Deep Learning and Traffic Classification: A critical review with novel insights from real deployment

Lixuan Yang, **Alessandro Finamore**, **Dario Rossi**
Huawei Technologies, France
{name.surname}@huawei.com

Abstract

The increased success of Convolutional Neural Networks (CNNs) has recently re-sparked interest towards traffic classification (TC). New literature shows the potential of reaching perfect classification accuracy, yet at the core of those works still reside the same limitations surfaced in the 1st wave of TC techniques, started in early 2000s and culminated with the application of Machine Learning (ML) for "early classification". To better highlight and discuss such issues, in this paper we report on novel insights based on a commercial-grade traffic classification engine. We aim to perform a critical review of the state of the art, introducing the research community to problems arising at a commercial scale (30× the largest number of classes in academic literature), discussing the pitfalls and how to avoid them. Leveraging a dataset comprising millions of flows and thousands of labels, we perform a fair comparison of classic Machine Learning (ML) and novel Deep Learning (DL) architectures based on the same inputs. We additionally put emphasis that comparing techniques on the mere raw performance make no sense unless they are fairly compared in terms of their computational requirements, an often forgotten key aspect affecting deployment.

1 Introduction

Classification of Internet traffic is a well investigated subject, whose research interest started in the early 2000s, to supplant light packet inspection (i.e., port-based) and deep packet inspection (i.e., payload based) technologies with statistical tools able to characterize broad traffic classes, and the specific applications within each class. Seminal works such as [Roughan *et al.*2004], ignited a *first wave* of classification approaches [Moore and Papagiannaki2005, Bernaille *et al.*2006, Crotti *et al.*2007, Bonfiglio *et al.*2007, Kim *et al.*2008, Nguyen and Armitage2008] essentially focused on extracting features for classifying a relatively small set of applications, relying on classic Machine Learning (ML) approaches based on careful –but human intensive– feature engineering processes. This first wave culminated with very simple yet effective techniques, referred to as "early traffic classification" [Bernaille *et al.*2006, Crotti *et al.*2007] that readily used time series information (e.g., the size and direction of the first few packets in a flow) to take classification decisions.

The tremendous successes of Convolutional Neural Networks (CNN) in the image recognition field [Krizhevsky *et al.*2012] ignited a *second wave* of traffic classification approaches leveraging Deep Learning (DL) techniques [Wang2015, Taylor *et al.*2016, Wang *et al.*2017b, Wang *et al.*2017a, Lopez-Martin *et al.*2017, Chen *et al.*2017, Vu *et al.*2017, Aceto *et al.*2018, Shapira and Shavitt2019, Lotfollahi *et al.*2020]. DL is becoming particularly appealing in reason of domain-specific hardware (known as "tensor processing units") that started appearing in the last few years, and featuring hardware acceleration making CNN a viable and appealing option for real-time traffic classification. In reason of the tremendous push toward encryption in the Post-Snowden era, this second wave of research is particularly relevant since industrial players are now actively looking at deploying statistical classification approaches – that so far mostly remained an academic exercise, as recently pointed out in [Pacheco *et al.*2018].

Yet, there is still a gap between the industrial interests and the attention of academic research. First and foremost, whereas commercial DPI tools are able to handle *hundreds to thousands* of application classes, statistical techniques developed in the academic world consider only a *few tens* of classes, which is significantly simpler than commercial needs and is a major blocking points to deployment of classification techniques [Pacheco *et al.*2018]. The first contribution of this work is to share important insights gained from real deployments, showing not only that classification becomes more difficult as the number of classes grows, but also illustrating phenomena that only arise in a large number of classes regime. To do so, we leverage a commercial-grade dataset comprising tens of millions of flows, and thousands of application labels (about 30× bigger than the state of the art [Taylor *et al.*2016]).

Second, the research patterns that happened in the first wave of academic research (i.e., proposing a specific set of engineered features to train a ML technique) are repeating in the second wave too (i.e., proposing a specific set of raw input data to train a DL architecture), with an *excessive focus* *on raw performance*: the net result is a race to beat the former champion, whereas competing approaches are rarely [Aceto *et al.*2018] compared in an independent fashion. Here, rather than proposing new techniques, our contribution is to contrast state of the art ML and DL techniques on exactly the same input (packets size time series for "early traffic" classification). We purposely showcase that ML accuracy performance can be made to exceed DL performance, which is mostly tied to the complexity of the model, and introduce the notion of weights-per-class (i.e., the total number of DL/ML model parameters divided by the number of classes): this allows to better quantify the accuracy-vs-complexity tradeoff that academic research ignores, and that will instead play a paramount role in the deployability of these research prototypes.

