The Cost of Packet Loss on ML-Based Traffic Analysis

Johann Hugon
ENS Lyon

Paul Schmitt
Cal Poly

Francesco Bronzino
ENS Lyon

***Abstract*—Machine Learning (ML)-based traffic analysis relies on a data processing pipeline consisting of multiple steps that filter, process, and collect statistics, or features from raw network traffic. These steps are typically performed by in-network measurement systems deployed in existing network fabric (e.g., programmable switches) or using off-the-shelf hardware (e.g., commodity servers). In both deployment scenarios, these systems come with limited processing budgets that must be finely tuned to precisely collect the required features. Unfortunately, the ever growing traffic volume on modern networks can exhaust these budgets, ultimately resulting in packet loss. In this paper, we investigate the impact of packet loss on the performance of ML-based traffic analysis systems. As losses introduce bias in the final features set provided to the machine learning model, we hypothesize that they will negatively impact model performance. We evaluate this hypothesis by analyzing the performance of two different ML models—service classification and QoE analysis—trained on a dataset of video flows, and we measure the impact of two different packet loss models: probabilistic and bursty losses. Our results show that sporadic packet loss has little impact on performance. Conversely, bursty losses, which are more common for packet processing systems, can lead to a significant negative impact.**

I. Introduction

Machine Learning (ML) has become an essential tool for network traffic analysis, enabling tasks ranging from service classification to intrusion detection with unprecedented accuracy [1]. Recent studies have demonstrated ML's effectiveness in analyzing high-speed and encrypted traffic [2], [3], [4], where traditional rule-based approaches often fail [5]. However, deploying ML systems for traffic analysis on live, high-speed links requires a preprocessing pipeline that is able to operate within strict system constraints [6], [7], [8]. Preprocessing pipelines typically consist of several critical stages: packet extraction from the link, protocol parsing, filtering, and feature computation. These operations can be performed either in real-time (online) or asynchronously (offline), with each approach presenting distinct challenges. Yet, in both scenarios, the systems responsible for these actions face fundamental operational constraints that can lead to packet loss, which has the potential of compromising the quality, or even correctness, of features provided to downstream ML models.

Various approaches to network traffic analysis face different constraints. Traditional tools like Tcpdump[9] allow capturing network traffic, but often struggle with high-speed links due to kernel network stack bottlenecks [10]. While storing traces for offline usage might reduce processing and memory constraints, the cost of uploading large data volumes to a remote location becomes critical during high traffic times. More modern techniques [11], such as zero-copy packet processing through technologies like DPDK [12] or XDP [13], can prevent memory and storage exhaustion by reading packets directly from NIC memory or cache, but require the entire pipeline to process packets at line-rate—when processing capacity cannot match network throughput, packets are inevitably dropped, resulting in information loss that propagates to ML model features. In-network solutions like in-switch ML [14], [15] offer an alternative approach but provide very limited resources due to the need to perform routing in parallel with analysis tasks.

Of course, the dynamic nature of network traffic makes it nearly impossible to design measurement systems that operate without packet loss under all conditions, if not at the cost of reduced model accuracy. Even pipelines that perform well in controlled environments often fail when confronted with real-world traffic pattern[16], [17]. This raises a fundamental, yet under-explored question: *How does packet loss affect the performance of ML-based traffic analysis systems?* Recent work [18] has shown that packet loss can have significant effects on the performance of ML models. However, while this work proposes solutions to mitigate such degradation, it fails to provide a comprehensive understanding of the underlying impact that packet loss has on the performance of ML models. In this paper, we aim to fill this gap by answering two key questions:

Q1. Does packet loss affect model performance independently of its distribution? Recent work by Babaria et al. [18] has demonstrated that packet loss can heavily impact accuracy for service identification models. However, they solely used a simplistic probabilistic model for loss, where every packet is considered independently. In contrast, in operational environments, packet loss typically occurs in bursts during periods of system stress [19]. Rather than isolated drops, these bursts—ranging from a few packets to thousands—disrupt the temporal continuity of features. We hypothesize that this pattern of loss more accurately reflects real-world conditions and has a disproportionately negative impact on model performance compared to random individual packet drops.

