`AppClassNet`: A commercial-grade dataset for application identification research

Chao Wang, Alessandro Finamore, Lixuan Yang, Kevin Fauvel, Dario Rossi[*]
Huawei Technologies SASU, Paris Research Center, France — `first.last@huawei.com`

ABSTRACT

The recent success of Artificial Intelligence (AI) is rooted into several concomitant factors, namely theoretical progress coupled with abundance of data and computing power. Large companies can take advantage of a deluge of data, typically withhold from the research community due to privacy or business sensitivity concerns, and this is particularly true for networking data. Therefore, the lack of high quality data is often recognized as one of the main factors currently limiting networking research from fully leveraging AI methodologies potential.

Following numerous requests we received from the scientific community, we release `AppClassNet`, a commercial-grade dataset for benchmarking traffic classification and management methodologies. `AppClassNet` is significantly larger than the datasets generally available to the academic community in terms of both the number of samples and classes, and reaches scales similar to the popular ImageNet dataset commonly used in computer vision literature. To avoid leaking user- and business-sensitive information, we opportunely anonymized the dataset, while empirically showing that it still represents a relevant benchmark for algorithmic research. In this paper, we describe the public dataset and our anonymization process. We hope that `AppClassNet` can be instrumental for other researchers to address more complex commercial-grade problems in the broad field of traffic classification and management.

CCS CONCEPTS

• **Networks → Network measurement**; • **Computing methodologies → Machine learning**.

KEYWORDS

Open Dataset, Application Identification, Traffic Classification, Machine Learning, Supervised learning, Neural networks

1 INTRODUCTION

In recent times, Deep Learning (DL) techniques have attained significant advances in the fields of Computer Vision and Pattern Recognition (CVPR) and Natural Language Processing (NLP). Such advances have equally benefited from (*i*) the ground-breaking theoretical research, (*ii*) the availability of open-source software platforms, (*iii*) the creation of efficient hardware accelerators and, last but not least, (*iv*) the availability of *large corpora* training sets. For instance, crowdsourcing efforts in the CVPR field led to the well-known *ImageNet* [1] which comprises *tens of millions of image samples*, each annotated with one among *tens of thousands of class labels*. Similarly, in the NLP field the *Common Crawl* [2] project gathered several *hundreds of billions of text tokens*.

This is in stark contrast with the networking field where a commonly identified limit to AI deployment is the lack of publicly available large scale datasets. We point out that, over the years, several research groups have indeed made commendable effort to release datasets for networking research. In the context of network operation and management (O&M) a quite fundamental building block is constituted by the ability to recognize the type of traffic, or specific application, that generated a particular stream of packets. Different waves of research, in the early 2000s and late 2010s, have tackled this use-case with Machine Learning (ML) and Deep Learning (DL) approaches respectively. As a side effect of the scientific research, a number of datasets have been released (Sec.2). At the same time, while these datasets are valuable to foster repeatability and cross-comparison, academic groups generally lack (*i*) access to real operational networks environment, (*ii*) diverse operational environment (e.g., residential access vs enterprise campuses) and (*iii*) commercial-grade labeling tools.

To help tackling these limitations, we contribute to the research community `AppClassNet`, a commercial-grade dataset for the purpose of application identification. We used the private version of such dataset in our previous work on federated training [70], zero-day application detection [69] and inference acceleration at line rate based on either hardware accelerators to speedup computation [32] or approximated-key caching to save computation altogether [31]. `AppClassNet` is obtained from 10 TB worth of real traffic and comprises over 10 M flows each of which is annotated by a commercial and proprietary DPI system supporting thousands of labels. The public version that we release and describe in this document retains most of the richness of the original dataset (99% of the flows and bytes) of the most popular applications, corresponding to 500 fine-grained application labels. Roughly, `AppClassNet` is the networking equivalent of the popular ImageNet Large Scale Visual Recognition Challenge (ILSVRC) [1], an ImageNet subset comprising 1,000 classes and nearly 1.5M images. As this sheer size is significantly larger than what it is currently available to the academic community, we hope that `AppClassNet` can be an instrumental benchmark for the community and foster research exploration towards more realistic and ambitious challenges.

We point out that making publicly available (a subset of) any private dataset is a delicate process, as opportune anonymization is paramount to ensure that the data released is deprived of both privacy- and business-sensitive information. Likewise, we also need to make sure that the public version of the dataset obtained through such anonymization still fits the original purpose – ML/DL models created with the public and private version of the dataset offer comparable performance. We point out that the goal of this paper is not to propose a new technique for opportune anonymization. Rather, we discuss the pre-processing steps taken to deprive the dataset from sensitive information as (*i*) to avoid a flawed "security

[*]This work was done during Chao Wang final MSc internship in Huawei Paris Research Center

Table 1: Subset of publicly available datasets for traffic classification

Dataset	Year	Classes	Flows	Flows/class	Comment
UniBS [35]	2009	7	78.9k	11.3k	Pcap; Ground truth devised with kernel-level modification
Bujlow et al.[24]	2015	25	559k	22k	Cross-comparison of 4 open-source and 2 commercial DPI tools
Mirage[14]	2019	40	100k	2.5k	Active pcap traffic collection for 40 mobile apps from 380 participants
Cross platform[3]	2012	411	102k	248	Active pcap collection of 411 apps from several students participants
ReCon[51]	2015	512 (+5)	29k (+141 k)	56 (+28k)	Automated collection for top-apps (+multiple versions for 5 apps)
Andrubis[39]	2014	1 M	41M	41	Automated collection of startup traffic for apps from 15 market places
AppClassNet	2022	*200+300*	*9.7M+0.3M*	*48.3k+1.1k*	*200 popular + 300 unpopular applications, covering ≃99% of traffic volume*

by obscurity" approach and (*ii*) to highlight the limitations of the public datasets with respect to the original private one. We believe that AppClassNet is a good enough compromise, as it allows researchers to focus on algorithms design in a challenging network use-case, despite the anonymization deprives most of the common networking domain knowledge from the dataset.