In this paper, we perform a critical review of the state of the art (Sec. 2), and leverage a commercial grade dataset (Sec. 3) and state of the art ML/DL techniques (Sec. 4) to unveil phenomena visible only at a commercial scale and illustrate the too often ignored complexity-accuracy tradeoff (Sec. 5). We finally discuss implications of our findings (Sec. 6).

2 Related work

As the first wave of traffic classification (coarse grained, few classes) and application identification (fine grained, many classes) research has been well covered by numerous surveys [Nguyen and Armitage2008, Boutaba *et al.*2018], in this section we summarize the main lessons learned from this first phase, and focus our attention to the closest and most recent work, i.e., those using neural networks for traffic classification, summarized in Tab. 1

First ML wave. In the first decade of 2000, a first wave of traffic-classification (coarse grained) or application-identification (fine grained) studies leveraged "classic" ML techniques, initially applying ML to engineered flow features (FF) as in [Moore and Zuev2005], or packet payload (PP) as in [Moore and Papagiannaki2005, Bonfiglio *et al.*2007], culminating in simple yet effective classification techniques based on timeseries (TS) data such as the size (S), direction ($\pm$), and seldom interarrival time (ΔT) of the first few packets of a flow [Bernaille *et al.*2006, Crotti *et al.*2007]. The interest of lightweight TS approaches is that (i) they operate "early" at the beginning of a flow, as opposite to "post mortem" as for techniques based on FF that are computed after a flow ends, and that (ii) they sustain line-rate operation with minimal additional computational complexity. Indeed, whereas payload-based techniques are inherently limited in the amount of memory they need to access/move even when processing is done on GPUs [Vasiliadis *et al.*2011], early TS techniques [Bernaille *et al.*2006, Crotti *et al.*2007] have been amenable to line rate classification in excess of 5 Mfps [del Rio *et al.*2012] using only general purpose CPU.

Second DL wave. The second wave of research re-considered all possible inputs exploited in the first phase —from PP [Wang2015, Wang *et al.*2017b, Wang *et al.*2017a, Lotfollahi *et al.*2020], to FF [Taylor *et al.*2016, Chen *et al.*2017, Vu *et al.*2017], to TS [Lopez-Martin *et al.*2017, Shapira and Shavitt2019], and hybrid FF+TS [Chen *et al.*2017]— but has not been exempt from obvious flaws and pitfalls related to input selection.

For instance, we argue that owing to encryption, most work exploiting packet payload PP is essentially learning the content of the TLS Server Name Indication (SNI) extension [Eastlake and others2011], i.e., binding a flow to the hostname advertised in clear in the SNI: ultimately, this means that 1d-CNN approaches[1] leaveraging PP are *a complex mean to do relatively trivial HTTPS protocol dissection and pattern matching* – which to the best of our knowledge nobody pointed out clearly so far. Moreover, while DL is an elegant and automated way to statistically learn SNI "dictionaries", the real question is whether CNN inference can be brought to an operational point with lower computational footprint than traditional techniques – which is never questioned either.

Similarly, we remark that TS work employs *port numbers* [Lopez-Martin *et al.*2017, Chen *et al.*2017]: e.g., as advertised by authors [Lopez-Martin *et al.*2017], the accuracy decreases to 82% when port are not used in the input, and the fact that the average accuracy is 94% when TS consists of a single packet[2] confirms that the CNN (and LSTM) architectures described in those works are *ultimately learning port-based classification that they are supposed to supplant*.

More generally, research work of the second phase studies and proposes different DL architectures (and hyperparametrization), sharing the same crucial weakness already noted in the first wave, i.e., the difficulty of cross comparing in a fair manner these different architectures: indeed, as it clearly emerges from Tab. 1, every work uses different datasets, with different sample size (from 20k to 750k samples), and with different target classes (from 2 to 100), achieving performance in excess of >99% (under specific conditions), making an apple-to-apple comparison a daunting task (e.g., even work relying on the same datasets cannot be directly compared, for which *open challenges* such as the one pushed by NetAML conference are welcome).

Fortunately, commendable work such as [Aceto *et al.*2018] started appearing, aiming to an independent evaluation of previously published work. The comparison carried out in [Aceto *et al.*2018] (of [Taylor *et al.*2016, Wang *et al.*2017b, Wang *et al.*2017a, Lopez-Martin *et al.*2017], and by mean of datasets different from the one used by original authors) reveals a different scenario from the one pictured by the original publications: (i), the expected performance, in practice, drops significantly below <90% for any architecture; (ii) there is no clear winner, although 1d-CNN have consistently better results among the candidate approaches; (iii) 1d-CNN has a limited gain over shallow Multi Layer Perceptron (MLP) over

[1]Similarly happens for 2d-CNN, that additionally artificially construct "spatial" dependencies in the payload that are make sense for images of the physical world, but are less motivated for textual/binary protocols

[2]As per Fig. 10 in [Lopez-Martin *et al.*2017]: additionally, the stricking performance are due to the fact that HTTP, SSL and DNS (that are amenable to port 80, 443 and 53 respectively), account for over 80% flows.

