When Simple Model Just Works: Is Network Traffic Classification in Crisis?

Kamil Jerabek[2], Jan Luxemburk[1,3], Richard Plny[1,3], Josef Koumar[1,3], Jaroslav Pesek[1,3], and Karel Hynek[1,3]

[1]Czech Technical University in Prague, Faculty of Information Technology, Czech Republic
[2]Brno University of Technology, Faculty of Information Technology, Czech Republic
[3]CESNET, a.l.e., Prague, Czech Republic
Corresponding author email: ijerabek@fit.vutbr.cz

***Abstract*—Machine learning has been applied to network traffic classification (TC) for over two decades. While early efforts used shallow models, the latter 2010s saw a shift toward complex neural networks, often reporting near-perfect accuracy. However, it was recently revealed that a simple k-NN baseline using packet sequences metadata (sizes, times, and directions) can be on par or even outperform more complex methods. In this paper, we investigate this phenomenon further and evaluate this baseline across 12 datasets and 15 TC tasks, and investigate why it performs so well. Our analysis shows that most datasets contain over 50% redundant samples (identical packet sequences), which frequently appear in both training and test sets due to common splitting practices. This redundancy can lead to overestimated model performance and reduce the theoretical maximum accuracy when identical flows have conflicting labels. Given its distinct characteristics, we further argue that standard machine learning practices adapted from domains like NLP or computer vision may be ill-suited for TC. Finally, we propose new directions for task formulation and evaluation to address these challenges and help realign the field.**

1. Introduction

Traffic classification (TC) based on machine learning (ML) has become essential in network security research, particularly with the increasing prevalence of encrypted communication. As traditional methods like deep packet inspection lose effectiveness, ML provides a viable and often the only alternative for identifying communication patterns. Advances in deep representation learning in fields like computer vision and natural language processing have inspired similar approaches for traffic classification. Recent work by Luxemburk et al. [1] proposed a universal flow representation and unexpectedly found that a simple baseline based only on packet sequence and a k-NN classifier performed on par with or exceeded state-of-the-art (SOTA) methods across multiple datasets. We refer to this baseline as *input-space*, reflecting its use of raw packet sequence metadata without any additional feature processing. Nevertheless, the mentioned study only briefly addressed the findings and emphasized the need for further research.

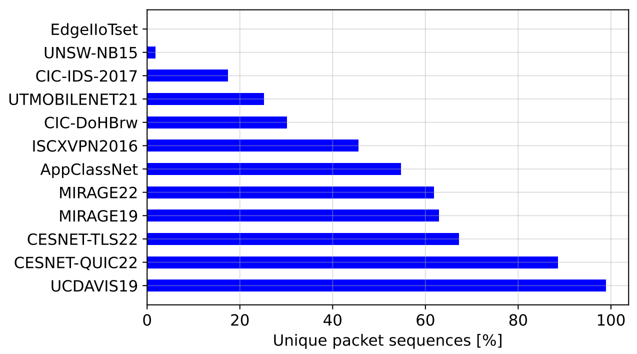

Figure 1. Fraction of unique sequences of packet lengths and directions present in the examined datasets.

In this work, we replicate and analyze the results of the simple input-space baseline across 12 datasets and 15 TC tasks, and discover that its strong performance can be largely explained by the presence of extensive redundancy within the datasets. As shown in Figure 1, half of the evaluated datasets would shrink to less than 50% of their original size if duplicates were removed and only unique samples were counted. We also show how such redundancy affects evaluation practices by estimating the theoretical maximum accuracy achievable per dataset.

Our findings raise concerns about the relevance of current evaluation methodologies in the TC field. Unlike other domains such as computer vision or NLP, network traffic originates mostly from API requests, which may vary in content but typically share request-response similarities on the packet level. Traditional evaluation approaches, however, fail to account for this characteristic. To address this limitation, we propose recommendations for adapting evaluation protocols to produce more trustworthy and representative results.

The paper is organized as follows. Section 2 describes advancements in the field prior to this work. Section 3 defines the baseline method and presents results on 12 datasets. Section 4 provides an in-depth analysis of the network traffic data. The outcomes and recommendations are covered in the discussion Section 5. The last Section 6 concludes the work.

2. Related Works

The network traffic classification has experienced significant evolution over time. In the early 2000s, researchers began applying machine learning techniques at a larger scale to network traffic. Several surveys [2], [3] describe this pioneering period, during which the community explored various approaches and candidate features, typically categorized into flow-level and packet-payload features. The flow-level features used in [4], [5] contain information directly from the flow, such as IP addresses, number of transmitted bytes and packets, and other statistical information gathered from the packet sequence. The payload features used in [6] typically contain information from the L7 protocol headers. The rise of traffic encryption has reduced the relative importance of payload features, but they still remain valuable in some scenarios.

Despite no general consensus on the type of universal traffic classification features, the research community in [7], [8], [9] has gradually adopted time-series information derived from the initial packets of each flow as one of the primary types of features. These features, often referred to as the Sequence of Packet Lengths and Times (SPLT), comprise the lengths of individual packets, their direction (inbound vs. outbound), and the inter-packet arrival times.

In the latter half of the 2010s, TC has moved to the deep learning era. Numerous studies [10], [11], [12], [13], [14], [15] leveraged recent advances in data science to enhance classification performance, applying deep learning techniques to a variety of network traffic scenarios. These methods also use the combination of all the features previously mentioned. Several datasets were introduced at that time, some of which have since become de facto standards in the research community. Important examples include ISCXVPN2016 [16], UNSW-NB15 [17], and CIC-IDS-2017 [18], each cited thousands of times.

However, beyond their age, these datasets are known to contain significant issues. For instance, Lanvin et al. [19] identified critical flaws in the CIC-IDS-2017 dataset, including inconsistent timestamps, mislabeled samples, and duplicate flows. Although they released a corrected version, the community continues to predominantly rely on the original dataset. Other studies, aware of these problems, have performed custom filtering to mitigate the issues. Aceto et al. [20] analyzed the ISCXVPN2016 dataset and identified several inconsistencies, providing a detailed description of the filtering steps required to make it usable. Nevertheless, the majority of subsequent studies continue to rely on the original, unfiltered version of the dataset.

Despite acknowledged concerns regarding dataset quality—as previously described and also discussed in several surveys [21], [22]—the research community has continued to evaluate deep learning models primarily using the data in the original form. Classical shallow machine learning methods have been largely overlooked, amid a prevailing consensus that more sophisticated models are essential to achieve further performance gains. More recently, Luxemburk et al. [1] compared their novel embedding-based approach with a simple input-space baseline. Surprisingly, the baseline often matched or even outperformed more complex deep learning methods. However, the study did not thoroughly investigate the underlying reasons for the baseline effectiveness, nor was the evaluation extended across multiple datasets.

In this study, we adopt the baseline method proposed by Luxemburk et al. [1]. We evaluate its performance across 12 widely used datasets from the traffic classification domain. Furthermore, we conduct an in-depth analysis to investigate the underlying factors contributing to the unexpectedly high performance of this simple baseline. Based on our findings, we offer recommendations for refining experimental protocols within the TC research community.

3. Exploring k-NN Performance

In this section, we summarize the baseline approach [1] and evaluate its performance over 15 classification tasks. To ensure a fair comparison, all experiments are conducted under consistent conditions using standardized evaluation metrics. Moreover, we apply Optuna-based hyperparameter tuning to optimize selected parameters and establish the baseline performance limits. Our goal is to validate the strength of this baseline as an encrypted traffic classification method and highlight some significant limitations of popular TC datasets.

3.1. Definition of Input-Space Baseline

Input-space baseline is a k-NN classifier using L1 distance and the following feature set: sizes, inter-packet times in milliseconds (IPT), and directions (encoded as ±1) of the first *N* (up to 30) payload packets. The use of the L1 metric measures distances through simple sums of differences at individual packet positions and features. Of course, this means that the baseline cannot ever capture differences across packet positions (such as missed and retransmitted packets or shifted patterns in the sequences). As the three packet features have different units and scales, we define the following four parameters to allow their reweighting:

1) N – the number of packets;
2) DIR_{scale} – the scaling factor for directions;
3) IPT_{scale} – the scaling factor for IPTs;
4) $IPT_{maxclip}$ – IPTs are clipped to this maximum value before scaling.