Q2. In the presence of loss, does accuracy degradation depend on ML model input features? In their paper, Babaria et al. [18] discuss how lost packets lead to missing or distorted features that may compromise model accuracy

979-8-3315-1478-5/25/$31.00 ©2025 IEEE

for various service identification solutions. Without complex (i.e., computationally expensive) stateful connection tracking, these inconsistencies become difficult to identify. For example, in DPDK-based systems, incoming packets can overwrite older, still unread packets in the buffer before processing completes, creating undetected gaps in the data. In such cases, the model may receive features that are inconsistent with the true state of the network. However, while these distortions can be evident in the case of service identification, where only a handful of packets are used to build features, they might be less evident for scenarios like video startup time inference where features are mostly aggregates collected across many (i.e., hundreds if not thousands) of packets.

To answer these questions, we focus our work on two distinct ML-based traffic analysis tasks: (1) service identification, a classification problem; and (2) video startup delay inference, a regression problem. For these use cases, we study how packet loss affects performance metrics for these by applying both a probabilistic model that randomly drops individual packets, as well as a two-state Markov chain model that simulates realistic bursts of packet drops. We find that the impact of sporadic drops is noticeable, but negligible below low thresholds. However, bursty drops can have a large impact on model performance, even with a low probability of occurrence.

The remainder of this paper is organized as follows: Section II discusses relevant prior work; Section III details our experimental methodology; Section IV presents results across both use cases; finally, Section V summarizes our findings and their implications. All analysis code and results are made available as a Jupyter notebook[1] to ensure reproducibility.

II. Related Work

The quality of datasets is widely recognized as a crucial factor affecting the overall performance of ML models [20], [21]. While prior research (such as Mauri et al. [22]) has investigated the impact of data poisoning, where adversaries deliberately modify training data, our work specifically addresses data quality issues at inference time rather than during model training.

Foroni et al. [23] generated variations of the same dataset with different noise levels to measure the impact on task performance. Our approach is similar in that we introduce controlled noise—specifically packet loss—at predetermined rates to evaluate performance degradation. However, our work is distinguished by its application to network systems and consideration of domain-specific constraints such as bursty packet losses.

More recently, Cavitt et al. [24] investigated the negative effects of packet drops in power systems. They employed machine learning models to detect losses and implemented various replacement policies to mitigate the impact. Their research demonstrated that performance degradation could be minimized through appropriate data replacement strategies.

[1] https://github.com/ENSL-NS/Cost-of-Packet-Loss

Service Identification	Video startup delay inference
Packets size, count	Packets size, number
Packets inter arrival time	Throughput
Bytes per packets	Segments size, count
TCP Flags, Window, RTT	Segments duration
Bytes per packet	Inter segments time
Bytes in flight	
Retransmissions	

TABLE I: Features Sets

While their work focused on spoof detection in power systems, our study examines detectable but untraceable losses in network environments.

Yang et al. [25] addressed packet loss in encrypted traffic classification by developing an Anti-Packet-Loss method based on a Masked Autoencoder. Their approach intentionally masks portions of training traffic data to enhance the encoder's ability to reconstruct missing information. Their evaluation showed 90% classification accuracy even with 15% packet loss, significantly outperforming conventional deep learning methods. Although their research targets the same problem of packet loss during inference, our work differs by evaluating the impact on machine learning models that are not specifically designed to handle packet drops.

Most recently, Babaria et al. [18] proposed FastFlow a solution to mitigate the effects of packet loss on ML models for service identification. FastFlow employs a sequential decision-based classification model that leverages a collection of LSTM models trained with reinforcement learning to infer traffic classes using the first few packets of a flow. In the evaluation, the authors show that FastFlow is more resilient to packet loss than other state-of-the-art ML models. However, their work solely applies per-packet probabilistic losses and does not provide a comprehensive understanding of the underlying impact that packet loss has on the performance of ML models. To the best of our knowledge, our work is the first to systematically study the impact of packet loss on ML-based traffic analysis systems, using both probabilistic and bursty loss models and on multiple traffic analysis tasks.

III. Methodology

In this section, we discuss the methodology used to evaluate the impact of packet loss on ML-based traffic analysis systems. We begin by describing the two use cases we selected for our analysis: service identification and video startup time. We then discuss the dataset used for our analysis, including the packet loss models applied.

A. Traffic analysis tasks

We focus our analysis on two well-studied tasks in the network community: Service identification [26] and ideo startup time inference [2].