Table 1: Related work

1st: ML	Input	Architecture	Samples	Classes	Performance	Notes
[Roughan *et al.*2004]	2 FF (avg. D, avg S)	k-NN vs LDA	5.1M	4 (7)	95% (91%)	100% adding an extra feature related to ΔT
[Moore and Papagiannaki2005]	PP	9 DPI heuristics	573M	10	99%	macro-classes and manual ground truth creation
[Bernaille *et al.*2006]	TS_5 (±S)	K-means	n.a.	10	≈90%	
[Crotti *et al.*2007]	TS_3(±S, ΔT)	2d gaussian filter	30k	4	≈90%	multi-dimensional statistical fingerprint

2nd: DL	Input	Architecture	Samples	Classes	Performance	Notes
[Wang2015]	PP (1000B)	1D CNN	300k	58	90% in top-25	Introduces DL to TC
[Taylor *et al.*2016]	FF (40)	RF (non-DL)	131k	110	99%	Introduces APP classification
[Wang *et al.*2017b]	PP (784B=28×28)	2D CNN	750k	20	99%	payload as image
[Wang *et al.*2017a]	PP (784B)	1D CNN	750k	12	≈ 90%	payload as blob
[Lopez-Martin *et al.*2017]	TS_{20} $(\pm S, \Delta T, p)$	LSTM + 2D-CNN	266k	15	96% (82% w/o p)	several models: (CNN+RNN-2a)
[Chen *et al.*2017]	TS_{10} $(\pm S, \Delta T, p)$ + FF (28)	CNN	22k	5+(5 real)	99% (88%)	
[Vu *et al.*2017]	FF (22)	GAN (vs DT and RF)	682k	2	all 99%	SSH vs non-SSH
[Aceto *et al.*2018]	PP, FF, TS	MLP, SAE LSTM, CNN	138k	49	80-86%	DL [Lopez-Martin *et al.*2017, Wang2015, Wang *et al.*2017b, Wang *et al.*2017a, Lotfollahi *et al.*2020] vs RF [Taylor *et al.*2016]
[Shapira and Shavitt2019]	TS → 2D histo (1500^2)	LeNet-5 like	21k	10	99%	
[Lotfollahi *et al.*2020]	Payload	1D CNN and SAE	unclear	17	98%	dataset is public, but flows are not specified

***Input**: flow features (FF), packet payload (PP), flow duration (D); time series (TS) of packet size (S), direction (±), interarrival (ΔT) and ports (p).*
***Architecture**: Random Forest (RF); Multi-layer perceptron (MLP); Stacked Autoencoders (SAE), Convolutional neural networks (CNN); Long-short term memory (LSTM); Generative adversarial networks (GAN)*

the same input (+6%) or Random Forest (RF) over FF input (+3%). We underline that such insights are possible only when broadening the evaluation scope beyond the typical race to reach 100% classification accuracy.

Despite its merits, [Aceto *et al.*2018] still partially falls into an apple-vs-orange comparison. For instance, the classic RF model inherited from [Taylor *et al.*2016] is based on engineered flow features (FF), whereas the CNN models are either based on packet payload (PP) [Wang *et al.*2017b, Wang *et al.*2017a] or packet time series (TS) [Lopez-Martin *et al.*2017]. As such, is is extremely difficult to attribute results improvement to the learning technique (i.e., ML vs DL) or the model input (e.g., FF vs TS). To counter this problem, sharing the same spirit of [Aceto *et al.*2018] we perform an independent evaluation of two state of the art ML/DL techniques, applied to exactly the same input (TS), on a commercial grade dataset.

3 Dataset

Our datasets comes from two separate product lines, offering device and services for the *Enterprise campus* and *Customer OLT/ONT* market segments respectively. In a nutshell the datasets comprise of packet-level captures from 4 real customer deployments, spanning over 1 week each, where individual flows are annotated with labels provided by a commercial-grade engine as we describe next.

Collection environments. The datasets are collected from customers in China. We underline that traffic encryption in China is not as pervasive as in the Western world yet. Moreover, as commonplace in the Enterprise market, branches employs HTTPS proxy, so that DPI can work unperturbed. This explains the availability of a very large number of label: overall, the dataset comprises *3231 application labels, which more than* 30× *the largest number of classes considered in the academic literature* [Taylor *et al.*2016]. The dataset is a private Huawei asset containing sensitive information and cannot unfortunately be shared – as often remarked in the literature, the lack of common dataset is one of the major limit of this field (see discussion in Sec.6).