The specific configuration used in [1] was $N = 10$, $DIR_{scale} = 1$, $IPT_{scale} = \frac{1}{10}$, and $IPT_{maxclip} = 1000$. In this work, for each dataset, we use its validation set and the Optuna framework [32] to find a baseline configuration with the best performance. We acknowledge that the parameter optimization process, which is described in more detail in Section 3.4, makes the baseline more complex. However, in comparison to black-box models like CNNs, it still remains simple and straightforward to interpret.

TABLE 1. PUBLIC DATASETS AND EVALUATION SPLITS USED IN OUR EXPERIMENTS.

Dataset	Task	#Flows	#Classes	Splits (Train / Val / Test)	#Reps	Split strategy
CESNET-TLS22 [23]	TLS	141M	191	1M / 0.1M / 1M	5	time-based (weeks 1 vs 2)
CESNET-QUIC22 [24]	QUIC	153M	102	1M / 0.1M /1 M	5	time-based (weeks 1 vs. 2,3,4)
ISCXVPN2016 [16]	VPN	10.5k	2	60% / 20% / 20%	5	random
MIRAGE19 [25]	Android apps	122k	20	80% / 10% / 10%	5	prepared random splits
MIRAGE22 [26]	Video-meeting apps	59k	9	80% / 10% / 10%	5	prepared random splits
UTMOBILENET21 [27]	Mobile apps	9.5k	17	80% / 10% / 10%	5	prepared random splits
UCDAVIS19 [28]	QUIC	7k	5	7k / — / 83 or 150	1	fixed
AppClassNet [29]	TLS	10M	500	1M / 0.1M / 1 M	5	random
EdgeIIoTset [30]	IIoT	73M	15	1M / 0.1M / 1M	5	random
UNSW-NB15 [17]	Network attacks	2.5M	10	0.5M / 0.1M / 0.5M	5	random
CIC-IDS-2017 [18]	Network attacks	2M	8	0.5M / 0.1M / 0.5M	5	random
CIC-DoHBrw [31]	DoH	1.5M	2	0.5M / 0.1M / 0.5M	5	random

3.2. Evaluation Protocol

We selected 12 traffic classification or IDS datasets to evaluate the input-space baseline. Some datasets provide more tasks in the form of multiple separate test sets (UCDAVIS19) or multiple labels (ISCXVPN2016). In total, we have 15 classification tasks; for each, we use the same evaluation protocol explained below.

First, we put in extra effort to obtain the same data as that used in SOTA and follow the same filtering and preprocessing steps. We used all datasets available in the tcbench framework[1], which offers curated and pre-split versions of popular datasets. For ISCXVPN2016, we contacted the authors of the current best SOTA [33], who sent us their preprocessed version. For the remaining datasets, we did our best to choose the most relevant SOTA and follow its preprocessing steps. The exact train/test/split procedure of each dataset is described in Section 3.3.

With the data available, preprocessed, and split into train/validation/test sets, we run five repetitions and average the results. In each iteration, the training set is used to train a k-NN classifier, which we implement as a Faiss [34] index that provides an efficient nearest neighbor search. Then, for each test sample, we obtain a prediction as the label of the closest (top1) training sample. Apart from this *top1* approach, we also evaluate distance-based majority voting that is detailed in Section 3.5.

3.3. Datasets

Twelve traffic classification datasets focused on various tasks were chosen for comparison with SOTA. The dataset selection was limited to only those that were published in a format where packet sequences of network flows were obtainable; for example, PCAP format or flow records. When the dataset was provided in PCAP format, we used ipfixprobe[2] flow exporter to extract the packet sequences. In this section, we describe the selected datasets and provide a structured overview in Table 1.

CESNET-TLS22 and **CESNET-QUIC22** were both collected at backbone lines spanning two and four weeks, respectively. The datasets' authors, Luxemburk et al. [23], [35], used the whole week for training and the week or weeks after for testing; we refer to this as a *time-based* train-test split. In our case, we follow the described approach, using the first week of each dataset for training and the remaining weeks (in the case of TLS, one week; in the case of QUIC, three weeks) for testing. Section 5.1.2 discusses the challenges and benefits of time-aware evaluation and examines the observed performance degradation in more detail.

ISCXVPN2016 was published in 2016 and represents a captured real traffic generated by lab members [16]. For each traffic type, the authors captured regular sessions and sessions over VPN. Historically, researchers have used various approaches to preprocess this dataset. We appreciated the approach of Nascita et al. [33], who removed broadcast flows and cleaned the dataset of other sources of noise. They provided us with their preprocessed version consisting of around 10.5k samples.

Mirage series datasets **MIRAGE19** [25] and **MIRAGE22** [26] focus on services and capture real users; the first is based on interactions with 20 Android applications, the latter is focused on video meeting applications such as Zoom, Webex, or Teams (9 applications in total). We used the prepared train/validation/test splits available in tcbench, with filtered flows shorter than 10 packets. **UTMOBILENET21** captures Android application communication, providing packet sequences in CSV format [27]. The authors of tcbench cleaned the data, assembled flows, filtered flows with less than 10 packets, and divided the remaining 9.5k samples into five splits, which we use in this work.

The first QUIC traffic dataset **UCDAVIS19** was published in 2019 [28] and contains five Google services: Google Drive, Google Docs, Google Search, Google Music, and YouTube. It includes a pretraining partition and two test sets—human (83 samples) and script (150 samples). For this dataset, we decided not to use the prepared splits in tcbench. Rather, we used the entire pretraining as the training set and evaluated the two test sets (which we consider as separate classification tasks) without averaging over multiple repetitions.

AppClassNet from 2022 contains TLS flows drawn from 500 mobile and web applications. Authors of the

1. https://tcbenchstack.github.io/tcbench/
2. https://github.com/CESNET/ipfixprobe

dataset, Wang et al. [29] provide an official split, but for consistency we averaged over five repetitions.

Industrial IoT (IIoT) dataset **EdgeIIoTset** [30] mixes legitimate IIoT telemetry (MQTT) with diverse network attacks in the same capture within a lab setup. Normal operation and attack campaigns were run intermittently between 21 November 2021 and 10 January 2022. IDS dataset **UNSW-NB15** was captured in 2015 at UNSW Canberra's Cyber Range Lab [17]. Benign traffic was replayed through a live test network while nine attack types were injected with a generator. The authors exported the training and test sets; however, for consistency, we generated our own splits. The very popular IDS dataset **CIC-IDS-2017** [18] captures five days of generated traffic with seven types of distinct attacks. Background benign traffic is generated by 25 synthetic employees via tool B-Profile, and the malicious traffic is generated by 15 individual tools. The authors used a star topology with the victim in the center. We filtered out empty flows and broadcasts. The DoH dataset **CIC-DoHBrw-2020** contains encrypted flows labeled as non-DoH, benign-DoH, or malicious-DoH, created by MontazeriShatoori et al. [31]. It captures automated browsing of the Alexa Top-10k in Google Chrome and Mozilla Firefox. Malicious DoH traffic is generated in parallel by three distinct tools. The authors place a dedicated DoH proxy in the center of a star topology.

3.4. Optuna Tuning

For all datasets—except for UCDAVIS19, which does not have a validation set—we search for the best input-space baseline configuration. We use Optuna, which implements state-of-the-art hyperparameter search algorithms, to optimize validation classification performance averaged over five train/validation/test splits. The best-found parameters are then used for evaluation on the test sets. The results of the optimized baseline are presented in Table 3. In the *Optim.* column, we can observe that the gains from optimization are modest—ranging from 0.5 to 1 percentage point for traffic classification datasets—and negligible for IDS datasets. AppClassNet is an outlier, with optimization yielding an improvement of nearly 5 percentage points over the "default" input-baseline parameters reused from [1]. The best configurations for each dataset are presented in Table 2.

For each dataset, a distinct configuration of hyperparameters (described in Section 3.1) was selected based on validation performance. During the optimization process, the packet size sequence remained unchanged, while the other two sequence components—packet directions and inter-packet times—were scaled using multiplicative factors DIR_{scale} and IPT_{scale}, as listed in Table 2. This scaling adjusted the relative influence of these features during the computation of L1 distances. Moreover, inter-packet times were clipped to a maximum value of $IPT_{maxclip}$ to reduce the impact of outliers in this feature.