Service identification. Service identification is a typical traffic classification task [4], [27]. The task involves classifying network traffic at the flow level into corresponding applications, such as YouTube and Netflix. We focus particularly

on early application identification, which consists of using the first few packets—typically ten or fewer—to identify the application [26]. In our scenario, we utilize the first ten packets and derive the set of features described in Table I. These features are based on network parameters such as byte quantity in each direction, packet inter-arrival time, and transport metrics like TCP windows or bytes in flight. All features listed in Table I are divided by direction: client-to-server and server-to-client. Where possible, they are further subdivided into various statistical metrics, including standard deviation, average, maximum, minimum, median, kurtosis, and skewness. This approach aims to capture hidden patterns that raw features may not effectively reveal. While some existing works use fewer packets (e.g., only the first four), we elected to use ten packets as a best-case scenario, recognizing that using fewer packets would render the analysis even more susceptible to packet loss and distortion of input features.

Video startup time inference. Video startup time inference is a fundamental task in Quality of Experience (QoE) analysis that involves predicting the duration between a user's request to play a video and the moment playback actually begins. This metric is widely recognized as a critical factor affecting user satisfaction and engagement with video streaming services. In our scenario, we leverage network traffic features collected during the initial connection and buffering phases to infer the startup delay experienced by users. Unlike service identification, which relies on the first few packets, video QoE inference typically utilizes larger temporal windows of aggregated data—generally spanning multiple seconds, such as ten-second intervals. We specifically focus on startup time prediction for two strategic reasons: first, it enables us to validate the hypothesis that larger windows of aggregated data might exhibit greater resilience to packet loss; and second, it presents a regression problem rather than classification, thereby allowing us to evaluate the impact of packet loss on a different class of machine learning tasks with continuous output variables. For this use case, we do not consider features collected from the transport layer, as they have been demonstrated to be less effective for this task [2]. Instead, we focus on features related to the video segments, such as the number of segments downloaded, the size of each segment, and the time taken to download each segment. We also include features related to the inter-segment time, which is the time between the end of one segment and the start of the next. This approach allows us to capture the dynamics of video streaming and how they relate to startup delays. Finally, as listed in Table I, we extract features about traffic volumes.

B. Dataset and Model Training

Dataset. Our analysis is based on a subset of the dataset collected by Bronzino et al. [2] in their work on video quality inference. The dataset consists of 9,213 labeled traces from four major video streaming providers (Netflix, YouTube, Amazon Prime Video, and Twitch), which we split into two parts: a training set comprising 7,390 traces (80.21%) and a testing set containing 1,823 traces (19.79%), with some traces excluded due to labeling issues. Subsequently, each trace was divided into separate traces for individual flows, and the data was filtered to include only video-related flows. This process yielded a training dataset of 2,535,163 flows and a testing dataset of 627,524 flows, maintaining a similar split ratio of 19.84%. The slight variation in percentages occurs because some traces contain more flows than others.

Model training. For both tasks, we selected Random Forest [28] as classification and regression algorithm, as previous research has demonstrated its superior precision and recall with lower false positive rates for our use cases [26], [2]. For model training, we employ AutoGluon [29], a widely adopted AutoML library. AutoGluon automatically explores effective combinations of input features and hyperparameters to generate the most effective model. Note that the goal of this paper is not to achieve the best possible model performance, but rather to evaluate the impact of packet loss on the performance of ML models. Therefore, we did not perform any hyperparameter tuning or feature selection beyond what AutoGluon provided. To mitigate overfitting, we constrained the minimum sample split to 10 and limited the maximum depth of the trees to 10. These constraints reduce the likelihood of capturing insignificant noise patterns and enhance the model's generalization capabilities. The model was trained on 80% of the dataset without any application of packet loss, ensuring that it learned from clean, uncompromised data. Packet loss was introduced exclusively during testing to simulate real-world inference conditions, based on the assumption that the model had been trained on a lossless dataset. We evaluated the impact of varying packet loss rates on model performance using the weighted F1 score as our primary metric for classification. This weighted scoring approach was chosen to prevent misinterpretation of results due to class imbalance in the dataset, while the F1 score itself provides a balanced measure that accounts for both precision and recall. For the regression task, we used the error distributions to observe a detailed analysis of the model's performance.

C. Loss Models

We aim to evaluate the impact of packet drop on model performance by implementing two distinct loss models: a probabilistic model and a bursty model.

