper, we tackled TC of Tor, I2P, and JonDonym via a hierarchical approach that, beyond classification performance gains, carries *by design several advantages* in terms of modularity, training efficiency, distributed deployment, and "tunable view" of classification outputs. In detail, the proposed framework was designed with varying constraints, resulting in implementations with different degrees of complexity (in terms of classifiers, features, and reject option).

Our experimental analysis—carried on the Anon17 public dataset—allowed reasoning on which degree ATs traffic can be told apart, considering different granularities such as the *anonymity network* adopted (L1), the *traffic type* tunneled in the network (L2), and the *application* category generating such traffic (L3). Moving from naive to fine-grain optimized implementations for the hierarchical classifier, results at different granularities highlighted how HC guarantees interesting performance gains, especially at the finest level (L3). More specifically, applications were classified with up to 75.56% F-measure, obtaining a gain of up to $+4.5\%$ with respect to an optimized flat implementation [5].

Further, it was proved that these errors can be mitigated further with outcome censoring mechanisms (avoiding error propagation), e.g. by discarding $\approx 20\%$ of classified flows, the proposed approach is able to guarantee $\approx 80\%$ F-measure. Also, *per-node* and *per-class* performance evaluation allowed to delve into misclassification patterns and their severity, proving that HC approach better confines misclassifications in the same AT and even the same traffic type. These evaluations witnessed the appeal and effectiveness of HC framework also in real environments and provided insights in identifying performance bottlenecks which lie on a very limited set of nodes in the hierarchy. Specifically, applications running within I2P correspond to the *hardest to be classified each other*, thus confirming the common thinking that such AT provides a higher privacy level (from TC viewpoint) w.r.t. Tor, at the expenses of increased latency.

The proposed HC framework suggests the following future directions: (i) use of *sophisticated* nodes and alternative hierarchical approaches; (ii) advanced HC optimization, e.g. automatic computation of reject thresholds possibly differing at each node; (iii) hierarchical architectures having nodes operating with different TC objects and with applications arranged in "multi-view" structures; (iv) HC implementation in the community-oriented TC platform TIE [31], allowing live (anonymous and not) traffic evaluation.

References

[1] G. Aceto and A. Pescapé, "Internet censorship detection: A survey," *Computer Networks*, vol. 83, pp. 381–421, 2015.

[2] A. Dainotti, A. Pescapé, and K. C. Claffy, "Issues and future directions in traffic classification," *IEEE Network*, vol. 26, no. 1, 2012.

[3] G. Aceto, D. Ciuonzo, A. Montieri, and A. Pescapé, "Multi-classification approaches for classifying mobile app traffic," *Journal of Network and Computer Applications*, vol. 103, pp. 131–145, 2018.

[4] A. Dainotti, A. Pescapé, and G. Ventre, "Worm traffic analysis and characterization," in *IEEE ICC*, 2007, pp. 1435–1442.

[5] A. Montieri, D. Ciuonzo, G. Aceto, and A. Pescape, "Anonymity services Tor, I2P, JonDonym: Classifying in the dark (web)," *IEEE Trans. Depend. Sec. Comput.*, pp. 1–1, 2018.

[6] N. Cascarano, A. Este, F. Gringoli, F. Risso, and L. Salgarelli, "An experimental evaluation of the computational cost of a DPI traffic classifier," in *IEEE GLOBECOM*, 2009, pp. 1–8.

[7] J. Yu, H. Lee, Y. Im, M.-S. Kim, and D. Park, "Real-time classification of internet application traffic using a hierarchical multi-class SVM," *KSII Transactions on Internet & Information Systems*, vol. 4, no. 5, 2010.

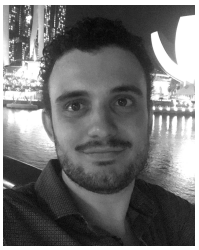
[8] L. Grimaudo, M. Mellia, and E. Baralis, "Hierarchical learning for fine grained internet traffic classification," in *IEEE IWCMC*, 2012, pp. 463–468.

[9] J. h. Kim, S.-H. Yoon, and M.-S. Kim, "Study on traffic classification taxonomy for multilateral and hierarchical traffic classification," in *IEEE APNOMS*, 2012, pp. 1–4.

[10] S.-H. Yoon, K.-S. Shim, S.-K. Lee, and M.-S. Kim, "Framework for multi-level application traffic identification," in *IEEE APNOMS*, 2015, pp. 424–427.

[11] Y. n. Dong, J. j. Zhao, and J. Jin, "Novel feature selection and classification of internet video traffic based on a hierarchical scheme," *Computer Networks*, vol. 119, pp. 102–111, 2017.