In the following, we first review the literature on datasets and collection methodologies (Sec. 2). Then, we introduce AppClassNet, covering both the anonymization methodology as well as contrasting the characteristic of the private and public version of the dataset (Sec. 3). We next verify that the performance of state of the art ML and DL techniques remains consistent between the private and public version of the dataset (Sec. 4). Finally, we discuss AppClassNet limitations and term of use (Sec. 5) and conclude the paper (Sec. 6).

2 BACKGROUND

The early 2000's witnessed a first wave of traffic classification with Machine Learning (ML) methodologies aiming to identify fine-grained applications labels (e.g., YouTube, Skype, PokemonGo) or coarse-grained services (e.g., video streaming, video call, gaming). Then, the late 2010 have seen an explosion of interest toward Deep Learning (DL) methodologies, essentially solving the same problem but with a different set of techniques (see Sec.2.1)

The need for common benchmarks has pushed researchers towards adoption of open-source datasets created to either address a specific goal or as a side product of their research. In this section, we review the relevant datasets and methods for traffic classification and management focusing more on the adopted input rather than the ML/DL tested on such data.

2.1 Use-case viewpoint

The goal of this paper is not to present a full background on the application identification literature, but rather to discuss which type of work could benefit from AppClassNet. The first wave of traffic classification, well surveyed in [47], employed "classic" ML techniques. As we overviewed in [69], such techniques adopted an input based on either engineered Flow Features (FF) [45] or Packet Payload features (PP) [44, 18], and culminated with the adoption of simple, yet effective, Time Series (TS) features based on properties of the first packets of a flow such as packet size and direction [17] (and seldom interarrival time [17, 28]). These lightweight TS approaches are particularly appealing since they (*i*) operate "early" at the beginning of a flow, as opposite to "post mortem" techniques which compute FF after a flow ends, and as such are suitable for traffic management and (*ii*) sustain line rate operation with limited computational costs [29].

The second wave of traffic classification [48, 20] employs DL techniques and essentially re-considers all inputs features previously introduced — from PP [65, 63, 62, 42], to FF [57, 27, 61], to TS [41, 54], and hybrid FF+TS [27]. An independent comparison of some of these recent works [57, 63, 62, 41] is carried out in [15], by mean of a single dataset [14] that differs from the ones used by original authors, and the analysis reveals a different scenario from the one pictured by the original publications: (*i*) TS features confirm their interest, but the expected performance drops significantly below <90% for any architecture; (*ii*) there is no clear winner, although 1d-CNN models have consistently better results among the candidate approaches; (*iii*) 1d-CNN has a limited gain against shallow Multi Layer Perceptron (MLP) over the same input (+6%) or against Random Forest (RF) over FF input (+3%). Under this viewpoint, we argue that application identification is still quite an active research area. As we shall see later, AppClassNet constitutes an even more challenging benchmark with respect to the dataset used in [15] or MIRAGE19 [14], and should empower researchers to spend more time on algorithmic aspects rather than investing effort in scattered data collection.

2.2 Datasets viewpoint

Over the years, several datasets have been open-sourced for benchmarking of traffic classification algorithms, summarized in Table 1. We point out that other datasets have been released (see [14] for a comparison). Yet, many of those are collected in the context of Intrusion detection systems (IDS) (as surveyed in [52]). As such, they have a different purpose (distinguishing malicious from benign traffic) and a much less fine-grained labeling (few attack types). Some of these datasets became popular within the traffic classification community, yet one can question their representativeness for traffic classification commercial deployments. Conversely, AppClassNet explicitly aims to offer the larger and more diversified set of traffic typical of commercial settings.

Generally speaking, two methodologies for datasets collection can be employed: (*i*) *Active collections* gather traffic from a specific set of applications, whereas (*ii*) *Passive collections* gather real environment traffic without a specific set of applications defined. While both methods require instrumentation effort, active collections setup are typically more involved. For instance, they typically employ kernel-level modification [35] to annotate executable names polled from kernel information along with packet traces or `strace` kernel logs[14], which allow for more fine-grained ground truth labels. This instrumentation effort coupled with the adoption of *real users* manually-driven procedures [14, 35, 3] limits the collections to relatively small scale experiments. Only rare exceptions reach large scale but rely on *automated but synthetic* collections rather than real users [39]. Conversely, passive collections benefit of large volumes of traffic and cover more diversified users behavior and set of applications, but incur in an extra effort to get ground truth label, e.g., labels can come from open-source [4, 5, 6] or commercial [7, 8, 9] DPI tools, as compared in [24]. `AppClassNet` falls in the latter category and is obtained from a passive collections of 10 TB worth of traffic labeled via a commercial DPI tool (see Sec. 3.4).

Complementary to [15] (that compares several DL architectures on a single dataset), [59] offers a longitudinal analysis of feature importance across datasets (including some of the ones reported in Table 1). The study confirms that (aside TLS-related and inter-flow timing information), the packet size, direction and inter-arrival within a single flow are the features with the highest discriminative power. Thus, both [15] and [59] confirm TS as relevant across DL architectures and datasets, making it the “canonical” input for application identification tasks. As such, `AppClassNet` limits the available input to the TS data. In releasing `AppClassNet`, we point out that we do not believe it should be the *one and only* benchmark for traffic classification. Rather, we hope that it can become *one of the several* benchmarks (e.g., those in Table.1 and beyond) that are normally used in ablation studies, as commonly done in computer vision research.

3 DATASET

As early outlined, the private dataset we used in [70, 69, 32, 31] needs to be deprived of business-sensitive information prior to public release. In this section we start discussing the goals for the transformation process (Sec.3.1), next we outline the available and selected anonymization method (Sec.3.2 and Sec.3.3) and finally we contrast the characteristics of the *original private* against the *publicly released* dataset (Sec.3.4).