Probabilistic loss model. In this initial model, we establish a probability p of packet drop. This probability is applied independently to each packet within a flow and to each flow within the dataset. This approach yields an aggregate loss measure for the dataset, expressed as a percentage of total packets lost. By increasing the value of p, we systematically increase the overall percentage of loss in the dataset. While percentage of packet loss is a common evaluation metric in network systems, this approach has certain limitations, particularly in its inability to adequately capture the nuanced behavior of packet loss within real-world network pipelines.

Bursty loss model. The second approach aims to more effectively characterize burst drops by representing them as

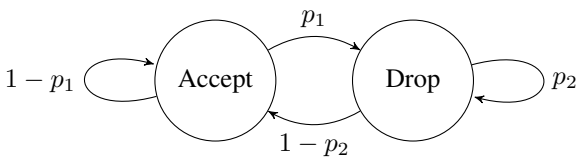


Fig. 1: State Transition Diagram of the Burst Models

a two-state Markov chain. Markov chains have been widely employed to model packet drops for decades [30], [31]. They are particularly effective for describing bursts of drops by defining a "good" state and a "bad" state, along with the various probabilities of transitioning between them or remaining in the current state. In our implementation, these two states correspond to the Accept state and the Drop state, as illustrated in Figure 1. As shown in the figure, p_1 represents the probability of transitioning from the Accept state to the Drop state, while p_2 denotes the probability of remaining in the Drop state. By calibrating these transition probabilities, we can simulate different frequencies and intensities of burst drops. A high value of p_1 increases the likelihood of burst initiation, representing the probability of network degradation in a system. Meanwhile, p_2 represents the probability of continuing the burst of dropping packets, which leads us to interpret $1 - p_2$ as the probability of resolving this drop condition. The lower the value of p_2, the more rapidly the pipeline will respond to the burst and recover from the drop condition.

Application of loss models to the dataset. We apply our two loss models to the test portion of the dataset, which comprises 627,524 flows containing a total of 201,164,351 packets. The resulting packet drop statistics are presented in Tables II and III. In Table II, the first column represents the probability p of dropping a packet, the second column shows the total number of remaining packets after applying the losses, the third column displays the number of packets dropped, the fourth column indicates the resulting percentage of packet loss, and the final column the quantity of missing flows. Table III follows a similar structure, with the distinction that the first column p from Table II is replaced by two columns, p_1 and p_2, representing the transition probabilities of our two-state Markov chain. The data demonstrate that the probabilistic drop closely aligns with the overall percentage of packet loss observed in the trace. Notably, even a relatively low probability of burst initiation (p_1) can result in substantial packet loss, exceeding 10% of the original trace.

It is important to note that the loss models are applied before extracting the first N packets of a flow. Consequently, in scenarios where machine learning (ML) models require data from the first ten packets and packets between positions five and nine are dropped, features are subsequently extracted from packets zero to four and from ten to fourteen. This approach ensures that the ML model consistently receives the same quantity of packets for each flow, maintaining consistent input dimensionality despite varying packet loss patterns. However,

p	# Packets	Diff	%	# Missing Flows
0.0	200,164,351	0	0.00	0
0.005	199,164,584	999,767	0.50	145
0.01	198,162,375	2,001,976	1.00	288
0.02	196,158,686	4,005,665	2.00	446
0.05	190,155,893	10,008,458	5.00	2,680
0.1	180,148,648	20,015,703	10.00	9,946

TABLE II: Traces after application of probabilistic loss

p_1	$1 - p_2$	# Packets	Diff	%	# Missing Flows
0.0	0.0	200,164,351	0	0.00	0
0.005	0.1	190,644,830	9,519,521	4.76	10,800
	0.01	133,593,296	66,571,055	33.26	168,647
	0.001	33,560,174	166,604,177	83.23	399,910
	0.0001	4,285,699	195,878,652	97.86	506,266
0.01	0.1	181,957,332	18,207,019	9.10	25,673
	0.01	100,084,829	100,079,522	50.00	271,475
	0.001	18,320,521	181,843,830	90.85	474,142
	0.0001	2,186,921	197,977,430	98.91	551,860

TABLE III: Traces after application of burst loss

as mentioned in II and III, a large drop can lead to dropping the entire connection. Thus, it is mandatory to include these missing flows when evaluating the performance of the models, as forgetting them can lead to high accuracy despite a high drop rate.

IV. Analysis

In this section, we describe our analysis on the two use-cases: service identification and video quality inference. For both use-cases, we first present results on probabilistic losses, followed by results on bursty losses.