Table 2: Commercial-grade dataset description

Classes	%	Flows	%	Byte	%
10	0.3%	3.4M	32.6%	5.6 TB	53.0%
20	0.6%	4.6M	43.6%	7.2 TB	67.1%
50	1.5%	7.2M	68.3%	8.8 TB	82.7%
100	3.1%	8.7M	82.9%	9.8 TB	91.9%
200	6.2%	9.9M	94.0%	10.3 TB	97.1%
250	7.7%	10.2M	95.5%	10.4 TB	98.2%
1000	30%	10.5M	99.5%	10.6 TB	99.9%
3231	100%	10.5M	100%	10.6 TB	100%

Traffic imbalance. Clearly, applications have a skewed popularity which induces a rather typical class imbalance, as illustrated without loss of generality in Tab.2 for one customer. For the typical number of classes K in the literature (i.e., 10 to 50), considering the most popular applications entails that, while these apps represent a tiny portion of the application catalog (i.e., from 0.3% to 1.5%) they nevertheless correspond to a sizeable portion of the traffic flows (e.g., 32.6% to 68.3%) and bytewise volume (e.g., 53.0% to 82.7%) respectively. However, about 1/3 and 1/5 of the flows and bytes are not covered even when $K = 50$, making these effort still faraway from what would be viable for commercial products.

Additionally, as the classification problems become mechanically more difficult as the number of classes increase, despite academic models work well (generally, in excess of 99%) for 10-50 cherry-picked classes, it is not obvious to project the results of the same architecture when confronted to hundreds (if not thousands) of classes – which is the core of part of our investigation (see Sec.5).

4 Models

Our aim in this section is to introduce state-of-the art models of the two ML and DL "waves", applied on the exact same input, namely packet time-series (TS) data, appealing due to both good accuracy and simplicity. We seek for models that are representative of performance lower-bounds, to provide conservative assessment on commercial grade perspective, and to avoid falling in the arms race to just provide yet-another-solution that is better than the previous ones – which, as we shall see, allows to convey some key observations.

Output. To allow grasping performance differences intrinsically related to the output scale of the problem, we construct models apt at a fine grain application identification of K classes. Recalling Tab.1, DL classifier proposed in the literature typically consider $K{<}50$ classes. Even work considering a larger set of applications, either reports the accuracy of the top classes (top-25 [Wang2015]) or the dataset used is practically limited to fewer classes (e.g., the top-15 classes represent over 99% of the traffic in [Lopez-Martin *et al.*2017]). There are only a few works that actually deal with K=50 classes [Aceto *et al.*2018,Taylor *et al.*2016]. Conversely, recalling Tab.2, in our case it is necessary to consider $K' = 200$ classes to cover 95% (97%) of the flows (bytes). We thus consider two scenarios with $K{\in}\{50, 200\}$ classes, that are representative of state-of-the art academic evaluation [Aceto *et al.*2018] vs business needs respectively, and large enough to allow us to make key observations. We leave as future work an in-depth evaluation of how to identify all 3231 classes available in the dataset.

Input. Models leverage timeseries input (TS) using 10 (or 100) packets size and direction only for UDP (or TCP). The timeseries is thus a $\pm S \in \mathbb{Z}$ that can be rescaled into $[0, 1]{\subset}\mathbb{R}$ by normalizing packet size over maximum MTU. This input has been consistently been found to yield to excellent performance across both ML [Bernaille *et al.*2006, Crotti *et al.*2007] and DL [Lopez-Martin *et al.*2017, Chen *et al.*2017] waves, and as further confirmed by independent validation studies [Aceto *et al.*2018]. It is also appealing given the simplicity to collect such metrics.

This simple input could be complemented by further TS information, such as ΔT interarrival or other useful header bits (e.g., TCP flags). However, we prefer to avoid leveraging additional questionable input (e.g., port based information from the packet header) as per the previous discussion. Additionally, we remark that the first wave of literature found ΔT to be a valid input, yet prone to induce classification errors [Bernaille *et al.*2006, Crotti *et al.*2007]: from network domain knowledge, whenever ΔT is elapsed between two consecutive packets of the same flow direction, then it may correlate with the application behavior; however, when ΔT is measured between a packet and its response from the other endpoint, then the delay represents the Round Trip Time (RTT) that correlates more with the distance between the endpoints, than with application behavior.

As such, the one-dimensional $\pm S$ timeseries alone stands out as a reasonable choice. On the one hand, this simple input yields to a conservative (which is desirable from a scientific standpoint) performance lower bound, that as such does not reveal sensitive product-related performance (which is also desirable from a business standpoint). On the other hand, it allows an unbiased apple-to-apple comparison of DL and ML on the same input — which is our main goal.