3.1 Private and business-sensitive information

We refer to anonymization as a transformation process that alters a dataset for depriving it of both (i) *private information* as well as (ii) *business-sensitive information*

3.1.1 Private information. Privacy-sensitive information is data that can be traced back to a specific user and his/her behavior and location. This includes information about *exact timing*, *IP addresses*, *protocol headers* field values (e.g., HTTP URLs and header options such as `User-Agent`) and *packet-payload* (e.g., the body of HTTP `GET` and `POST` messages). Non-private information instead comprises other data such as protocol headers field values (TCP flags and options, IP message size, IP header length) and *relative timing* (e.g., the time elapsed between two packets, the RTT). The TS-related input features (e.g., raw packet size) previously described fall into this class of non-private information. We understand that from a network-expert viewpoint it would be desirable to employ prefix-preserving IP anonymization techniques as well as to retain exact timing information across flows [59], as they could assist the classification of multiple flows of the same endpoint, or multiple IP addresses in the same range. However, IP addresses are considered as private information according to both the European GDPR [10] and the Chinese PIPL [11] regulations. Hence, to minimize risks of privacy leaks, IP addresses and absolute timing information are not even available in our private version of the dataset.

3.1.2 Business-sensitive information. Business-sensitive data are orthogonal to users privacy but equally important. In fact, despite not being privacy-sensitive, early described TS-related input features (e.g., raw packet size and direction) have business-sensitive value. Indeed, if exact *application signatures* and *application labels* were to be released, then a third-party could train (and sell) an operationally useful model. This is of course delicate from a business viewpoint, for which it would be desirable that models created on public dataset have no operational/commercial (which is the goal of this section). At the same time, to retain interest for the academic community, the public dataset should have comparable performance with respect to models obtained using the private dataset, i.e., working on the public dataset should provide a task of roughly equal hardness as the one of the private dataset (Sec.4).

3.2 Anonymization background

As previously introduced our private dataset does not contain sensitive user-related information. While IP addresses anonymization is a topic that has attracted historical [66] and recent interest [43] in reason of GDPR regulation, those techniques do not fit our needs. Similarly, as our input consists of TS data, at first sight work on time-series anonymization could appear more appropriate (e.g., see [56] and references therein). However, those methods generally aim to protect user privacy (e.g., the input are GPS trace logs), which is generally relevant, but orthogonal to our need.

Luckily, with the growing need to share data and code for reproducibility in ML/DL, a series of work [40, 37, 67, 68] started to tackle *general mechanisms* for altering the data with the goal of privacy protection and resilience to de-identification, i.e. preventing an attacker from identifying matching pairs between raw and encoded samples. These work have generally been applied on image datasets aiming to limit the distortion of the shared dataset [40], possibly hiding private data using noise from very large available image datasets [37] or through learned transformation [67, 68]. Yet among the above works, only [68] attempts at estimating the amount of guesswork that an attacker should spend to succeed a re-identification attack — in our case, a successful re-identification attack allows to uncover the true application labels of the dataset, hence breaking business privacy. More worrisome, [68] also shows previous work to be [40, 37] easily breakable.

Moreover, we observe that these techniques are very recent and, in some cases, not even peer-reviewed yet: thus, the downside of using any of these methods is that security flaws may just not have been uncovered yet. To make a couple of examples, data smashing layers of Split Learning [60] (proposed in 2018, not peer-reviewed but quite popular) were demonstrated to not be secure [49] (in 2021, published at ACM CCS but not yet popular at all). Similarly, NeuraCrypt [67] claims to be secure (in 2021, not yet peer reviewed), while other research claims the opposite [25] (in 2021, not yet peer reviewed). Finally, even the most promising method (Syfer [68] that attempts at directly estimating the level of security) are based on extremely simplifying assumptions (e.g., uniform data distribution [68], while our dataset exhibit significant skew) – the actual level of privacy protection of these general mechanisms remains unclear at this stage.

3.3 Methodology

As per the above discussion, it appears that general and provably secure methodologies are unfortunately not readily available. As security is essentially a risk-vs-cost tradeoff, we ultimately settle on a practical approach where we estimate the cost to reverse engineer the released dataset exceeds the cost of performing a data collection ex novo. In fact, if the data collection cost (i.e., engineering testbeds such as those listed in Table 1 as the community has done in the last decades) is smaller than the cost of reverse engineering, then there would be no business incentive in the latter. Reverse engineering would still remain an intellectual exercise but we forbid it in `AppClassNet` terms of use (Sec.5).

We argue that a good compromise is the joint use of (i) techniques that alter the *amplitude of individual time series elements*, i.e., the size x_i of the i-th packet as in $y = hash(x_i)$ combined with (ii) techniques that *shuffle the whole sequence*, e.g., by swapping i-th and j-th elements. We empirically show that neither of the two techniques is sufficiently strong to avoid leakage of sensitive information if used in isolation, while the combination of the two achieves the desired effect without completely destroying the dataset usability.

3.3.1 *Component-level techniques.*

Consider the homomorphic encryption function:

$$y = hash(x + s) \tag{1}$$

where $hash(\cdot)$ is a cryptographic hash function, x is a time series element and and s an arbitrary salt. Although the cryptographic function cannot be inverted (unless it is compromised) it is not stronger than a mono-alphabetic substitution cipher. Indeed, our input time series are unbalanced due to an *expected skew in the packet size distribution* – by leaving TCP ACK packets in the time series or simply checking full-payload packets one can identify very popular patterns. Thus, an attacker can easily guess the salt s by brute force, by considering only the most common packet sizes. This would not be possible in case packet sizes were equally distributed, but such condition is unrealistic from data collected from real networks. For the same reason, it is unclear if this skew can negatively impact the security of Syfer [68], which assumes balanced input distribution. Clearly, the adoption of multiple salts (e.g., per application, or per position in the time-series) can increase security, with the downside of more severe modification to the original data patterns and distribution.