The optimal relative importance of packet directions and IPTs varied across datasets, with no clear pattern. The optimal number of packets used for classification, N, was ten or fewer for most datasets, suggesting that the initial part of each flow carries sufficient discriminative information for identifying a wide range of traffic classes. One notable observation is that CESNET-TLS22 and CESNET-QUIC22, although being similar datasets, differ in the presence of application-level control packets (QUIC ACKs), which leads to higher optimal N for CESNET-QUIC22.

3.5. Distance-Based Majority Voting

In addition to the *top1* approach, we experimented with various forms of majority voting among the nearest samples. While standard majority voting among the k nearest neighbors did not yield substantial gains, distance-based voting proved more promising. Instead of voting among a fixed number of neighbors, we consider all samples within a predefined distance threshold T_{maj}. For each dataset, this threshold is selected based on validation performance, using a procedure similar to the one used for optimizing the input-baseline parameters. The performance gains provided by distance-based majority voting are summarized in Table 3, with the most substantial improvements observed for the three tasks of ISCXVPN2016.

3.6. Results

The performance of the input-space baseline was evaluated against the highest-performing SOTA methods identified in existing literature (more details in Appendix B). To ensure a fair and direct comparison, we did our best to replicate the data processing and splitting methodology used by each selected SOTA approach.

As presented in Table 3, the optimized input-space baseline demonstrates competitive accuracy across a range of datasets. Accuracy was used as the primary evaluation metric, consistent with most related work, with the exception of three comparisons that report F1 scores. The results highlight that the baseline outperforms SOTA methods on two datasets, MIRAGE19 and UTMOBILENET21, and it is only 0.09% behind on the widely used CIC-IDS-2017. Additionally, on most of the remaining datasets, the baseline performs comparably to more sophisticated models.

TABLE 2. THE OPTIMIZED BASELINE PARAMETERS.

Dataset	N	DIR_{scale}	$IPT_{maxclip}$	IPT_{scale}
CESNET-TLS22	7	80	2250	0.045
CESNET-QUIC22	17	102.5	4400	0.015
ISCXVPN2016-A.	5	115	1650	0.015
ISCXVPN2016-T.	7	135	850	0.555
ISCXVPN2016-E.	6	5	750	0.64
MIRAGE19	9	90	200	0.03
MIRAGE22	8	42.5	1400	0.09
UTMOBILENET21	10	10	200	0.02
AppClassNet	20	137.5	-	-
EdgeIIoTset	14	132.5	550	0.005
UNSW-NB15	27	297.5	3150	0.44
CIC-IDS-2017	9	187.5	850	0.335
CIC-DoHBrw	6	67.5	1400	0.285

TABLE 3. Comparison of SOTA and input-space baseline performance across multiple datasets. All values represent an accuracy in percentage or F1 score, denoted by $\mathbb{A}$ and $\mathbb{F}_1$ respectively. Input Space – the kNN performance with default scaling and N=10. Optim. – the performance achieved after Optuna tuning. Distance Maj. – the final result after additional distance-based majority voting.

Dataset	Category	SOTA	Input Space	Δ	Optim.	Δ	Distance Maj.	SOTA Δ
CESNET-TLS22	Web service	[36]: 97.2 ($\mathbb{A}$)	90.96	1.06	92.02	0.32	92.34	−4.86
CESNET-QUIC22	Web service	[35]: 80.87 ($\mathbb{A}$)	55.22	17.37	72.59	0.32	72.91	−7.96
ISCXVPN2016-A.	General traffic	[33]: 79.92 ($\mathbb{A}$)	70.94	0.9	71.84	2.35	74.19	−5.73
ISCXVPN2016-T.	General traffic	[33]: 81.71 ($\mathbb{A}$)	72.92	0.45	73.37	2.34	75.71	−6
ISCXVPN2016-E.	General traffic	[33]: 93.01 ($\mathbb{A}$)	90.61	0.49	91.1	0.48	91.58	−1.43
MIRAGE19	Mobile app	[13]: 80.06 ($\mathbb{F}_1$)	79.98	0.68	80.66	0.12	80.78	0.72
MIRAGE22	Mobile app	[13]: 97.18 ($\mathbb{F}_1$)	95.63	0.64	96.27	0.04	96.31	−0.87
UTMOBILENET21	Mobile app	[37]: 81.91 ($\mathbb{F}_1$)	83.8	−0.02	83.78	0.34	84.12	2.21
UCDAVIS19-Script	Google service	[37]: 98.63 ($\mathbb{A}$)	98		✗		✗	−0.63
UCDAVIS19-Human	Google service	[37]: 80.45 ($\mathbb{A}$)	71.08		✗		✗	−9.37
AppClassNet	Web service	[29]: 88.3 ($\mathbb{A}$)	76.25	4.76	81.01	0	81.01	−7.29
EdgeIIoTset	IDS	[38]: 99.97 ($\mathbb{A}$)	99.79	0	99.79	0.03	99.82	−0.15
UNSW-NB15	IDS	[39]: 98.85 ($\mathbb{A}$)	97.95	0.15	98.1	0.21	98.31	−0.54
CIC-IDS-2017	IDS	[38]: 99.93 ($\mathbb{A}$)	99.8	0.02	99.82	0.02	99.84	−0.09
CIC-DoHBrw	DoH	[40]: 99.81 ($\mathbb{A}$)	98.54	0.02	98.56	0.1	98.66	−1.15

TABLE 4. ISCXVPN2016 comparison with SOTA not using payload data as model input.

Dataset	SOTA w/o Payload	Best Input Space	SOTAΔ
ISCXVPN2016-App	63.92 ($\mathbb{A}$)	74.19	10.27
ISCXVPN2016-TrafficType	65.56 ($\mathbb{A}$)	75.71	10.15
ISCXVPN2016-Encapsulation	85.45 ($\mathbb{A}$)	91.58	6.13

On the ISCXVPN2016 dataset, the baseline shows lower performance relative to the best SOTA result, which incorporates packet-payload features. However, when compared to a SOTA method that uses only packet sequences as input, the baseline significantly outperforms it, as shown in Table 4. The more complex CESNET datasets, collected from a real network environment, present a greater challenge. These datasets involve time-separated captures and more diverse traffic conditions. In this context, the baseline is surpassed by SOTA models, which benefit from stronger generalization capabilities. This result underscores the value of more advanced methods in handling temporal variability and real-world heterogeneity. The UCDAVIS19 dataset also poses difficulties. While the baseline performs comparably to SOTA methods on the Script test subset—whose behavior is more closely aligned with the training set—it underperforms on the Human test subset, where behavioral variation is more pronounced. The AppClassNet dataset is a special case. It features randomized metadata from 20-packet segments across different parts of flows, making it structurally distinct. Despite this specificity, the baseline still achieves a reasonable level of accuracy.

Overall, the results indicate that this simple baseline approach can attain near-optimal performance on several datasets, and in some cases, even surpass more sophisticated SOTA methods. This finding is not an anomaly: it is consistent across various datasets and classification tasks in the field. Furthermore, it suggests that certain tasks, as currently defined by dataset creators, may not be as complex as assumed. The underlying causes of this phenomenon are examined in the following sections.

4. Reason Exploration

On average, the performance gap between input-space baseline and SOTA is −2.88%. This surprisingly strong performance raises an intriguing question: *How is it possible that such a simple model can get this close to complex SOTA methods?* In this section, we investigate the underlying causes and explain phenomena that have long gone unaccounted for, which in turn reveal fundamental properties of the traffic classification problems that set them apart from other machine learning domains.