DL model (1d-CNN). As representative DL model, we use a 1d-CNN that is (i) well fit to the TS series data, and that was found to have superior performance (ii) by authors comparing 2d-CNN and 1d-CNN on PP input [Wang *et al.*2017b, Wang *et al.*2017a], and (iii) further re-confirmed by the independent evaluation in [Aceto *et al.*2018]. We stress that we rule out recursive networks (and LSTM in particular), in reason of the additional computational complexity, that is not well supported by hardware accelerators. We stick to most common design choices, with the input layer feeding a stack of convolutional filters having depth 3 and 16(UDP) or 32(TCP) filters of size $1{\times}3$ with ReLU activation and max-pooling layers, followed by one fully connected layer of size 64(UDP)-128(TCP) for a total depth of 4 layers before the final SoftMax classification (of size K). While detailed hyperparameters differ across each work, this architecture is rather typical and is exploited (with minor differences) by [Wang *et al.*2017b, Wang *et al.*2017a, Lopez-Martin *et al.*2017, Chen *et al.*2017, Shapira and Shavitt2019]. In reason of space, and to simplify tractation, we will identify the complexity of each CNN model tested with space complexity, i.e., the overall number of model weights W – tied to both training and inference times. [3]

ML model (XGBoost). As representative ML model, we use extreme gradient boosting (XGBoost) [Friedman2001], a tree-based ensemble technique combining numerous individually weak learners, widely acknowledged as the state of the art in numerous applications domains [Chen and Guestrin2016]. We consider a fixed number of trees $T = 100$ and limitedly explore capping the individual tree depth $d \in \{2, 4, 7\}$ as a mean to control the total number of nodes W – which we take as proxy of model complexity.

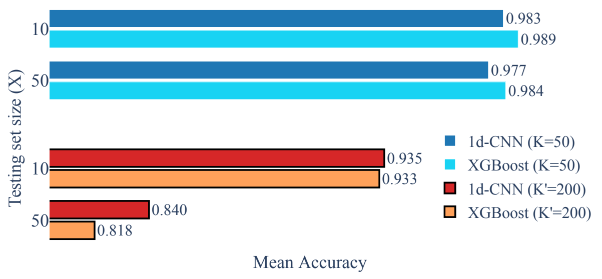

Figure 1: *Class cardinality bias*: When training the same DL/ML model for a large number of classes $K' \gg K$ accuracy drops also for the top $X \leq K$ classes, which is unnoticed in academic literature.

5 Evaluation

We illustrate potential biases related to accuracy evaluation (Sec.5.1), and dissect models under the generally ignored complexity angle (Sec.5.2).

5.1 Accuracy

Class cardinality bias. As early introduced, academic models are trained for no more than $K{=}50$ classes, and performance

[3]We use two models, one for TCP and one for UDP, to better address the different number of classes they contribute (30 for TCP and 170 for UDP)

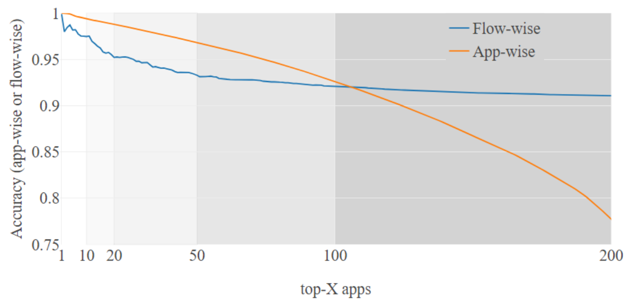

Figure 2: *Mean accuracy bias.* Average per-flow (affected by class imbalance) or per-application accuracy (each application counted equally). Notice that the per-application accuracy overestimate performance in the small class regime $K < 100$, and vice-versa happens for $K > 100$.

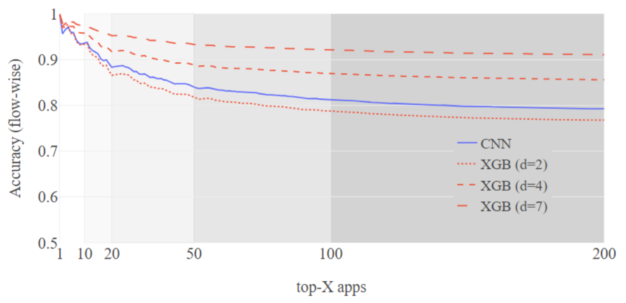

Figure 3: *Model bias.* By tuning model hyperparameters, it is easy to obtain operational points that show superiority of one class of approaches (biased to ML in this example).

are generally reported for $X \leq K$ classes, with $X \in \{10, 50\}$. We contrast the academic evaluation settings with the typical business case where the same model (i.e., with the same architecture and hyperparametrization) is instead trained to recognize $K' = 200$ classes and performance are then reported for $X \leq 200$ classes, considering for the time being still $X \in \{10, 50\}$. Results of this experiment are reported in Fig.1, where we purposely select a DL and a ML model that when K=50, (i) have for the top-X=50 classes a similar flow-level accuracy to what reported in the literature ($\approx$99%) and (ii) have approximately the same accuracy (97.7% vs 98.4%).