3.3.2 *Sequence-level techniques.*

Consider now a bijective permutation of the original time series $\overline{x}$ as:

$$\overline{y} = perm(\overline{x}) \tag{2}$$

which alters time series elements order. Moreover, we define such transformation to be "static", i.e., the same reordering is applied to all time series in the dataset. Whereas at first sight, selecting one out of the $n!$ permutation of a n-component long sequence would protect the business sensitive data, the *expected skew in the application popularity* will again defeat the security of the approach. Indeed, permutation of the sequence does not alter the sequence unicity, so that the most frequent sequence is still easily identifiable. In this case, with a relatively low cost, an attacker could gather by his/her own means (e.g., with active experiments) new sample time series $\overline{x}$ of the most popular applications (according to Cisco VNI, Sandvine, etc.) and next attempt at reverse engineering the permutation.[1] As in the previous case, this can be countered, e.g., by using several permutations (e.g., per group of applications, or per individual application, etc.), which would further break the temporal correlation in the sequence, and reduce the usefulness of the dataset for research purposes. As in the previous case, the impact of the imbalance in the application-popularity might negatively affect the strength of security provided by approaches such as [68].

3.3.3 *Combined Component and Sequence level.*

Clearly, combining permutation and hashing makes reverse engineering harder to the point where it is may no longer be cost-effective (as the reverse engineering cost grows above the data collection cost), thus rendering it a mere exercise intellectual exercise (that we prohibit in the terms of use described in Sec.5).

Trading off between scrambling the data to the point of being useless and the risk of sensitive information leak, we settle to:

$$y_i = \frac{hash_{64}(x_j + s_i) - 2^{63}}{2^{64}} \tag{3}$$

where the (i, j) pair represents a generic pair of columns which are swapped once for the whole dataset (e.g., so that the first three 0-payload packets of a TCP application are not expected to be SYN, SYN-ACK, ACK). Additionally, a set of arbitrary parameters s_i is used to salt the 64-bit wide hash function $hash_{64}(\cdot)$ for hashing the i-th samples. Thus packets with the same sizes are hashed to different values depending on their permuted positions in the sequence. Finally, the output is normalized in [-1,1] for convenience.

The use of multiple salts makes reverse engineering cost higher. Indeed, while in principle for SYN and SYN-ACK packets it still possible to rely on the popularity skew to guess their likely positions in the permuted sequence, and next brute-force the corresponding salt s_i, however guessing a single s_i is not informative to learn s_j for the other sequence positions. As such, even finding the top-most popular sequence is more complex, which we estimate to be good enough from our practical purpose.

3.4 Private vs Public dataset characteristics

3.4.1 *At a glance.*

As mentioned before, the original dataset corresponds to about 10 TB of per-flow logs enriched with labels obtained from a commercial and private DPI tool. Such data comprises both TCP and UDP traffic, and for each flow a time series is collected with first 20 packets size and direction (for TCP, ACK packets are preserved). In other words, the dataset has a 20 (features) + 1 (label) tabular schema (i.e., direction is encoded as sign for packet sizes).

Table 2 breakdowns the public-vs-private datasets composition. While the private dataset comprises 3,073 applications, the public

[1] Assuming that the packet size sequence is *exactly* the same, i.e., packet sizes do not differ even a single byte, which is unlikely (but not impossible) for some time series: in case attackers do not get the exact same sequence of packet sizes, a 1-bit input difference flips on average 1/2 of the output bits due to hashing.

Table 2: Commercial-grade dataset description

	Property	Popular	Unpopular	Public	Private
Slice	Nickname	top-200	next-300	pub-500	all-3k
	Public release	✓	✓	✓	✗
Labels	App (#)	200	300	500	3,073
	App (%)	6.5%	9.8%	16.3%	100.0%
	Flow/app (max)	1,000,864	3,050	1,000,864	1,000,864
	Flow/app (avg)	48,299	1,071	19,963	3,272
	Flow/app (min)	3,080	382	382	1
Volume	Flows (#)	9.7 M	0.3 M	10.0 M	10.1 M
	Flows (%)	95.7%	3.2%	98.9%	100.0%
	Bytes (#)	10.0 TB	0.3 TB	10.4 TB	10.4 TB
	Bytes (%)	96.4%	2.8%	99.2%	100.0%

AppClassNet includes only the top-500 (16.3% of all labels) which account for 98.9% of flows and 99.2% of bytes. Out of the public dataset, we further identify two subsets: the *top-200* most popular applications correspond to 95.7% (96.4%) of flows (bytes) with 48,299 samples/app on average (well suited for classic supervised training), complemented by the set of *next-300* (and thus less) popular application, with 1,071 samples/app on average (well fit for zero-day traffic classification, or few shot learning).

Notice the high class imbalance in both top-200 and next-300 dataset slices: the top-1 application comprises over 1 M samples, the 200-th most popular application only comprises 3,080 samples (about 325× time less) and the 500-th only 382 (the least numerous class in AppClassNet). This skew is rather typical in network traffic but is significantly higher than in image datasets. For instance, the nearly 1.5M images in ImageNet [53] are divided into 1,000 classes, each having between 668–3,047 training samples, i.e., a modest 4.5× difference between the most and least popular classes.

3.4.2 *Impact of transformation.*

The described transformation alters some of the properties of the dataset which we briefly illustrate here and that we discuss more broadly in Sec. 5.