4.1. Redundant Packet Sequences

We studied sample distributions and uncovered a consistent pattern across all datasets. A substantial fraction of samples are redundant as their feature vectors are exactly identical. In other words, **the datasets are full of duplicates**. The exact fractions of duplicates are shown in Figure 2, with the overall picture being that over 50% of samples are redundant in more than half of the datasets. Moreover, duplicate samples are not confined to a single class. Identical packet sequences appear across multiple classes, creating an unsolvable problem in which the same feature vectors are assigned conflicting labels. The implications for traffic classification can be summarized as follows:

Duplicates within the same class can make a classification task trivial, especially when datasets are split randomly into training, validation, and test sets. Identical samples end up in both training and test sets, which makes the problem trivial to solve, particularly for distance-based classifiers such as the input-space baseline defined in Section 3.1. Possible solutions that are rarely used in related works include *(1)* incorporating duplicate filtering in the evaluation pipeline of traffic classifiers, or, when proposing a new method, *(2)* acknowledging the presence of duplicates and comparing performance to a suitable distance-based baseline, or *(3)* avoiding random splits altogether and instead using time-based or disjoint splits, as discussed in Section 5.1. We believe that ignoring the high number of duplicates will lead to overconfident performance estimates and can stall the progress of traffic classification methods.

Mixed duplicates across multiple classes pose another challenge: for a subset of such samples, correct classification is fundamentally impossible. Potential solutions include adopting multi-label approaches, relabeling duplicate samples, or adding new features to make them distinguishable. We elaborate on this issue in Section 4.2, where we also introduce a new metric for quantifying the extent of the mixed-duplicate problem: *maximum achievable accuracy*. For some datasets, this upper bound can be as low as 93%. Researchers may attempt to surpass these thresholds, unaware that doing so is inherently impossible. In other cases, we observe only a tiny gap between the *maximum achievable accuracy* and the input-space baseline accuracy, indicating that the given task is effectively solved and does not require more complex methods.

A clear correlation emerges between the proportion of redundant flows (green bars in Figure 2) and the best performance achieved with input-space baseline (the *Distance Maj.* column of Table 3). The greater the fraction of duplicates in a given dataset, the better the performance of the input-space baseline. This relationship is further evaluated using the Spearman correlation test. We set the null hypothesis as: "There is no correlation between the performance of input-space baseline and the proportion of duplicate flow within the same class" and chose a significance threshold of $\alpha = 1\%$. Test results are a p-value of 0.0016 and a correlation coefficient of 0.74, so we can reject the null hypothesis. We are sure that behind this confirmed correlation is the effect described earlier: duplicate samples end up in both training and test sets, which makes the classification trivial. Figure 3 and Appendix Figure 4 offer a detailed view into the distribution of duplicate samples based on their packet sequence length, and the following section discusses the underlying causes of duplicate samples in network traffic.

4.1.1. Nature of Network Traffic. Encrypted network communication—when observed through the feature vector of packet metadata sequences—exhibits a rather deterministic nature that can lead to a lot of duplicate samples. Both client and server follow a predefined protocol (TLS handshake, HTTP2, QUIC); different people fetch the same resources with identical API requests; and then there are scripted bots, automatic updates, etc. All these factors contributing to duplicate samples are further amplified for short communications with a small number of exchanged packets.

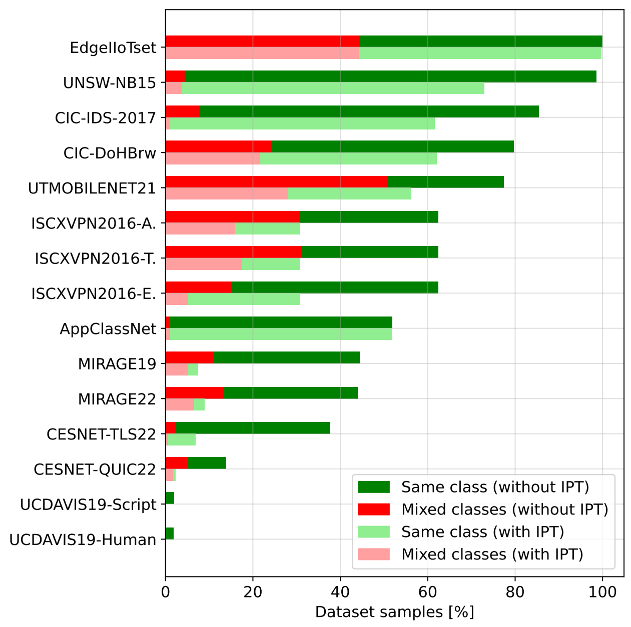

Figure 2. Fractions of duplicate samples in all datasets. Green bars represent duplicates within a single class, while red bars represent clusters of duplicates with conflicting labels. We show two variants: with and without inter-packet times in the feature vector. As expected, there are fewer duplicates when the timing information is used; however, the number of duplicates still exceeded our expectations, given that inter-packet times depend on the fluctuating network conditions and their millisecond resolution.

On the other hand, fluctuating network conditions introduce noise, especially into inter-packet times, but can also affect packet sizes due to packet loss and retransmissions. This is often due to the client's position within the network, with factors such as physical distance, throughput, and network congestion playing a significant part. This phenomenon is evident in datasets captured from real-world environments, such as CESNET or AppClassNet datasets. In AppClassNet, Yang et al. [10] discovered that a single application can exhibit two different "behavior profiles" based on the residential vs. enterprise environments. The question is whether these variations are beneficial for separating traffic classes and, if not, how to train classification models to ignore them. A popular technique is data augmentations, which alter training data to produce more robust models. For instance, Xie et al. [41] proposed a set of TCP-aware augmentations that alternate packet sequences to simulate packet loss, TCP fast retransmissions, or different MSS settings.

4.2. Mixed Duplications and Maximum Achievable Accuracy

We have already established that duplications mixed across multiple classes practically create a multi-label problem. Yet, hardly anyone in the traffic classification domain has ever acknowledged this or taken any countermeasures.

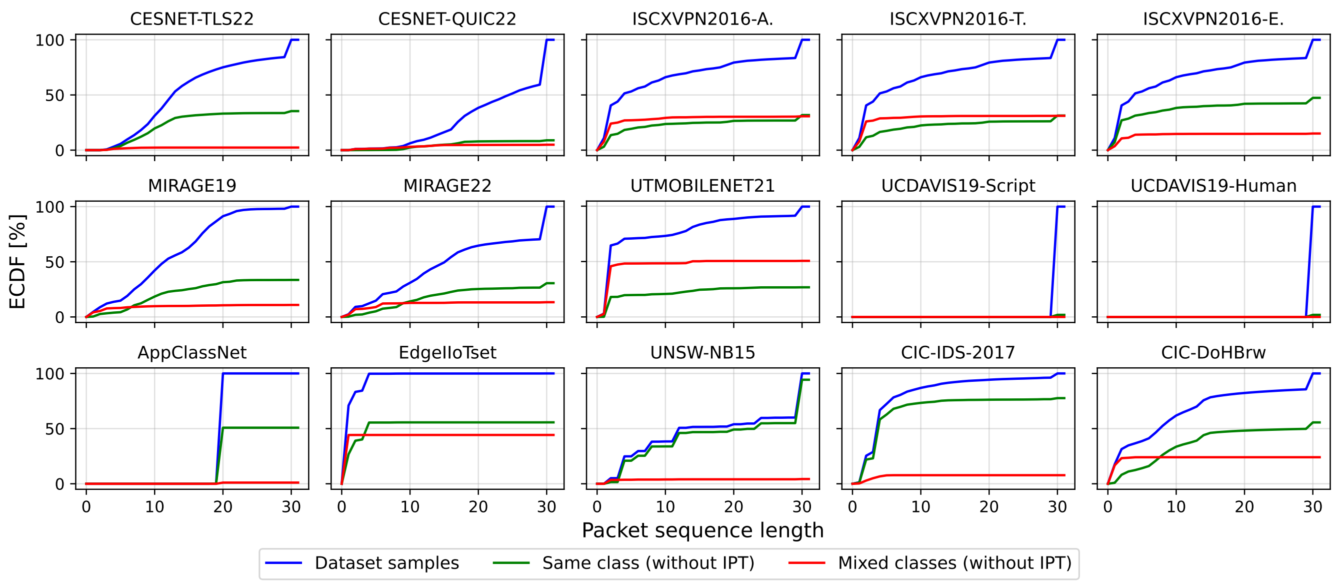

Figure 3. ECDF curves show the fractions of dataset samples, same-class duplications, and mixed-class duplications across different packet sequence lengths. The distribution of packet sequence lengths *(blue)* reveals that some datasets contain fixed-size samples (AppClassNet 20 packets, UCDAVIS19 30 packets)—while others, like EdgeIIoTset and UTMOBILENET21, consist mostly of very short flows. Most mixed-class duplications *(red)* originate from short sequences, whereas same-class duplications *(green)* tend to mirror the overall sample distribution and are spread across all sequence lengths.