To put it simple, when ML/DL models are trained to recognize at most K=50 classes, Fig.1 shows no noticeable performance degradation in the overall accuracy when the performance results are reported for the top-10 or the whole top-50. Conversely, when models are trained to recognize a 4$\times$ larger number of K=200 classes we then observe that accuracy degradation is already noticeable for top-10, and degradation is sizeable for the top-50 classes. *In other words, focusing on a small number of classes makes the problem trivial to solve (even without DL) and hides problems arising when the space becomes sufficiently dense.*

Mean accuracy bias. This is further exacerbated by considering Fig.2, showing for models trained on $K' = 200$ apps, the mean accuracy of the top-X apps, where ranking is done either in terms of app popularity (most popular –by means of number of flows– app first), or in terms of app performance (best classified app first). The picture clearly shows that, (i) despite classification accuracy of individual applications worsen as X increases towards K', (ii) the overall number of correctly classified flows remains satisfactory in reason of the application popularity skew – *otherwise stated, focusing on the average accuracy, and particularly for a small number of classes $K < K'$, hides performance problems that arise only when $K \gg 100$ may render commercial deployment non viable.*

DL bias. Finally, we argue that in the second wave of classification, the benefits of DL may be exaggerated — which is a well known side effect of the publish or perish arms race, and has already been documented in computer science [Lipton and Steinhardt2019] as well as other fields [Hutson2020]. Given lack of space, we exemplify this in Fig.3, where we purposely show that ML accuracy can be made (significantly) better than DL one by tuning a single ML hyperparameter (the maximum tree depth in the example) – *though we stress that we could easily have adopted the opposite viewpoint, as it may inadvertently happen in the rush to publish of new DL results.*

5.2 Complexity

The hidden confounding factor in the above results is the model complexity. To make the analysis simple, we abstract each specific DL/ML model instance with the overall number of parameters W: clearly, while space complexity do not directly translate into computational complexity (as this depends on the DL/ML architecture, the specific operations executed at inference time) or energy expenditures (as this depends on, e.g., the availability of hardware accelerators), focusing on space complexity W allows to abstract from specific implementation (i.e., software, hardware acceleration, system design choices). Particularly, as models are disparate in their capabilities, to make a fair comparison across them it is necessary to not only report the absolute model size W, but especially the model size normalized over the number of output classes W/K.

Fig.4 illustrates the accuracy vs complexity tradeoff for both our and literature models, where accuracy (complexity) comparison should be interpreted qualitatively (quantitatively). In particular, the scatter plot shows that literature employs up to millions [Chen *et al.*2017] of CNN weights per class, with the most parsimonious approach [Shapira and Shavitt2019] still employing over 60k weights per class. In contrast, the CNN models we considered employ between 2.2k (110k/50) and 285 (57k/200) weights per class.

Two observations emerge from the picture. On the one hand, when the number of classes is large ($K \approx 200$), it becomes necessary to increase models space to maintain accuracy performance. On the other hand when the number of classes is small ($K \approx 50$), it is unreasonable to use an humongous number of weights to discriminate them, particularly since it is possible to design parsimonious models with

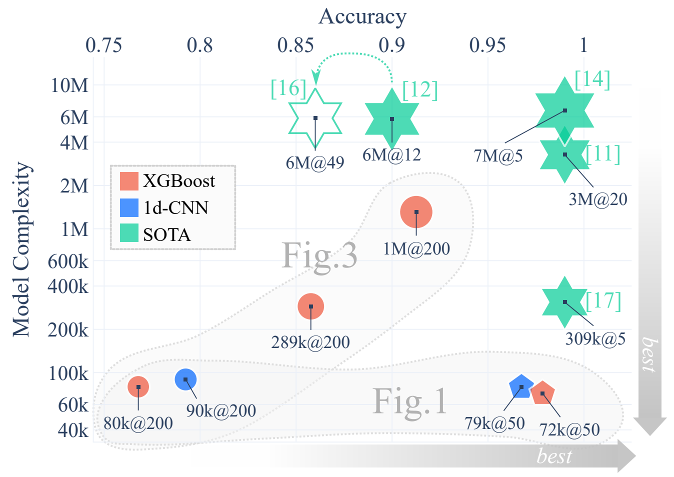

Figure 4: *Complexity*. Scatter plot of model space complexity W vs accuracy for our ML (blue circles) and DL (red circles) models, where empty circles are used for models of Fig.3. State of the art references (green stars) are included for *qualitative* comparison. Shapes are annotated with models size and class cardinality, with the shape radius is proportional to the W/K ratio.