Fig. 1 illustrates the effect of hashing considering the first element of the time series (the effect is qualitatively the same on other time series elements). Domain knowledge states that the first packets of a flow are commonly dominated by small packet sizes. For instance, in TCP traffic, those are 0-payload SYN packets due to TCP three-way handshakes, where variability is also due to the presence/absence of IP and TCP options, as well as the presence of UDP traffic in the traffic mixture. In the public dataset (Fig. 1 right chart), the first component has been chosen by permutation (so that it can be any of the 20 elements of the time series) and additionally has been transformed via homomorphic encryption: while the normalized packet size distribution still exhibits clear modes, any domain expertise in this case is meaningless — there is no way to tell whether the modes tie to full-payload segments, 0-payload acknowledgements, or other dynamics. Otherwise stated, *networking domain knowledge is no longer relevant for* AppClassNet.

Additionally, permutations may affect other typical ablation studies, such as the sensitivity of classification performance to time series length as investigated by many works in the literature. However, due to the permutation, the first k components of the time series no longer correspond to the first k packets of a flow, so extrapolate networking-domain conclusions should be avoided (e.g.,

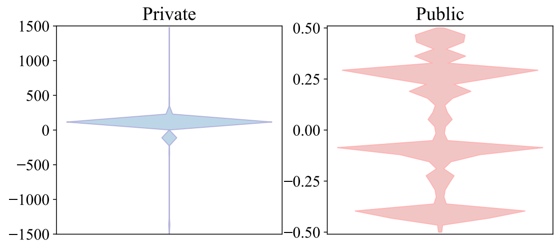

Figure 1: Violin plot of the first signature component in the Private (left: packet size) vs Public datasets (right: permuted, salted and hashed value).

"fewer packets suffice for successful classification"). For instance, mutual information between the first component and the class label is higher in the public dataset, since the first element of the time series is not (confidently) a TCP SYN anymore.

4 BASELINE

In this section we verify that, despite AppClassNet no longer has business-relevance (as it is not a real dataset due to the transformation), it is still meaningful from an academic research viewpoint (as a relevant dataset of benchmarking).

We do so by studying the performance (Sec. 4.2) across different state of the art ML/DL models (Sec. 4.1). However, the goal of this editorial note is not to perform a fully fledged study (e.g., as in [15]), but rather to contrast the baseline performance obtained from both public and private version of the dataset.

4.1 Baseline models

We adopt both traditional ML and DL state of the art methods as baselines. First, we decided to evaluate the prediction performance of two traditional tree-based models with different complexities: a single CART [23] decision tree and an ensemble of trees via a Random Forest [22]. Then, we integrate in our benchmark some commonly adopted state of the art DL methods: a classic 1d-CNN [33] and ResNet [36].

Specifically, we used Scikit-Learn [50] for traditional ML baselines (DecisionTreeClassifier, RandomForestClassifier) with their default parameters and the random state set to 0. For DL baselines, we implemented the 1-d CNN (3 convolutional ReLU layers) and ResNet (2 residual blocks) in PyTorch and trained models for 1,000 epochs, with batch size of 1,024 and a 0.01 learning rate. To ease the bootstrapping cost of using AppClassNet, we also provide the code for both to import the data and to replicate results [12].

4.2 Baseline performance

We contrast baselines performance of ML and DL models in Table 3. For each model we report the raw classification accuracy for top-200 and the model accuracy rank (the lower, the better) on the test set. Without loss of generality, we release AppClassNet structured with separate train/validation/test splits to simplify reproducibility (although the use of alternative splits for k-fold cross-validation is

Table 3: Performance of several ML/DL baselines on the top-200 slice of the public `AppClassNet` vs private datasets

	Decision Tree†			Random Forest†			1d-CNN‡			ResNet‡		
Dataset	*Rank*	*Acc*	*Depth*	*Rank*	*Acc.*	*Depth*	*Rank*	*Acc.*	*Params*	*Rank*	*Acc.*	*Params*
Private	2	93.5%	71	1	94.7%	65	4	84.7%	275k	3	90.8%	318k
Permuted	2	93.5%	71	1	94.7%	65	4	81.9%	275k	3	88.8%	318k
Public (`AppClassNet`)	2	86.9%	78	1	88.3%	88	4	72.1%	275k	3	80.3%	318k

†ML: default scikit-learn configuration, `random_state=0`

‡DL: `epochs=1000`, `batch_size=1024`, `learning_rate=0.01`

authorized and rather recommended). The dataset has been normalized as a preprocessing step using Scikit-Learn [50] (MinMaxScaler). Additionally, the depth (average depth) of the decision tree (random forest) and the total number of trainable parameters for DL approaches are reported as proxy of models complexity.

Permutation effect. We compare the performance across 3 variations: the private dataset, an intermediate step where only permutation is applied, and the public `AppClassNet` where both permutation and component hashing are applied. The intermediate step is added as a litmus test: we expect permutation to have little impact on tree-based ML, where the order of features is not important; we conversely expect permutation to have a larger impact on the DL models, as permutation breaks the correlation in the sequence that convolutional filters are explicitly designed to extract. This is indeed what can be observed in Table 3: performance of tree-based approaches remain identical, whereas DL-model accuracy looses about 2-3%.

Hashing effect. Next, we observe the effect of hashing. Intuitively, hash functions spread component far apart (1/2 of bits change in the output for 1-bit change in the input) resulting in additional branching in trees that inflates their depth by 43% (average random forest depth by 37%) and reduces accuracy by 6%. DL-models size is instead fixed but result into a slightly higher accuracy loss of 9% compared to tree-based models. Clearly, one can adjust both architecture parametrizations and obtain different results, as we show in [69].

Consistent rank. Considering performance from a higher level viewpoint, we observe that ranking of the considered models is consistent across private/permuted and public datasets. This is a very important indication in our opinion that `AppClassNet` can be profitably used for research purposes. Albeit the quantitative results provide a conservative estimate of performance in real settings (by about 6%-10%), the qualitative comparison retains full informative value.