A. Service Identification

Probabilistic. We apply the probabilistic loss model to the test partition of the dataset, with results presented in Table IV. In this table, $p = 0.0$ represents the baseline scenario with no packet drops, where data is served to the model without alteration. This serves as our benchmark for comparison; the F1 score reaches 0.971 under these optimal conditions. We note that accuracy and F1 score exhibit similar values, indicating both strong precision and recall across most classes, even without any packet drops. We also highlight that, for results with packet drops, we treat the missing connections as misclassified to avoid artificially inflating performance metrics.

At $p = 0.005$, we observe only a minor reduction in F1 score, which remains robust at 0.967. A slightly higher loss rate of $p = 0.01$ further decreases the F1 score to 0.963, though the impact remains relatively small. However, when the loss rate increases to $p = 0.02$, the F1 score drops more noticeably to 0.954, indicating the beginning of performance degradation. Beyond this threshold, the decline becomes more pronounced, with the F1 score decreasing to 0.925 at $p = 0.05$ and further to 0.879 at $p = 0.1$. These results demonstrate that while low levels of packet loss ($p < 0.02$) have minimal effect on classification performance, significant degradation occurs once packet loss exceeds $p = 0.02$, with substantial

p	Accuracy	F1 Score
0.0	0.972	0.972
0.005	0.967	0.967
0.01	0.964	0.963
0.02	0.954	0.954
0.05	0.927	0.925
0.1	0.885	0.879

TABLE IV: Impact of probabilistic losses on Service Identification use case

p_1	$1-p_2$	Accuracy	F1 Score
0.0	0.0	0.972	0.972
0.005	0.1	0.926	0.918
	0.01	0.618	0.551
	0.001	0.265	0.208
	0.0001	0.152	0.108
0.01	0.1	0.878	0.862
	0.01	0.432	0.363
	0.001	0.159	0.117
	0.0001	0.087	0.061

TABLE V: Impact of burst losses on Service Identification use case

performance decline at higher loss rates. However, while the impact is noticeable, we observe that results remain acceptable even at high loss rates, with F1 scores close or above to 0.8.

Burst. As illustrated in Table V, we apply the burst model to the test dataset. First, we notice that even when $1-p_2$ is low, the F1 score remains more affected than with probabilistic losses, at equivalent percentages of drops. This can be explained by examining Table III, where drops for $p_1 = 0.005$ and $p_2 = 0.1$ eliminate only 4.76% of the overall packets while missing $10,800$ flows, whereas its equivalent in the probabilistic model at $p = 0.05$ drops 5% of the traffic for merely $2,680$ flows. This demonstrates the potential context loss caused by burst patterns. When burst losses occur, the model may miss entire connections or long portions of them, leading to a more significant impact on performance.

However, as $1-p_2$ reaches 0.01, the F1 score greatly drops, falling below 0.6. This suggests that burst loss, even at moderate levels, has a more detrimental effect on performance compared to the probabilistic loss model. As $1-p_2$ increases further to 0.001, where a large portion of the traffic has been dropped, the decline in performance becomes more pronounced. This indicates that burst loss patterns have a significant negative impact on the model's ability to maintain performance, even with a slightly lower probability of occurrence.

B. Video Startup Delay Inference

Probabilistic. Similarly to the previous use case, we apply the probabilistic loss model to the test partition of the dataset, with results presented in Figure 2a. In the figure, we show error distributions for the regression task, as the difference between the real startup time collected from the video player and the predicted startup time from the model. The box plots show both median and interquartile ranges (IQR). We observe a slight impact on the median error as p increases. Starting at -129 for $p = 0$, we observe a phase of stagnation with a median at -140 for $p = 0.005$; -144 for $p = 0.01$; it begins to drop at $p = 0.02$ with a median of -170; it further decreases at $p = 0.05$ to -246 before ultimately reaching -344 at $p = 0.1$. Overall, while the error increses, the median remains relatively stable until $p = 0.05$. This indicates that the model remains relatively consistent while suffering from packet drops. While we observe greater variation for IQR values, symptomatic of the model predicting less consistent values, the overall performance remains relatively stable. This confirms the intuition that using aggregates over many packets makes the model less sensitive to sporadic packet loss.