just hundreds of weights-per-class achieving same-or-better performance (low right corner of the scatter plot). *As such, by neglecting model complexity, the risk is to propose solutions that are classic equivalent of shooting a mosquito with a cannon.*

6 Discussion

In this paper, we critically review and compare models of the first and second wave of classification. We show that whereas classifying a few classes is a relatively simple task, performance of the same architectures degrades if stressed with a large number of classes. We also show that generally models are not evaluated from a complexity viewpoint, which possibly yields to models that are unnecessarily complex for relatively simple tasks – endangering the practical relevance of the research.

Open dataset: pooling efforts. In reason of our results, constructing an open corpus with rich *class diversity* and large *class cardinally* K should be a priority goal to allow for meaningful and fair cross-comparison of research proposals. While this is within reach for large industrial players, however legal and business aspects prevent them to share openly their datasets – this is old news, but unfortunately apply also to the datasets used in this paper.

While this is a daunting effort for a *single academic* partner, pooling effort across *multiple research groups* N in a coordinated manner can be an effective strategy to achieve this goal. For instance, asking each partner to gather K/N classes, and coordinating so that $\mathcal{K}_i \cap \mathcal{K}_j =$, i.e., no overlap for any two groups. Lots of groups are doing active measurement collections for specific application types (video, games, etc.) and different goals than traffic classification (congestion control, QoE, etc.), so that the true burden lies in the coordination. Yet, this is commonplace in other communities (e.g., Imagenet has 15million images in 20k classes) offering a positive examples that the traffic classification community should adopt. The NetAML challenge is a good starting point, as it could federately and systematically grow the data for such a challenge over time.

Deployability: the elephant in the room. Given that traffic classification is a mature research topic, we believe that focusing on raw classification performance of a supervised model, albeit of novel DL models, is no longer justified. To finally make academic models to step out of academic venues, it is imperative to tackle other pressing problems impacting models deployability in the real world [Pacheco *et al.*2018].

Deployability issues also includes aspects related to *model training*, such as for instance continuous (to tackle drift and zero day applications [Carela-Español *et al.*2016]), or distributed (e.g., privacy respectful federated learning [Yang *et al.*2020]) learning. Deployability issues especially includes aspects related to *model inference*, such as computational costs (e.g., to ensure the model execution is within the CPU/energy budget [Gallo *et al.*2020]) as well as auditing/explainability of classification decisions for the non experts (which has practical relevance since unlike decision trees, DL models have no direct explanation [Beliard *et al.*2020, Meng *et al.*2020])

Acknowledgement

We wish to thank Cedric Beliard for his initial contributions (and mosquito repellent, way more useful than cannons).

References

[Aceto *et al.*, 2018] Giuseppe Aceto, Domenico Ciuonzo, Antonio Montieri, and Antonio Pescapé. Mobile encrypted traffic classification using deep learning. In *Proc. IEEE TMA*, 2018.

[Beliard *et al.*, 2020] Cedric Beliard, Alessandro Finamore, and Dario Rossi. Opening the deep pandora box: Explainable traffic classification. In *Proc. IEEE INFOCOM, Demo session*, 2020.

[Bernaille *et al.*, 2006] Laurent Bernaille, Renata Teixeira, Ismael Akodkenou, Augustin Soule, and Kave Salamatian. Traffic classification on the fly. *ACM SIGCOMM Computer Communication Review*, 36(2):23–26, 2006.

[Bonfiglio *et al.*, 2007] Dario Bonfiglio, Marco Mellia, Michela Meo, Dario Rossi, and Paolo Tofanelli. Revealing skype traffic: when randomness plays with you. In *Proc. ACM SIGCOMM*, 2007.

[Boutaba *et al.*, 2018] Raouf Boutaba, Mohammad A Salahuddin, Noura Limam, Sara Ayoubi, Nashid Shahriar, Felipe Estrada-Solano, and Oscar M Caicedo. A comprehensive survey on machine learning for networking: evolution, applications and research opportunities. *Journal of Internet Services and Applications*, 9(1):16, 2018.

[Carela-Español *et al.*, 2016] Valentín Carela-Español, Pere Barlet-Ros, Albert Bifet, and Kensuke Fukuda. A streaming flow-based technique for traffic classification applied to 12+ 1 years of internet traffic. *Telecommunication Systems*, 63(2):191–204, 2016.

[Chen and Guestrin, 2016] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proc. ACM KDD*, 2016.