Challenging dataset. Finally, the dataset is interesting as it represents an instance where classic tree-based ensembles perform better than DL on tabular data [55]. Clearly, the table has been obtained for specific architectures and hyper-parametrizations, and results are meant as a qualitative reference baseline in spirit with the Ockham razor principle. Thus, we do not rule out DL models completely, e.g., as DL gradient backpropagation has a significant competitive advantage for zero-day application detection[69]. Moreover, Random Forest models contain 30M leaves with a slim improvement (<1%) compared to a simpler decision tree (286k leaves). Hence, one cannot conclude that Random Forest models obtained with default parameters are a suitable solution to the problem (as it is, quite frankly, an overkill). Rather, the academic community should be aware that the quest for 1% accuracy improvement comes with a model complexity cost, which should not be neglected.

Summary. The anonymization applied to our private dataset represents a good compromise as ML/DL models suffer only a modest accuracy loss with respect to the private dataset and qualitative results show consistent ranking across models – `AppClassNet` is a “realistic” and conservative benchmark aiming to empower researchers to address challenging production-level problems having larger scale [69] than datasets commonly adopted in literature.

5 DISCUSSION

While we hope that `AppClassNet` becomes one of the standard benchmarks for traffic classification problems, we also need to clarify its allowed and prohibited usages (Sec.5.1), as well as its limitations as result of the performed anonymization (Sec.5.2).

5.1 Allowed and prohibited usage

Openly sharing dataset has associated cost (e.g., the anonymization process). Additionally, it exposes the data provider to additional risks (e.g., reverse engineering) and consequent potential harm (e.g., leak of business sensitive information) [16]. As such, we clearly state the allowed and prohibited usages of `AppClassNet`. We understand that, while doing so, we are not fully sheltered from malicious third parties abuse. At the same time, we appeal researchers seeking to use `AppClassNet` to follow ACM Code of Ethics and Professional Conduct [34], which clearly puts *avoid harm* among its first and foremost articles (Art 1.2), and to adhere to the following guidelines.

5.1.1 *Allowed usage.*

Explicitly allowed usage of the open-source dataset is limited to *Academic usage*, and in particular limited to algorithms for inference and training related to the use case of traffic classification, as described in this paper. While it is out of the scope of this section to provide a fully exhaustive list of legally binding allowed usages, we make concrete examples for the sake of clarity. Allowed usage for *inference-related ML/DL* problems include for instance: application identification (supervised classification on fine-grained labels[69]), zero-day application detection (e.g., open set recognition[69]), inference speedup (by hardware acceleration[32], approximate caching[31], quantization). Allowed usage for *training-related ML/DL algorithms* include for instance: distributed training by sharing models instead of data (federated learning[38, 70]), training from few samples (few-shot learning[64]), pre-processing techniques to simplify training (transfer or self-supervised learning[26, 58]), techniques to add/remove classes to/from existing models (incremental/decremental learning[21, 19]), techniques to automatically design new classification engines (neural architecture search[30]). While it is outside of the scope of this paper to also provide a research agenda for open problems related to traffic classification, we note that combinations of above research tasks is also possible and novel. AppClassNet is a good playground for autoML design of accurate models that are also computationally simple from an inference perspective, which has only seldomly been taken into account. We finally suggest to researchers working on unrelated traffic classification use cases early listed, or whenever in doubt about the legitimacy of using AppClassNet in their work, to contact us [34].

5.1.2 Prohibited usage. We explicitly *prohibit usages of the dataset for commercial use.* We additionally *prohibit any reverse engineering* attempts, in line with deontological considerations concerning network measurements [16]. In reason of the anonymization techniques employed, the current dataset has direct little commercial value. As such, we mostly discuss reverse engineering practices. Indeed, while academic work performing reverse engineering would not per se constitute a violation of commercial usage, it could significantly lower the barrier for third parties to use the reverse engineered dataset for commercial use. As such, *de-anonymization techniques* that would try to expose users, classes, or application labels, signatures and any other non directly available information of the private dataset, from any technique applied on data available in the public dataset, *are explicitly prohibited.* We understand that, once data is available publicly, this risk exists. However, we hope to make it clear that, shall AppClassNet be victim of a reverse engineering attempt (fruitful or not), this would significantly threaten the ability for industrial players to release datasets such as this one in the future. We thus appeal once more to [34] and suggest to contact for a usage that is not explicitly authorized.

5.2 Limits and alternatives

As earlier introduced, AppClassNet does not aim to be *the one and only* traffic classification benchmark. As such we list its limits and alternatives for the interested readers.

5.2.1 Limits. The limited amount of information in AppClassNet constrains the possible classification operations. For instance, owing to the lack of flow-level identifiers, it is not possible to perform aggregated endpoint classification, or correlate multiple classification across multiple flows from the same endpoint [59]. Since the dataset considers only the first 20 packet sizes and directions for a flow, it is unsuitable for multi-modal classification [13] or continuous classification across multiple windows of the same (sub)flow [46].

Moreover, several studies have analyzed the classification performance as a function of the sequence length, which is altered by permutation in AppClassNet, making such analysis worthless. Similarly, while approximate-key caching [31] would work given that the natural skew of the traffic is preserved, the approximation function prefix(·) that works best on the private dataset would not be suitable on the public dataset due to permutation.

The dataset does not contain flow arrival rate, and as such it cannot be used as a realistic workload for traffic classification engines. However, it can still be used as a controlled workload [32] for stress testing the target classification system. As such, AppClassNet is more geared for contribution on ML/DL techniques, than for novel system-level aspects (e.g., feature engineering).

5.2.2 Alternatives. We observe that while collecting large volumes of real labeled network data is a daunting effort for a *single academic* partner, a coordinated pooling effort across *multiple research groups* can be an effective strategy to achieve this goal – the list of datasets reported in Table 1 and the additional datasets reviewed in [59, 14] indeed constitutes a good example and starting point.