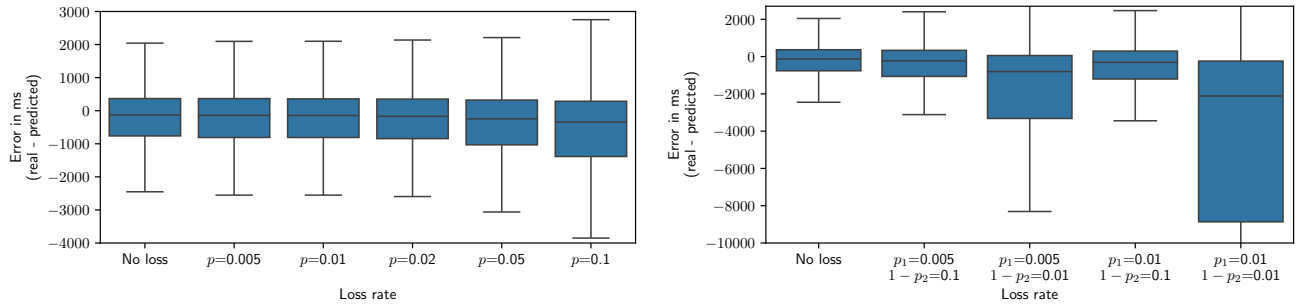
Burst. In contrast to probabilistic losses, for the burst loss we observe a much greater impact as soon as we reach $1-p_2 = 0.1$ (as illustrated in Figure 2b), with the median dropping to -230 and -308. We do not show in the figure the results for higher burst lengths for space saving reasons. However, we observe that the median drops heavily to -806 for $p_1 = 0.005$ and -2114 for $p_1 = 0.01$. This indicates that the model is significantly affected by burst losses, even at low probabilities of occurrence. The IQR also increases significantly, indicating a wider range of errors in the predictions.

To understand the reason behind this behavior, we analyze the features used by the model. We first study the feature importance obtained at training time using the mean and standard deviation of accumulated impurity decrease within each tree of the Random Forest model. We observe that the most important features all relate to the download of video segments during the initial streaming phase. This is reasonable, as the model attempts to capture how quickly the video player is capable of downloading the necessary video buffer required to start reproduction. In particular, we observe that the most important feature is the average time to download a video segment, which effectively represents this dynamic (i.e., smaller values suggest higher download rates). We then study the distribution of the values of this feature across different test datasets (shown in Figure 3). We observe that the distribution of the feature is significantly affected by the burst loss model, with median values increasing as drops increase. This result suggests that key packets, used for the segment identification technique [2], are being lost, leading to a significant increase in the detected time to download video segments.

Overall, this result shows that, contrary to expectations, even a model that relies on aggregates over many packets can be significantly affected by burst losses.

V. Conclusion

In conclusion, this study provides valuable insights into how packet loss within the feature extraction pipeline impacts machine learning model performance. We present comprehensive results for two use cases, evaluated under both probabilistic and burst loss models. Our findings demonstrate that while probabilistic losses with low p values ($p < 0.02$) have minimal impact on model performance, burst losses cause significantly



(a) Probabilistic loss

(b) Burst loss

Fig. 2: Impact of loss on the ideo Startup Delay Inference use case

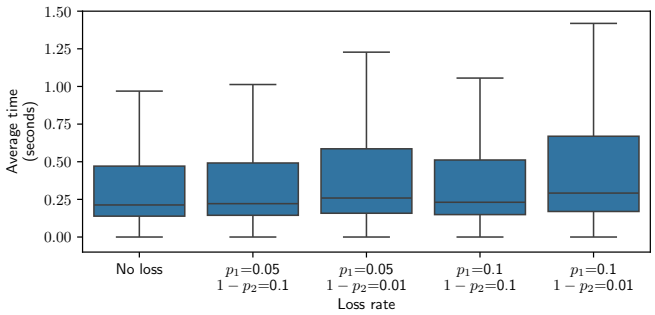


Fig. 3: Average time to download a video segment

more degradation even at equivalent drop rates. This holds true for both service identification and video startup delay inference tasks, counter-intuitively to the notion that models that use aggregate statistics as input features are less affected by packet loss. To enhance transparency and reproducibility, after acceptance we provide a Jupyter notebook[2] containing all code, data, analyses, and visualizations to enable replication of our findings.

References

[1] M. A. Ridwan *et al.*, "Applications of machine learning in networking: A survey of current issues and future challenges," *IEEE Access*, 2021.

[2] F. Bronzino *et al.*, "Inferring streaming video quality from encrypted traffic: Practical models and deployment experience," *Proc. ACM Meas. Anal. Comput. Syst.*, 2019.