In this section, we define a dataset metric called *maximum achievable accuracy*, which takes into account samples that cannot be correctly classified. The expression in Eq. 1 captures the best-case scenario: it assumes a perfect model that correctly classifies all unique (N_{unique}), all redundant samples associated with a single class (N_{same}), and, for each "cluster" of duplicates with conflicting labels (C_i), it predicts the most frequent class. As defined, this metric provides an upper bound on the maximum achievable classification accuracy given the structure of the dataset.

TABLE 5. THE GAP BETWEEN THE MAXIMUM ACCURACY AND THE BEST PERFORMANCE OF INPUT-SPACE BASELINE OF EACH DATASET. ALL METRICS ARE AVERAGES OVER FIVE FOLDS.

Dataset	**Best Input Space**	**Max Acc.**	**Gap Δ**
CESNET-TLS22	92.34	99.41	7.07
CESNET-QUIC22	72.91	99.57	26.66
ISCXVPN2016-A.	74.19	98.87	24.68
ISCXVPN2016-T.	75.71	94.5	18.79
ISCXVPN2016-E.	91.58	98.87	7.29
MIRAGE19	78.56†	96.36	17.8
MIRAGE22	90.23†	96.90	6.67
UTMOBILENET21	78.24†	93.46	15.22
UCDAVIS19-Script	98	100.0	2
UCDAVIS19-Human	71.08	100.0	28.92
AppClassNet	81.01	99.93	18.92
EdgeIIoTset	99.82	99.85	0.03
UNSW-NB15	98.31	99.17	0.86
CIC-IDS-2017	99.84	99.97	0.13
CIC-DoHBrw	98.66	98.79	0.13

† Column *Best Input Space* reuses best results from Table 3, with the exception of MIRAGE19, MIRAGE22, and UTMOBILENET21 datasets. For these, we recomputed accuracy instead of F1-score, and did not use filtering of short flows with fewer than 10 packets.

$$\text{Max Acc.} = \frac{N_{unique} + N_{same} + \sum_{i=1}^{j} \#Maj(C_i)}{N} \quad (1)$$

We computed the maximum achievable accuracy for test sets of all datasets and compared it to the best performance of our input-space baseline. The results are presented in Table 5. For most datasets, the maximum achievable accuracy falls within the range of 98–99%; however, three popular traffic classification datasets—MIRAGE19, MIRAGE22, and UTMOBILENET21—stand out with values of 96.36%, 96.9%, and 93.46%, respectively.

The gap between a simple baseline and the theoretical perfect performance represents room for improvement—a "research playground" for exploring more complex methods. For three particular datasets—all of them focused on IDS—CIC-DoHBrw, CIC-IDS-2017, EdgeIIoTSet, this gap is smaller than 0.15%. This indicates that the classification tasks provided by these datasets are already solved almost perfectly by the baseline method and are therefore unsuitable for benchmarking future research proposals.

IDS tasks are known to be sensitive to the number of false positives due to the risk of alert fatigue. This makes them particularly vulnerable to the mixed-duplicates problem: duplicate samples among both benign and attack classes will inevitably cause a high number of false positives. To evaluate this effect, we estimate[3] the lower bound

3. For the computation of the minimal FPR, we used a slightly different approach: for each mixed cluster of benign and malware samples, we assigned the predicted label as malware, rather than using the majority class as in the computation of the maximum achievable accuracy. We believe this better reflects the behavior expected of an IDS model—namely, prioritizing the detection of malware.

TABLE 6. Analysis of redundancies in datasets focusing on the detection of security threats.

	UNSW-NB15	CIC-IDS-2017
Malicious fraction	3.24 %	20.89 %
Mixed with benign	60.30 %	52.40 %
Minimal FPR	2.65 %	9.12 %

* We excluded the EdgeIIoTset dataset since 90% of its samples are labeled as malicious. Due to this unrealistic fraction of malicious traffic (real environments typically have ratio << 1%), we found it unsuitable for this FPR analysis.

of the false positive rate (FPR) of two IDS datasets. Table 6 shows the estimated minimal FPR, the fraction of malicious samples in each dataset, and the fraction of malicious samples that are mixed with benign traffic. A surprising observation is that in both datasets, more than half of the malware samples are mixed up (i.e., identical samples) with benign traffic, indicating a potential issue in the annotation process. Given this degree of class overlap and the resulting high estimated FPRs, it is difficult to consider these datasets suitable for tuning a realistic intrusion detection system.

5. Discussion

A straightforward application of the k-NN algorithm on raw flow packet sequence metadata—an established data source in network traffic classification—can yield surprisingly high accuracy, closely matching sophisticated SOTA methods. As demonstrated by the results, this is not an isolated case linked to a single task or dataset. Rather, consistently high accuracy is observed across all commonly used datasets in the field.

A closer examination reveals that these datasets contain a significant degree of redundancy. In most datasets, more than 50% of samples have at least one exact counterpart with an identical sequence of packet metadata values. This redundancy appears to be a characteristic of computer network communication. Importantly, this redundancy is not limited to short flows but is also observed in longer communication sequences.

Moreover, exactly the same and thus duplicate samples can also have different class labels. Such phenomena occur in all examined datasets, with the exception of the UCDAVIS19 Human and Script testing subset, which contains fewer samples and follows a different creation process. This cross-class redundancy limits the maximum achievable accuracy. Although 100% accuracy is often viewed as the theoretical upper bound for machine learning models, due to such duplicates with different class labels, it is not achievable for many of the datasets. The true upper limit is often lower, leaving the researchers unknowingly chasing higher performance scores despite the inherent data limitations.

The presented simple baseline method and the estimated upper bounds on accuracy suggest that many of the tasks defined by existing datasets offer only minimal room for improvement. This observation leads us to a central question, reflected in the title of this paper: *Is the network traffic classification in crisis?* Over the past decade, numerous SOTA methods have been introduced using these datasets, yet just a few studies show meaningful improvement beyond what is achievable with a basic baseline.

The evidence presented in the paper highlights redundancy as an intrinsic characteristic of the network traffic, causing the apparent simplicity of the tasks, which are solvable by simple algorithms. However, we argue that the core issue lies in the formulation of the challenges and the design of the experimental protocol. TC research has frequently adopted experimental methodologies from other machine learning domains, such as computer vision and natural language processing. Yet, as demonstrated in this work, the TC domain—along with its datasets—constitutes a distinct branch of data science, where conventional best practices often do not work.

The typical example of the adopted evaluation protocol is the random splitting and shuffling of data samples between training and evaluation dataset parts. While such an approach is generally considered good practice for evaluating generalization, it should not be used in the TC domain. As this study highlighted, the random splitting method is not ideal in the context of network traffic data due to the high likelihood of identical or near-identical samples appearing in both training and evaluation sets. Such practice thus undermines the validity of the results, as models may effectively be tested on data they have already seen and are thus solvable by simpler algorithms.

Based on the findings, the TC research community should thus adopt 1) more sophisticated evaluation protocols and pursue 2) novel approaches and features that would minimize the collision of the data samples from different classes.

5.1. Evaluation Protocols of TC

Experimental protocols must account for sample redundancy as an inherent characteristic of the data and should be designed to reflect the realism of actual deployment scenarios. We can argue that duplicates between training and testing sets are not problematic when such duplicates would naturally occur in real-world usage and are not artifacts of the evaluation setup. Nevertheless, random splitting and shuffling in the context of traffic classification disrupt the temporal and structural dependencies in real-world network traffic, potentially leading to overly optimistic performance estimates and simplifying the tasks. Therefore, we describe below two primary evaluation protocols that should be considered in traffic classification:

5.1.1. Disjoint Entities or Environments in Test and Train Sets. Ensuring that the training and testing datasets do not contain data from overlapping network entities or entire environments is a viable approach in the experimental protocol design. It supports the practical goal of a machine learning algorithm to generalize across previously unseen traffic of a similar type and category. Unfortunately, a disjoint data split requires appropriate support from the dataset, ideally in the form of data provenance [42].