Clearly, due to the anonymization process, the current dataset cannot be directly *merged* with other datasets (as learning on this dataset is non directly informative for other real applications). At the same time, used alongside alternatives such as those expressed above, AppClassNet can hopefully help the scientific community to focus and reinforce the soundness of scientific results. We believe that, given its characteristics (i.e., scale of samples and classes, conservative performance), AppClassNet provides a challenging and instrumental research playground as it helps ensuring that results are not biased by small datasets – this avoids excessively focusing on "easy problems" where many solutions can achieve the 90% accuracy "holy grail".

6 CONCLUSION

This paper describes AppClassNet, a dataset obtained from a private commercial-grade dataset comprising thousands of application labels. AppClassNet represents a substantial portion of the original private dataset, opportunely altered to deprive it of user-private and business-sensitive information.

We openly release the dataset at [12], along with code for baseline techniques, and we hope that it can become instrumental for the benchmarking of data-driven algorithms applied to networking-related classification problems.

REFERENCES

[1] [n. d.] https://www.image-net.org/download.php. ().
[2] [n. d.] https://commoncrawl.org/. ().
[3] [n. d.] https://recon.meddle.mobi/cross-market.html. ().
[4] [n. d.] https://wand.net.nz/projects/details/libprotoident. ().
[5] [n. d.] https://sourceforge.net/projects/l7-filter/. ().
[6] [n. d.] https://github.com/ntop/nDPI. ().
[7] [n. d.] https://www.cisco.com/c/en/us/products/ios-nx-os-software/network-based-application-recognition-nbar/index.html. ().
[8] [n. d.] https://www.ipoque.com/products/dpi-engine-rs-pace-2-for-application-awareness. ().
[9] [n. d.] https://support.huawei.com/enterprise/de/doc/EDOC1000012889?section=j00c. ().
[10] [n. d.] https://en.wikipedia.org/wiki/General_Data_Protection_Regulation. ().
[11] [n. d.] https://en.wikipedia.org/wiki/Personal_Information_Protection_Law_of_the_People's_Republic_of_China. ().
[12] [n. d.] https://figshare.com/s/bdf83bc55e37bc5b2a5f. ().
[13] Giuseppe Aceto et al. 2019. Mimetic: mobile encrypted traffic classification using multimodal deep learning. *Computer networks*, 165, 106944.
[14] Giuseppe Aceto et al. 2019. Mirage: mobile-app traffic capture and ground-truth creation. In *International Conference on Computing, Communications and Security (ICCCS)*. IEEE.

[15] Giuseppe Aceto et al. 2018. Mobile encrypted traffic classification using deep learning. In *Proc. IEEE TMA*.

[16] Mark Allman and Vern Paxson. 2007. Issues and etiquette concerning use of shared measurement data. In *ACM SIGCOMM Internet Measurement Conference (IMC)*, 135–140.

[17] Laurent Bernaille et al. 2006. Traffic classification on the fly. *ACM SIGCOMM Computer Communication Review*, 36, 2, 23–26.

[18] Dario Bonfiglio et al. 2007. Revealing skype traffic: when randomness plays with you. In *Proc. ACM SIGCOMM*.

[19] Lucas Bourtoule et al. 2019. Machine unlearning. *arXiv preprint arXiv:1912.03817*.

[20] Raouf Boutaba et al. 2018. A comprehensive survey on machine learning for networking: evolution, applications and research opportunities. *Journal of Internet Services and Applications*, 9, 1, 16.

[21] Giampaolo Bovenzi et al. 2021. A first look at class incremental learning in deep learning mobile traffic. In *IFIP Traffic Monitoring and Analysis (TMA)*.

[22] L. Breiman. 2001. Random forests. *Machine Learning*, 45.

[23] L. Breiman et al. 1984. *Classification and Regression Trees*. Taylor & Francis. ISBN: 9780412048418.

[24] Tomasz Bujlow et al. 2015. Independent comparison of popular dpi tools for traffic classification. *Computer Networks*, 76, 75–89. ISSN: 1389-1286. DOI: https://doi.org/10.1016/j.comnet.2014.11.001.

[25] Nicholas Carlini et al. 2021. Neuracrypt is not private. *CoRR*, abs/2108.07256. https://arxiv.org/abs/2108.07256.

[26] Ting Chen et al. 2020. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*. PMLR, 1597–1607.

[27] Zhitang Chen et al. 2017. Seq2img: a sequence-to-image based approach towards ip traffic classification using convolutional neural networks. In *Proc. IEEE BigData*, 1271–1276.

[28] Manuel Crotti et al. 2007. Traffic classification through simple statistical fingerprinting. *ACM SIGCOMM Computer Communication Review*, 37, 1, 5–16.

[29] P.M. Santiago del Rio et al. 2012. Wire-speed statistical classification of network traffic on commodity hardware. In *Proc. ACM IMC*.

[30] Thomas Elsken et al. 2019. Neural architecture search: a survey. *The Journal of Machine Learning Research*, 20, 1, 1997–2017.

[31] Alessandro Finamore et al. 2022. Accelerating deep learning classification with error-controlled approximate-key caching. *IEEE INFOCOM*.

[32] Massimo Gallo et al. 2021. Fenxi: deep-learning traffic analytics at the edge. *ACM/IEEE Symposium on Edge Computing (SEC)*.

[33] I. Goodfellow et al. 2016. *Deep Learning*. MIT Press.

[34] DW Gotterbarn et al. 2018. Acm code of ethics and professional conduct.

[35] Francesco Gringoli et al. 2009. GT: picking up the truth from the ground for internet traffic. *ACM SIGCOMM Computer Communication Review*, 39, 5, 12–18.

[36] K. He et al. 2016. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*.

[37] Yangsibo Huang et al. 2020. InstaHide: instance-hiding schemes for private distributed learning. In *Proceedings of the 37th International Conference on Machine Learning*. Volume 119, 4507–4518.