[3] T. Mangla *et al.*, "emimic: Estimating http-based video qoe metrics from encrypted network traffic," in *Network Traffic Measurement and Analysis Conference (TMA)*, 2018.

[4] R. T. Elmaghraby *et al.*, "Encrypted network traffic classification based on machine learning," *Ain Shams Engineering Journal*, 2024.

[5] M. Shen *et al.*, "Machine learning-powered encrypted network traffic analysis: A comprehensive survey," *IEEE Communications Surveys & Tutorials*, 2023.

[6] T. Swamy *et al.*, "Homunculus: Auto-generating efficient data-plane ml pipelines for datacenter networks," in *Proceedings of the 28th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 3*, 2023.

[7] S. Liu *et al.*, "Serveflow: A fast-slow model architecture for network traffic analysis," 2024.

[8] G. Wan *et al.*, "Cato: End-to-end optimization of ml-based traffic analysis pipelines," 2024.

[2]https://github.com/ENSL-NS/Cost-of-Packet-Loss

[9] The Tcpdump Group, "Tcpdump," https://www.tcpdump.org/, 2025.

[10] Q. Cai *et al.*, "Understanding host network stack overheads," in *Proceedings of the 2021 ACM SIGCOMM 2021 Conference*, 2021.

[11] D. Cerović *et al.*, "Fast packet processing: A survey," *IEEE Communications Surveys & Tutorials*, 2018.

[12] The DPDK Project, "Data Plane Development Kit," https://www.dpdk.org, 2025, [Accessed: 2025-04-08].

[13] Høiland-Jørgensen *et al.*, "The express data path: fast programmable packet processing in the operating system kernel," in *Proceedings of the 14th International Conference on Emerging Networking EXperiments and Technologies*, ser. CoNEXT '18, 2018.

[14] Y. Li *et al.*, "Accelerating distributed reinforcement learning with in-switch computing," in *Proceedings of the 46th International Symposium on Computer Architecture*, 2019.

[15] R. Parizotto *et al.*, "Offloading machine learning to programmable data planes: A systematic survey," *ACM Comput. Surv.*, 2023.

[16] D. Arp *et al.*, "Dos and don'ts of machine learning in computer security," in *31st USENIX Security Symposium (USENIX Security 22)*, 2022.

[17] A. D'Amour *et al.*, "Underspecification presents challenges for credibility in modern machine learning," *J. Mach. Learn. Res.*, 2022.

[18] R. J. Babaria *et al.*, "Fastflow: Early yet robust network flow classification using the minimal number of time-series packets," *arXiv preprint arXiv:2504.02174*, 2025.

[19] F. Bronzino *et al.*, "Traffic refinery: Cost-aware data representation for machine learning on network traffic," *Proc. ACM Meas. Anal. Comput. Syst.*, 2021.

[20] Y. Gong *et al.*, "A survey on dataset quality in machine learning," *Information and Software Technology*, 2023.

[21] Z. Abedjan *et al.*, "Detecting data errors: where are we and what needs to be done?" *Proc. VLDB Endow.*, 2016.

[22] L. Mauri *et al.*, "Estimating degradation of machine learning data assets," *J. Data and Information Quality*, 2021.

[23] D. Foroni *et al.*, "Estimating the extent of the effects of data quality through observations," in *IEEE 37th International Conference on Data Engineering (ICDE)*, 2021.

[24] J. Cavitt *et al.*, "Detecting cyber attacks with packet loss resilience for power systems," *Sustainable Computing: Informatics and Systems*, 2022.

[25] C. Yang *et al.*, "Anti-packet-loss encrypted traffic classification via masked autoencoder," in *Wireless Artificial Intelligent Computing Systems and Applications*, 2025.

[26] J. Holland *et al.*, "New directions in automated traffic analysis," in *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*. Association for Computing Machinery, 2021.

[27] S. Sengupta *et al.*, "Exploiting diversity in android tls implementations for mobile app traffic classification," in *The World Wide Web Conference*. Association for Computing Machinery, 2019.

[28] L. Breiman, "Random forests," *Mach. Learn.*, 2001.

[29] N. Erickson *et al.*, "Autogluon-tabular: Robust and accurate automl for structured data," 2020.

[30] E. N. Gilbert, "Capacity of a burst-noise channel," *The Bell System Technical Journal*, 1960.

[31] E. O. Elliott, "Estimates of error rates for codes on burst-noise channels," *The Bell System Technical Journal*, 1963.