The UCDAVIS19 dataset serves as one example facilitating a disjoint split: while the training set and the Script testing subset are generated similarly, the Human testing subset is created through human interaction with behavioral variability and includes no redundancy with the training set (when considering all packet series metadata).

Some existing datasets, such as the widely used CIC-DoHBrw, can also be redefined to support this approach. Instead of applying random shuffling and splitting for model training and evaluation—as originally proposed by MontazeriShatoori et al. [31] and followed by subsequent studies—the dataset can be partitioned using disjoint IP addresses for both positive and negative classes. For instance, communication with some DoH resolvers can be included in the training set, while the model is evaluated on unseen communications with different resolvers. This reformulated task definition not only reflects practical deployment more accurately but also introduces greater potential for performance improvement, as demonstrated in Table 7.

TABLE 7. Original random shuffle compared to disjoint split on DoHBrw2020 dataset.

Task	Best Input Space	Max Acc.	Δ
Random shuffle	98.66	98.79	0.13
Disjoint split	79.61	98.59	18.98

Such an approach may encourage researchers to propose more generalizable methods, such as that of Jerabek et al. [43], which maintained consistently high performance where others failed, as evidenced in a comparative study [44] evaluated across different datasets, including scenarios with environment changes and disjoint splits.

5.1.2. Time-based Splitting. Time-based data splitting models a realistic deployment scenario by separating training and evaluation datasets chronologically, reflecting how data naturally arrives in an operational environment. The experimental protocol maintains temporal consistency, ensuring that the training data always precedes the testing data. This temporal awareness introduces additional challenges, as it accounts for data drift—the phenomenon where data distributions evolve over time due to factors such as updates, the emergence of new services, or other dynamic changes in the environment.

Nevertheless, time-based splits require datasets organized chronologically, which are relatively scarce. The examples might be CESNET datasets, such as CESNET-TLS22 and CESNET-QUIC22. We evaluated our baseline on the CESNET-QUIC22 dataset, which exhibits a significant data drift during Week 45. As reported by Luxemburk et al. [35], Google changed the certificates for most of its services in that week, causing shifts in feature distributions and a steep decline in model performance.

The results of our evaluation are shown in Table 8. We opted for the 1 week evaluation window and Week 44 was used for training, and following weeks 45, 46, and 47 are used as separate testing sets to show model performance degradation over time. As can be seen, the performance is gradually decreasing in the following weeks. When we compare the results of time-based splits with random splitting, we can see that the random split artificially increases the performance by at least 3.5%.

TABLE 8. Comparison of reported performance between Time-based split and Random and shuffled split. The performance is measured in accuracy. For the time-based split, the whole Week 44 was used for training.

Random Split	**Time-based Split**		Δ
Best Input Space	Test Week	Best Input Space	
79.41	Week 45	75.53	−3.88
	Week 46	71.88	−7.53
	Week 47	70.99	−8.42

5.2. Minimizing the Collisions Between the Data Samples From Different Classes

As illustrated in Figure 3, the proportion of sample collision is naturally higher in communications with fewer packets. Unfortunately, such sparse traffic patterns are characteristic of many malicious activities, which is the primary motivator of traffic classification. Examples include reconnaissance port scans, botnet communications, command-and-control channels, and stealthy data exfiltration. These activities are often deliberately short-lived and crafted to blend in with benign background traffic. As a result, reliably identifying them may require more advanced analytical techniques. We therefore encourage researchers to investigate novel approaches to address the limitations revealed in our study.

For instance, when single-flow classification reaches its limits due to overlaps with benign traffic classes, analyzing sequences of consecutive flows, grouped by protocol or network addresses, could offer a more effective alternative. This approach has the potential to improve detection performance by leveraging temporal or contextual correlations. However, pursuing such approaches requires the creation of new datasets tailored for this purpose. One example is the recently introduced dataset [45], which captures volumetric data from real-world network traffic.

Moreover, machine learning can be supported by different approaches from traditional network traffic processing, such as signature detection. Such heterogeneous solutions may prove more robust and suitable for real-world deployment, as demonstrated by the hybrid DoH detection method of Jerabek et al. [43], or data-fusion for crypto-mining detection proposed by Plny et al. [46]. In contrast, relying exclusively on machine learning, especially without a critical understanding of its limitations as highlighted in this work, may lead to suboptimal or even unusable results.

6. Conclusion

Machine learning has become an essential research area in network traffic classification and threat detection, offering solutions to problems that are otherwise difficult to address. Both traditional models and modern deep learning approaches have demonstrated high performance in this domain. However, recent work by Luxemburk et al. [1] challenges this trend, showing that a much simpler method based on k-NN can achieve comparable or even superior results to SOTA techniques. In this study, we confirmed their findings and extended the evaluation to 12 influential datasets commonly used in traffic classification. According to our results, the input-space baseline method performed on average with -2.88% difference compared to the best SOTA, and outperformed the SOTA in two cases.

Our analysis of the datasets revealed that the strong performance of the baseline is largely driven by extreme data redundancy in TC datasets—a phenomenon that has, until now, been largely overlooked by the community. Commonly used data splitting methodologies often result in identical or nearly identical samples appearing in both the training and test sets. This overlap leads to an overestimation of performance compared to real-world deployment. Moreover, we observed that the exact same samples are frequently assigned different target class labels, making perfect classification inherently unachievable. On four datasets, the performance of the input-space baseline falls within 1% of the maximum achievable accuracy, raising concerns about the relevance of these commonly used datasets as well as the validity of methods evaluated on them.

We highlight the necessity of developing evaluation protocols tailored specifically to the TC research domain. A straightforward adoption of methodologies from other machine learning fields proves inadequate, as demonstrated by the findings in this paper. Researchers should prioritize time-based and disjoint splitting strategies or develop new protocols to more accurately reflect real-world deployment conditions. Furthermore, a comparison with the simple input-space baseline should be an integral part of any evaluation pipeline, ensuring that the benefits of novel methods are substantiated. Unfortunately, the evidence presented in this paper raises serious concerns that TC research may be caught in a cycle of illusory progress. Only rigorous evaluation and revision of the used dataset can break the circle and provide genuine advancements applicable to real-world deployment.

Acknowledgments

This research was supported by the Ministry of Education, Youth and Sports of the Czech Republic in the project “e-Infrastructure CZ” (LM2023054). It was also supported by the Grant Agency of the CTU in Prague, grant No. SGS23/207/OHK3/3T/18.

References

[1] J. Luxemburk, K. Hynek, R. Plný, and T. Čejka, “Universal embedding function for traffic classification via quic domain recognition pretraining: A transfer learning success,” *arXiv preprint arXiv:2502.12930*, 2025.

[2] T. T. Nguyen and G. Armitage, “A survey of techniques for internet traffic classification using machine learning,” *IEEE communications surveys & tutorials*, vol. 10, no. 4, pp. 56–76, 2009.

[3] R. Boutaba, M. A. Salahuddin, N. Limam, S. Ayoubi, N. Shahriar, F. Estrada-Solano, and O. M. Caicedo, “A comprehensive survey on machine learning for networking: evolution, applications and research opportunities,” *Journal of Internet Services and Applications*, vol. 9, no. 1, pp. 1–99, 2018.

[4] N. Williams, S. Zander, and G. Armitage, “A preliminary performance comparison of five machine learning algorithms for practical ip traffic flow classification,” *SIGCOMM Comput. Commun. Rev.*, vol. 36, no. 5, p. 5–16, Oct. 2006. [Online]. Available: https://doi.org/10.1145/1163593.1163596

[5] T. Karagiannis, K. Papagiannaki, and M. Faloutsos, “Blinc: multilevel traffic classification in the dark,” in *Proceedings of the 2005 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, ser. SIGCOMM ’05. New York, NY, USA: Association for Computing Machinery, 2005, p. 229–240. [Online]. Available: https://doi.org/10.1145/1080091.1080119

[6] K. Wang and S. J. Stolfo, *Anomalous Payload-Based Network Intrusion Detection*. Springer Berlin Heidelberg, 2004, p. 203–222.

[7] L. Bernaille, R. Teixeira, I. Akodkenou, A. Soule, and K. Salamatian, “Traffic classification on the fly,” *ACM SIGCOMM Computer Communication Review*, vol. 36, no. 2, pp. 23–26, 2006.