[12] W. M. Shbair, T. Cholez, J. François, and I. Chrisment, "A multi-level framework to identify HTTPS services," in *IEEE/IFIP NOMS*, 2016, pp. 240–248.
[13] J. Barker, P. Hannay, and P. Szewczyk, "Using traffic analysis to identify the second generation onion router," in *IEEE/IFIP EUC*, 2011, pp. 72–78.
[14] D. Herrmann, R. Wendolsky, and H. Federrath, "Website fingerprinting: attacking popular privacy enhancing technologies with the multinomial Naïve-Bayes classifier," in *ACM CCSW*, 2009, pp. 31–42.
[15] A. Panchenko, L. Niessen, A. Zinnen, and T. Engel, "Website fingerprinting in onion routing based anonymization networks," in *ACM WPES*, 2011, pp. 103–114.
[16] A. Panchenko, F. Lanze, J. Pennekamp, T. Engel, A. Zinnen, M. Henze, and K. Wehrle, "Website fingerprinting at internet scale," in *NDSS Symposium*, 2016.
[17] V. Rimmer, D. Preuveneers, M. Juarez, T. Van Goethem, and W. Joosen, "Automated feature extraction for website fingerprinting through deep learning," in *NDSS Symposium*, 2018.
[18] P. Sirinam, M. Imani, M. Juarez, and M. Wright, "Deep fingerprinting: Undermining website fingerprinting defenses with deep learning," in *ACM SIGSAC CCS*, 2018, pp. 1928–1943.
[19] G. He, M. Yang, J. Luo, and X. Gu, "Inferring application type information from tor encrypted traffic," in *IEEE CBD*, 2014, pp. 220–227.
[20] M. AlSabah, K. Bauer, and I. Goldberg, "Enhancing Tor's performance using real-time traffic classification," in *ACM CCS*, 2012, pp. 73–84.
[21] K. Shahbar and A. N. Zincir-Heywood, "Benchmarking two techniques for Tor classification: Flow level and circuit level classification," in *IEEE CICS*, 2014, pp. 1–8.
[22] ——, "An analysis of Tor pluggable transports under adversarial conditions," in *IEEE CISDA*, 2017.
[23] ——, "Effects of shared bandwidth on anonymity of the I2P network users," in *IEEE WTMC*, 2017, pp. 235–240.
[24] ——, "Packet momentum for identification of anonymity networks," *Journal of Cyber Security and Mobility*, vol. 6, no. 1, pp. 27–56, 2017.
[25] A. H. Lashkari, G. Draper-Gil, M. Mamun, I. Saiful, and A. A. Ghorbani, "Characterization of Tor traffic using time based features," in *SCITEPRESS ICISSP*, 2017, pp. 253–262.
[26] L. Bernaille, R. Teixeira, I. Akodkenou, A. Soule, and K. Salamatian, "Traffic classification on the fly," *ACM SIGCOMM Computer Communication Review*, vol. 36, no. 2, pp. 23–26, 2006.
[27] C. N. Silla and A. A. Freitas, "A survey of hierarchical classification across different application domains," *Data Mining and Knowledge Discovery*, vol. 22, no. 1-2, pp. 31–72, 2011.
[28] V. F. Taylor, R. Spolaor, M. Conti, and I. Martinovic, "Robust smartphone app identification via encrypted network traffic analysis," *IEEE Trans. Inf. Forensics Security*, vol. 13, no. 1, pp. 63–78, Jan 2018.
[29] S. Burschka and B. Dupasquier, "Tranalyzer: Versatile high performance network traffic analyser," in *IEEE SSCI*, 2016, pp. 1–8.
[30] K. Shahbar and A. N. Zincir-Heywood, "Traffic flow analysis of Tor pluggable transports," in *IEEE CNSM*, 2015, pp. 178–181.
[31] W. De Donato, A. Pescapé, and A. Dainotti, "Traffic identification engine: an open platform for traffic classification," *IEEE Network*, vol. 28, no. 2, pp. 56–64, 2014.



Antonio Montieri (GSM'18) is a PhD Student at DIETI, University of Napoli Federico II, since 2017. He has received his MS Degree from the same University in 2015. His work is focused on network measurements, (encrypted and mobile) traffic classification and modeling, monitoring of cloud network performance. Antonio has co-authored 16 papers and 5 posters accepted for publication in international journals and conference proceedings.

Domenico Ciuonzo (S'11-M'14-SM'16) is an Assistant Professor at DIETI, University of Napoli "Federico II". He holds a Ph.D. in Electronic Engineering from the University of Campania "L. Vanvitelli". Since 2014, he has been in the editorial board of different IEEE and Elsevier journals. His work focuses on data fusion, wireless sensor networks, machine learning and network analytics.

Giampaolo Bovenzi is a PhD Student in Information Technology and Electrical Engineering at DIETI, University of Napoli "Federico II", since November 2018. He achieved MS degree (summa cum laude) at the same University in October 2018. His research interests focus on (anonymized and encrypted) traffic classification, network security (with applications to IoT), and blockchain.

Valerio Persico is a Post Doc at DIETI, University of Napoli "Federico II", where he received a Ph.D. in Computer and Automation Engineering in 2016. His work focuses on Internet measurements, cloud network performance, and traffic analysis and classification. He is the recipient of the best student paper award at ACM CoNext 2013.

Antonio Pescapè (SM'09) is a Full Professor of computer engineering at the University of Napoli "Federico II". His work focuses on Internet technologies and specifically on measurement, monitoring, and analysis of the Internet. Recently, he is working on bioinformatic and ICTs for a smarter health. Antonio has co-authored more than 200 conference and journal papers and is the recipient of a number of research awards.