[Chen *et al.*, 2017] Zhitang Chen, Ke He, Jian Li, and Yanhui Geng. Seq2img: A sequence-to-image based approach towards ip traffic classification using convolutional neural networks. In *Proc. IEEE BigData*, 2017.

[Crotti *et al.*, 2007] Manuel Crotti, Maurizio Dusi, Francesco Gringoli, and Luca Salgarelli. Traffic classification through simple statistical fingerprinting. *ACM SIGCOMM Computer Communication Review*, 37(1):5–16, 2007.

[del Rio *et al.*, 2012] P.M. Santiago del Rio, D. Rossi, F. Gringoli, L. Nava, L. Salgarelli, and J. Aracil. Wire-speed statistical classification of network traffic on commodity hardware. In *Proc. ACM IMC*, 2012.

[Eastlake and others, 2011] Donald Eastlake et al. Transport layer security (tls) extensions: Extension definitions. Technical report, IETF RFC6066, 2011.

[Friedman, 2001] Jerome H Friedman. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232, 2001.

[Gallo *et al.*, 2020] Massimo Gallo, Alessandro Finamore, Gwendal Simon, and Dario Rossi. Real-time deep learning based traffic analytics. In *Proc. ACM SIGCOMM, Demo session*, Aug. 2020.

[Hutson, 2020] Matthew Hutson. Eye-catching advances in some ai fields are not real. *Science Magazine*, May 2020.

[Kim *et al.*, 2008] Hyunchul Kim, Kimberly C Claffy, Marina Fomenkov, Dhiman Barman, Michalis Faloutsos, and KiYoung Lee. Internet traffic classification demystified: myths, caveats, and the best practices. In *Proc. ACM CoNEXT*, 2008.

[Krizhevsky *et al.*, 2012] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[Lipton and Steinhardt, 2019] Zachary C Lipton and Jacob Steinhardt. Troubling trends in machine learning scholarship. *Communications of the ACM*, 2019.

[Lopez-Martin *et al.*, 2017] Manuel Lopez-Martin, Belen Carro, Antonio Sanchez-Esguevillas, and Jaime Lloret. Network traffic classifier with convolutional and recurrent neural networks for internet of things. *IEEE Access*, 5:18042–18050, 2017.

[Lotfollahi *et al.*, 2020] Mohammad Lotfollahi, Mahdi Jafari Siavoshani, Ramin Shirali Hossein Zade, and Mohammdsadegh Saberian. Deep packet: A novel approach for encrypted traffic classification using deep learning. *Soft Computing*, 24(3), 2020.

[Meng *et al.*, 2020] Zili Meng, Minhu Wang, Jiasong Bai, Mingwei Xu, Hongzi Mao, and Hongxin Hu. Interpreting deep learning-based networking systems. In *Proc. ACM SIGCOMM*, 2020.

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

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

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

[Pacheco *et al.*, 2018] F. Pacheco, E. Exposito, M. Gineste, C. Baudoin, and J. Aguilar. Towards the deployment of machine learning solutions in network traffic classification: A systematic survey. *IEEE Communications Surveys and Tutorials*, 2018.

[Roughan *et al.*, 2004] Matthew Roughan, Subhabrata Sen, Oliver Spatscheck, and Nick Duffield. Class-of-service mapping for qos: a statistical signature-based approach to ip traffic classification. In *ACM IMC*, 2004.

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

[Taylor *et al.*, 2016] Vincent F Taylor, Riccardo Spolaor, Mauro Conti, and Ivan Martinovic. Appscanner: Automatic fingerprinting of smartphone apps from encrypted network traffic. In *Proc. IEEE EuroS&P*, 2016.

[Vasiliadis *et al.*, 2011] Giorgos Vasiliadis, Michalis Polychronakis, and Sotiris Ioannidis. Midea: a multi-parallel intrusion detection architecture. In *Proc. ACM CCS*, 2011.

[Vu *et al.*, 2017] Ly Vu, Cong Thanh Bui, and Quang Uy Nguyen. A deep learning based method for handling imbalanced problem in network traffic classification. In *ACM International Symposium on Information and Communication Technology*, 2017.

[Wang *et al.*, 2017a] Wei Wang, Ming Zhu, Jinlin Wang, Xuewen Zeng, and Zhongzhen Yang. End-to-end encrypted traffic classification with one-dimensional convolution neural networks. In *Proc. IEEE ISI*, pages 43–48, 2017.

[Wang *et al.*, 2017b] Wei Wang, Ming Zhu, Xuewen Zeng, Xiaozhou Ye, and Yiqiang Sheng. Malware traffic classification using convolutional neural network for representation learning. In *Proc. IEEE ICOIN*, 2017.

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

[Yang *et al.*, 2020] Lixuan Yang, Cedric Beliard, and Dario Rossi. In *Proc. IJCAI Federated Learning workshop*, 2020.