FastFlow: Early Yet Robust Network Flow Classification using the Minimal Number of Time-Series Packets

RUSHI JAYESHKUMAR BABARIA, University of New South Wales, Australia
MINZHAO LYU, University of New South Wales, Australia
GUSTAVO BATISTA, University of New South Wales, Australia
VIJAY SIVARAMAN, University of New South Wales, Australia

Network traffic classification is of great importance for network operators in their daily routines, such as analyzing the usage patterns of multimedia applications and optimizing network configurations. Internet service providers (ISPs) that operate high-speed links expect network flow classifiers to accurately classify flows early, using the minimal number of necessary initial packets per flow. These classifiers must also be robust to packet sequence disorders (drops and retransmissions) in candidate flows and capable of detecting unseen flow types that are not within the existing classification scope, which are not well achieved by existing methods. In this paper, we develop *FastFlow*, a time-series flow classification method that accurately classifies network flows as one of the known types or the unknown type, which dynamically selects the minimal number of packets to balance accuracy and efficiency. Toward the objectives, we first develop a flow representation process that converts packet streams at both per-packet and per-slot granularity for precise packet statistics with robustness to packet sequence disorders. Second, we develop a sequential decision-based classification model that leverages LSTM architecture trained with reinforcement learning. Our model makes dynamic decisions on the minimal number of time-series data points per flow for the confident classification as one of the known flow types or an unknown one. We evaluated our method on public datasets and demonstrated its superior performance in early and accurate flow classification. Deployment insights on the classification of over 22.9 million flows across seven application types and 33 content providers in a campus network over one week are discussed, showing that *FastFlow* requires an average of only 8.37 packets and 0.5 seconds to classify the application type of a flow with over 91% accuracy and over 96% accuracy for the content providers.

CCS Concepts: • **Networks** → **Network measurement**; **Network monitoring**; • **Computing methodologies** → **Machine learning approaches**.

Additional Key Words and Phrases: network traffic analysis; flow classification; reinforcement learning

ACM Reference Format:
Rushi Jayeshkumar Babaria, Minzhao Lyu, Gustavo Batista, and Vijay Sivaraman. 2025. *FastFlow*: Early Yet Robust Network Flow Classification using the Minimal Number of Time-Series Packets. *Proc. ACM Meas. Anal. Comput. Syst.* 9, 2, Article 23 (June 2025), 27 pages. https://doi.org/10.1145/3727115

1 Introduction

Network operators that offer Internet access services to broadband and mobile clients, or manage network infrastructure for large enterprises and campuses, are eager to gain visibility into the bandwidth demand by different applications, such as video streaming, online gaming, and

Corresponding author: Minzhao Lyu (minzhao.lyu@unsw.edu.au).
Authors' Contact Information: Rushi Jayeshkumar Babaria, University of New South Wales, Sydney, NSW, Australia; Minzhao Lyu, University of New South Wales, Sydney, NSW, Australia; Gustavo Batista, University of New South Wales, Sydney, NSW, Australia; Vijay Sivaraman, University of New South Wales, Sydney, NSW, Australia.

ACM 2476-1249/2025/6-ART23
https://doi.org/10.1145/3727115

conferencing, and their service providers (*e.g.,* YouTube and Netflix for video streaming), along with the associated experience received by the end-users, for the following major business and operational reasons. First, knowing which network services and providers are becoming popular in their deployment geography can directly impact the business planning of a network operator, such as creating selective bundles and premium subscription plans that have the potential to increase its profit margin. Second, having visibility into the experience associated with each application session facilitates proactive network troubleshooting to increase customer satisfaction and reduce potential revenue loss. Third, providing contextual labels for network flows through a network can serve as signals for dynamic performance optimization techniques, such as prioritizing network flows of latency-sensitive applications like video conferencing via low-latency 5G slices, and using network APIs for video streaming flows to guarantee sufficient bandwidth allocations.

Internet service providers (ISPs) deploy classifiers that categorize network flows into their application types for usage accounting and subsequent user experience measurement [12, 13, 30, 33, 53]. For practical deployment at large networks that can serve hundreds of thousands of users using diversified applications supported by millions of concurrent network flows, flow classifiers are expected to not only provide accurate flow labels that are of interest to network operators but also produce classification results for each flow at the earliest possible time, ideally by just seeing the initially arrived packets of a flow. This approach spares computing resources to classify more flows in the queue and quickly enables subsequent monitoring tasks (*e.g.,* inference on application user experience [12, 33]) that take the flow classification results as a prerequisite.