[8] C. Dewes, A. Wichmann, and A. Feldmann, “An analysis of internet chat systems,” in *Proceedings of the 2003 ACM SIGCOMM conference on Internet measurement - IMC ’03*, ser. IMC ’03. ACM Press, 2003, p. 51. [Online]. Available: http://dx.doi.org/10.1145/948205.948214

[9] M. Crotti, M. Dusi, F. Gringoli, and L. Salgarelli, “Traffic classification through simple statistical fingerprinting,” *ACM SIGCOMM Computer Communication Review*, vol. 37, no. 1, pp. 5–16, 2007.

[10] L. Yang, A. Finamore, F. Jun, and D. Rossi, “Deep learning and zero-day traffic classification: Lessons learned from a commercial-grade dataset,” *IEEE Transactions on Network and Service Management*, vol. 18, no. 4, p. 4103–4118, Dec. 2021. [Online]. Available: http://dx.doi.org/10.1109/TNSM.2021.3122940

[11] Z. Chen, B. Yu, Y. Zhang, J. Zhang, and J. Xu, “Automatic mobile application traffic identification by convolutional neural networks,” in *2016 IEEE Trustcom/BigDataSE/ISPA*. IEEE, 2016, pp. 301–307.

[12] W. Wang, M. Zhu, J. Wang, X. Zeng, and Z. Yang, “End-to-end encrypted traffic classification with one-dimensional convolution neural networks,” in *2017 IEEE international conference on intelligence and security informatics (ISI)*. IEEE, 2017, pp. 43–48.

[13] C. Wang, A. Finamore, P. Michiardi, M. Gallo, and D. Rossi, “Data augmentation for traffic classification,” in *International Conference on Passive and Active Network Measurement*. Springer, 2024, pp. 159–186.

[14] E. Akbari, S. A. Tahmid, N. Malekghaini, M. A. Salahuddin, N. Limam, R. Boutaba, B. Mathieu, S. Moteau, and S. Tuffin, “A critical study of few-shot learning for encrypted traffic classification,” in *2023 19th International Conference on Network and Service Management (CNSM)*. IEEE, 2023, pp. 1–9.

[15] N. Malekghaini, E. Akbari, M. A. Salahuddin, N. Limam, R. Boutaba, B. Mathieu, S. Moteau, and S. Tuffin, “Data drift in dl: Lessons learned from encrypted traffic classification,” in *2022 IFIP Networking Conference (IFIP Networking)*. IEEE, 2022, pp. 1–9.

[16] G. Draper-Gil, A. H. Lashkari, M. S. I. Mamun, and A. A. Ghorbani, "Characterization of Encrypted and VPN Traffic using Time-related Features:," in *Proceedings of the 2nd International Conference on Information Systems Security and Privacy*, Rome, Italy, 2016, pp. 407–414.

[17] N. Moustafa and J. Slay, "UNSW-NB15: A comprehensive data set for network intrusion detection systems," in *Proc. Military Communications and Information Systems Conference (MilCIS)*, Canberra, Australia, 2015, pp. 1–6.

[18] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," in *Proc. 4th International Conference on Information Systems Security and Privacy (ICISSP)*. SciTePress, 2018, pp. 108–116.

[19] M. Lanvin, P.-F. Gimenez, Y. Han, F. Majorczyk, L. Mé, and É. Totel, "Errors in the cicids2017 dataset and the significant differences in detection performances it makes," in *International Conference on Risks and Security of Internet and Systems*. Springer, 2022, pp. 18–33.

[20] G. Aceto, D. Ciuonzo, A. Montieri, and A. Pescapé, "Distiller: Encrypted traffic classification via multimodal multitask deep learning," *Journal of Network and Computer Applications*, vol. 183, p. 102985, 2021.

[21] C. Alex, G. Creado, W. Almobaideen, O. A. Alghanam, and M. Saadeh, "A comprehensive survey for iot security datasets taxonomy, classification and machine learning mechanisms," *Computers & Security*, vol. 132, p. 103283, Sep. 2023. [Online]. Available: http://dx.doi.org/10.1016/j.cose.2023.103283

[22] P. Goldschmidt and D. Chudá, "Network intrusion datasets: A survey, limitations, and recommendations," *Computers & Security*, p. 104510, 2025.

[23] J. Luxemburk and T. Čejka, "Fine-grained TLS services classification with reject option," *Computer Networks*, vol. 220, p. 109467, Jan. 2023. [Online]. Available: https://doi.org/10.1016/j.comnet.2022.109467

[24] J. Luxemburk, K. Hynek, T. Čejka, A. Lukačovič, and P. Šiška, "CESNET-QUIC22: A large one-month QUIC network traffic dataset from backbone lines," *Data in Brief*, vol. 46, p. 108888, Feb. 2023. [Online]. Available: https://doi.org/10.1016/j.dib.2023.108888

[25] G. Aceto, D. Ciuonzo, A. Montieri, V. Persico, and A. Pescapé, "MIRAGE: Mobile-app traffic capture and ground-truth creation," in *2019 4th International Conference on Computing, Communications and Security (ICCCS)*, 2019, pp. 1–8.

[26] I. Guarino, G. Aceto, D. Ciuonzo, A. Montieri, V. Persico, and A. Pescapè, "Contextual counters and multimodal deep learning for activity-level traffic classification of mobile communication apps during COVID-19 pandemic," *Computer Networks*, vol. 219, p. 109452, 2022.

[27] Y. Heng, V. Chandrasekhar, and J. G. Andrews, "UTMobileNetTraffic2021: A labeled public network traffic dataset," *IEEE Networking Letters*, vol. 3, no. 3, pp. 156–160, 2021.

[28] S. Rezaei and X. Liu, "How to achieve high classification accuracy with just a few labels: A semi-supervised approach using sampled packets," 2020.

[29] C. Wang, A. Finamore, L. Yang, K. Fauvel, and D. Rossi, "Appclassnet: A commercial-grade dataset for application identification research," *ACM SIGCOMM Computer Communication Review*, vol. 52, no. 3, pp. 19–27, Jul 2022.

[30] M. A. Ferrag, O. Friha, D. Hamouda, L. Maglaras, and H. Janicke, "Edge-IIoTset: A new comprehensive realistic cyber security dataset of iot and iiot applications for centralized and federated learning," *IEEE Access*, vol. 10, pp. 40 281–40 306, 2022.

[31] M. MontazeriShatoori, L. Davidson, G. Kaur, and A. Habibi Lashkari, "Detection of doh tunnels using time-series classification of encrypted traffic," in *5th Cyber Science and Technology Congress (DASC/PiCom/CBDCom/CyberSciTech)*, Vancouver, Canada, Aug 2020, pp. 63–70.

[32] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, "Optuna: A next-generation hyperparameter optimization framework," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, ser. KDD '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 2623–2631. [Online]. Available: https://doi.org/10.1145/3292500.3330701

[33] A. Nascita, A. Montieri, G. Aceto, D. Ciuonzo, V. Persico, and A. Pescapé, "Improving performance, reliability, and feasibility in multimodal multitask traffic classification with xai," *IEEE Transactions on Network and Service Management*, vol. 20, no. 2, pp. 1267–1289, 2023.

[34] J. Johnson, M. Douze, and H. Jégou, "Billion-scale similarity search with GPUs," *IEEE Transactions on Big Data*, vol. 7, no. 3, pp. 535–547, 2019.

[35] J. Luxemburk, K. Hynek, and T. Čejka, "Encrypted traffic classification: the QUIC case," in *2023 7th Network Traffic Measurement and Analysis Conference (TMA)*, 2023, pp. 1–10.

[36] K. Fauvel, F. Chen, and D. Rossi, "A lightweight, efficient and explainable-by-design convolutional neural network for internet traffic classification," ser. KDD '23. New York, NY, USA: Association for Computing Machinery, 2023, p. 4013–4023.

[37] A. Finamore, C. Wang, J. Krolikowski, J. M. Navarro, F. Chen, and D. Rossi, "Replication: Contrastive learning and data augmentation in traffic classification using a FlowPic input representation," ser. IMC '23. New York, NY, USA: Association for Computing Machinery, 2023, p. 36–51.

[38] J. Koumar, K. Hynek, and T. Čejka, "Network traffic classification based on single flow time series analysis," in *2023 19th International Conference on Network and Service Management (CNSM)*. IEEE, 2023, pp. 1–7.