[38] Jakub Konečný et al. 2016. Federated learning: strategies for improving communication efficiency. In *NeurIPS Workshop on Private Multi-Party Machine Learning*.

[39] Martina Lindorfer et al. 2014. Andrubis–1,000,000 apps later: a view on current android malware behaviors. In *IEEE International Workshop on Building Analysis Datasets and Gathering Experience Returns for Security (BADGERS)*, 3–17.

[40] Bo Liu et al. 2021. Dp-image: differential privacy for image data in feature space. (2021). arXiv: 2103.07073 [cs.CR].

[41] Manuel Lopez-Martin et al. 2017. Network traffic classifier with convolutional and recurrent neural networks for internet of things. *IEEE Access*, 5, 18042–18050.

[42] Mohammad Lotfollahi et al. 2020. Deep packet: a novel approach for encrypted traffic classification using deep learning. *Soft Computing*, 24, 3.

[43] Meisam Mohammady et al. 2018. Preserving both privacy and utility in network trace anonymization. In *ACMConference on Computer and Communications Security (CCS)*, 459–474. DOI: 10.1145/3243734.3243809. https://doi.org/10.1145/3243734.3243809.

[44] Andrew W Moore and Konstantina Papagiannaki. 2005. Toward the accurate identification of network applications. In *Proc. PAM*.

[45] Andrew W Moore and Denis Zuev. 2005. Internet traffic classification using bayesian analysis techniques. In *Proc. ACM SIGMETRICS*.

[46] Thuy TT Nguyen and Grenville Armitage. 2006. Training on multiple sub-flows to optimise the use of machine learning classifiers in real-world ip networks. In *IEEE LCN*, 369–376.

[47] Thuy TT Nguyen and Grenville J Armitage. 2008. A survey of techniques for internet traffic classification using machine learning. *IEEE Communications Surveys and Tutorials*, 10, 1-4, 56–76.

[48] F. Pacheco et al. 2018. Towards the deployment of machine learning solutions in network traffic classification: a systematic survey. *IEEE Communications Surveys and Tutorials*, 1–1.

[49] Dario Pasquini et al. 2021. Unleashing the tiger: inference attacks on split learning. In *ACM Computer and Communications Security (CCS)*.

[50] F. Pedregosa et al. 2011. Scikit-learn: machine learning in Python. *Journal of Machine Learning Research*, 12, 2825–2830.

[51] Jingjing Ren et al. 2018. A longitudinal study of PII leaks across android app versions. In *Network and Distributed System Security Symposium (NDSS)*. Volume 10.

[52] Markus Ring et al. 2019. A survey of network-based intrusion detection data sets. *Springer Computers Security*, 86, 147–167. ISSN: 0167-4048.

[53] Olga Russakovsky et al. 2015. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115, 3, 211–252. DOI: 10.1007/s11263-015-0816-y.

[54] Tal Shapira and Yuval Shavitt. 2019. Flowpic: encrypted internet traffic classification is as easy as image recognition. In *IEEE INFOCOM Workshops*.

[55] Ravid Shwartz-Ziv and Amitai Armon. 2021. Tabular data: deep learning is not all you need. *CoRR*, abs/2106.03253. https://arxiv.org/abs/2106.03253.

[56] Nazanin Takbiri et al. 2018. Matching anonymized and obfuscated time series to users' profiles. *IEEE Transactions on Information Theory*, 65, 2, 724–741.

[57] Vincent F Taylor et al. 2016. Appscanner: automatic fingerprinting of smartphone apps from encrypted network traffic. In *Proc. IEEE EuroS&P*.

[58] Md Shamim Towhid and Nashid Shahriar. 2022. Encrypted network traffic classification using self-supervised learning. In *IEEE NetSoft*.

[59] Thijs van Ede et al. 2020. Flowprint: semi-supervised mobile-app fingerprinting on encrypted network traffic. In *Network and Distributed System Security Symposium (NDSS)*. Volume 27.

[60] Praneeth Vepakomma et al. 2018. Split learning for health: distributed deep learning without sharing raw patient data. *CoRR*, abs/1812.00564. arXiv: 1812.00564. http://arxiv.org/abs/1812.00564.

[61] Ly Vu et al. 2017. A deep learning based method for handling imbalanced problem in network traffic classification. In *ACM International Symposium on Information and Communication Technology*.

[62] W. Wang et al. 2017. End-to-end encrypted traffic classification with one-dimensional convolution neural networks. In *Proc. IEEE ISI*.

[63] Wei Wang et al. 2017. Malware traffic classification using convolutional neural network for representation learning. In *Proc. IEEE ICOIN*.

[64] Yaqing Wang et al. 2020. Generalizing from a few examples: a survey on few-shot learning. *ACM Computing Surveys*, 53, 3, 1–34.

[65] Zhanyi Wang. 2015. The applications of deep learning on traffic identification. *BlackHat USA*.

[66] Jun Xu et al. 2002. Prefix-preserving IP address anonymization: measurement-based security evaluation and a new cryptography-based scheme. In *IEEE International Conference on Network Protocols (ICNP)*. IEEE, 280–289.

[67] Adam Yala et al. 2021. Neuracrypt: hiding private health data via random neural networks for public training. (2021). arXiv: 2106.02484 [cs.CR].

[68] Adam Yala et al. 2022. Syfer: neural obfuscation for private data release. (2022). arXiv: 2201.12406 [cs.LG].

[69] Lixuan Yang et al. 2021. Deep learning and zero-day traffic classification: lessons learned from a commercial-grade dataset. *IEEE Transactions on Network and Service Management*, 18, 4. DOI: 10.1109/TNSM.2021.3122940.

[70] Lixuan Yang et al. 2020. Heterogeneous data-aware federated learning. In *IJCAI Workshop on Federated Learning*.