[39] J. Koumar, K. Hynek, J. Pešek, and T. Čejka, "Nettisa: Extended ip flow with time-series features for universal bandwidth-constrained high-speed network traffic classification," *Computer Networks*, vol. 240, p. 110147, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1389128623005923

[40] R. Mitsuhashi, A. Satoh, Y. Jin, K. Iida, T. Shinagawa, and Y. Takai, "Identifying malicious dns tunnel tools from doh traffic using hierarchical machine learning classification," in *Information Security: 24th International Conference, ISC 2021, Virtual Event, November 10–12, 2021, Proceedings 24*. Springer, 2021, pp. 238–256.

[41] R. Xie, Y. Wang, J. Cao, E. Dong, M. Xu, K. Sun, Q. Li, L. Shen, and M. Zhang, "Rosetta: Enabling robust tls encrypted traffic classification in diverse network environments with tcp-aware traffic augmentation," in *Proceedings of the ACM Turing Award Celebration Conference - China 2023*, ser. ACM TURC '23. New York, NY, USA: Association for Computing Machinery, 2023, p. 131–132. [Online]. Available: https://doi.org/10.1145/3603165.3607437

[42] K. Werder, B. Ramesh, and R. Zhang, "Establishing data provenance for responsible artificial intelligence systems," *ACM Transactions on Management Information Systems (TMIS)*, vol. 13, no. 2, pp. 1–23, 2022.

[43] K. Jerabek, K. Hynek, O. Rysavy, and I. Burgetova, "Dns over https detection using standard flow telemetry," *IEEE Access*, vol. 11, pp. 50 000–50 012, 2023.

[44] K. Jerabek, K. Hynek, and O. Rysavy, "Comparative analysis of dns over https detectors," *Computer Networks*, vol. 247, p. 110452, 2024.

[45] J. Koumar, K. Hynek, T. Čejka, and P. Šiška, "Cesnet-timeseries24: Time series dataset for network traffic anomaly detection and forecasting," *Scientific Data*, vol. 12, no. 1, p. 338, 2025.

[46] R. Plný, K. Hynek, and T. Čejka, "Decrypto: Finding cryptocurrency miners on isp networks," in *Nordic Conference on Secure IT Systems*. Springer, 2022, pp. 139–158.

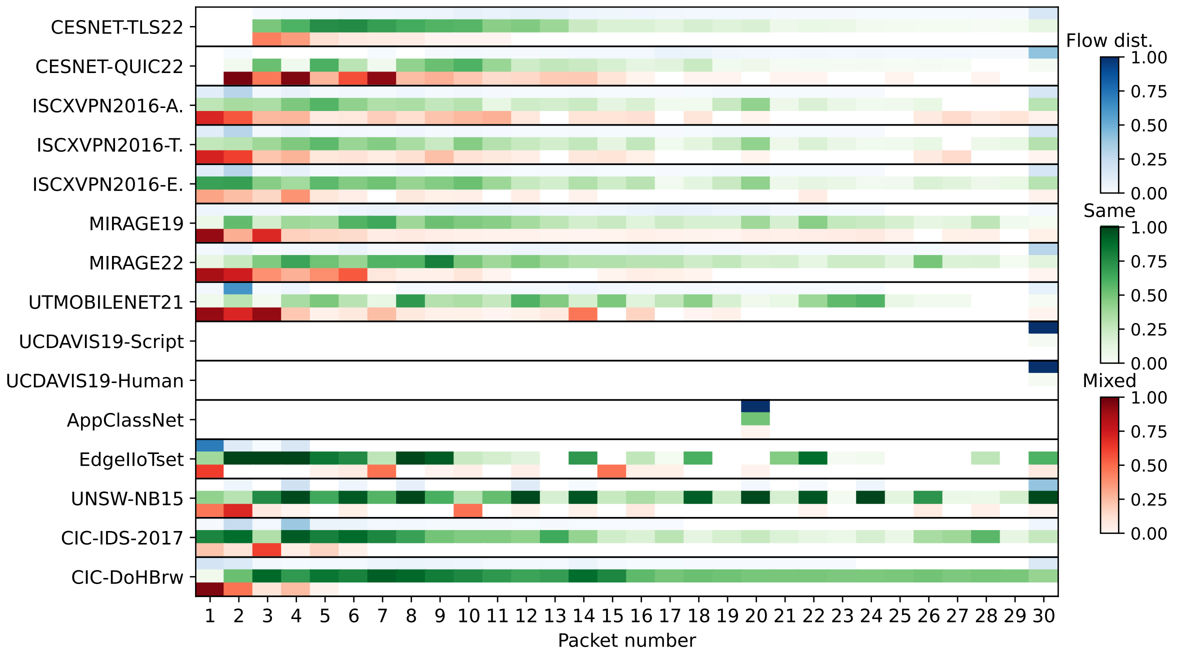

Figure 4. All datasets redundancy heatmap without IPT. The color-coded redundancy ratios are calculated across all packet sequences of equal length. Blue represents a fraction of all flows in the dataset, while green (same-class duplicates) and red (mixed-class duplicates) represent the fraction of redundant samples within flows of a given length.

Appendix A - Heatmap

Figure 3 provides a heatmap offering a detailed view of duplicate distributions across different flow lengths. In some datasets, such as ISCXVPN2016, UNSW-NB15, and CIC-DoHBrw, the fraction of redundant samples remains relatively stable regardless of sequence length. Mixed-class redundancy, however, exhibits a clearer trend: it declines significantly as flow length increases, although it never fully disappears. This indicates that sample redundancy is not confined to short flows but persists even in longer sequences, including those with 30 or more packets.

Appendix B - Details for SOTA comparison

For the sake of completeness, we indicate the exact tables of the referenced manuscripts from which the information on the best-performing classifiers for each dataset was obtained. We refer accuracy measure as $\mathbb{A}$, and weighted F_1 score as $\mathbb{F}_1$.

CESNET-TLS22, $\mathbb{A}$ – Table 5 of Fauvel et al. [36]. The best result of 97.2% is achieved with the LEXNet architecture.

CESNET-QUIC22, $\mathbb{A}$ – Table 1 of Luxemburk et al. [35]. The best result of 80.87% is achieved with LightGBM.

ISCXVPN2016, $\mathbb{A}$ – Table 3 of Nascita et al. [33]. The best results are achieved with the DISTILLER-Embeddings architecture, which uses payload as model input. For a payload-less model comparison, we use 1D-CNN (PSQ) of the same table.

MIRAGE19, $\mathbb{F}_1$ – Table 7 of Wang et al. [13]. Deltas from this table need to be added to the baseline performance of 75.43%. The best result is "MaskedStack ($p = 0.7$)" with 80.06% (75.43% + 4.63%).

MIRAGE22, $\mathbb{F}_1$ – Table 7 of Wang et al. [13]. Deltas from this table need to be added to the baseline performance of 94.92%. The best result is "MaskedStack ($p = 0.3$)" with 97.18% (94.92% + 2.26%).

UTMOBILENET21, $\mathbb{F}_1$ – Table 8 of Finamore et al. [37]. The best result for the $>$10pkts version is "Time shift" with 81.91%.

UCDAVIS19, $\mathbb{A}$ – Table 7 of Finamore et al. [37] containing results for an enlarged training set. The best result for *human* is "SimCLR + fine-tuning" with 80.45%, and for *script*, it is "Packet loss" with 98.63%.

AppClassNet, $\mathbb{A}$ – Table 3 of Wang et al. [29], which is the paper introducing the dataset. The best accuracy of 88.3% on the public version of AppClassNet is achieved with random forest.

UNSW-NB15, $\mathbb{A}$ – Table A.9 of Koumar et al. [39]. The paper introduces a new feature vector based on time-series features. Authors then used this feature vector for the LightGBM model, which yielded reported accuracy scores.

EdgeIIoTset, CIC-IDS-2017, $\mathbb{A}$ – Table 3 at Koumar et al. [38]; authors use time-series based features with XGBoost model.

CIC-DoHBrw, $\mathbb{A}$ – Table 4 of Mitsuhashi et al. [40]. Accuracy of filtering DoH (binary task), with XGBoost model, using rather classical statistical extension of flow.