Network traffic classifiers have started shifting from rule-based mechanisms that match flow metadata, such as port numbers and server name indication (SNI) fields in TLS handshake payloads, which have become less effective due to increasingly encrypted application traffic and the use of randomized or non-standardized port numbers. Instead, statistical models are now being used to classify network flows based on their time-series volumetric profiles, which are determined by the actual content carried per application and are agnostic to metadata obfuscation and encryption. However, for a large network like an ISP that can have millions of concurrent flows to be quickly classified for downstream tasks such as usage accounting and performance measurement in real-time, legacy metadata-based approaches can classify a candidate flow by its first or certain signaling packet(s), while time-series statistical models often require a lengthy time-series input (*e.g.,* packets) before concluding a confident flow classification result.

Therefore, to enable the practical deployment of time-series statistical flow classifiers in large-scale networks, prior research works [21, 28, 29, 31, 33, 38–41] have developed such models using a fixed number of initial packets in a flow that carry static signatures in initialization requests pertinent to a certain application (*e.g.,* video streaming or conferencing) or content provider. However, given the diversified flow types in an ISP network, each with its static initial content carried by a different number of initial packets, having this number fixed for every flow cannot always produce reliable classification. Specifically, with a fixed number, unpredictable packets carrying dynamic user content may be included for flows that inherently require a smaller number of initial packets for classification, and initial packets carrying static flow content may not be sufficiently captured for flows that require a larger number. Moreover, packet drops and retransmissions, which are common in network communications, further render a fixed number of initial packets for classifying diversified flows in realistic network environments ineffective.

To address this gap in network flow classification for large-scale networks, a run-time optimization is required to determine the minimal number of packets to balance prediction accuracy and system efficiency [48]. An insufficient number of time-series inputs can cause inaccurate results, whereas a number more than the just enough value can lead to delays in classification.

Such optimization objectives are formulated as a sequential decision-making problem by the machine learning community and approached by reinforcement learning techniques [25, 46]. Those provide inspiration for us to develop a time-series network flow classifier with sequential decision-making capability trained by reinforcement learning, which can determine the minimal number of time-series data points at run-time to confidently classify each candidate flow.

In this paper, we develop ***FastFlow***, a time-series flow classification method to accurately classify network flows for their application types and content providers using the estimated minimal number of packets carrying initial static content pertinent to their types. The minimal number is dynamically estimated for each candidate flow at run-time. As will be overviewed in §3, our method ***FastFlow*** represents raw packet streams of a flow as well-curated data sequences at dual granularity of packets and slots (*i.e.,* time intervals), and uses purposely designed time-series classifiers leveraging long-short-term memory (LSTM) architecture tuned by reinforcement learning techniques for run-time estimation on the minimal number of time-series inputs for classifying each flow. The method is inherently capable of detecting unknown flows as outliers instead of mislabeling them as one of the known types, and is robust to packet sequence disorders within a flow due to packet drops and retransmissions, which are common cases during deployment at large networks. Our contributions in this paper are three-fold.

Our **first contribution** (§4) defines a **dual-grained time-series flow data sequence** to represent volumetric statistics of a network flow as formal inputs to our time-series flow classifiers. The fine-grained flow data sequence is curated for precise packet statistics of a candidate flow covering not only packet sizes but also contextual information including directions and inter-arrival times which serve as strong indicators for flow types when the packet sequence perfectly matches its expected norm. It is complimented by a coarse-grained time-series data sequence that consists of volumetric statistics of a flow per slot, which exhibits better statistical robustness to packet sequence disorders caused by packet drops and retransmissions through interval-based aggregation. The joint use of both types of time-series data sequence via a run-time selection process provides quality representation of a flow with both precision and robustness to sequence disorder of its arriving packets.

In the **second contribution** (§5), we develop a **time-series flow classifier architecture** that leverages long-short term memory cells (LSTM) to classify network flows on the dual-grained time-series flow data sequence describing statistics of packets arrived at runtime. The classifier is trained by reinforcement learning techniques as a sequential decision making model with inference functions (*i.e.,* linear layers) flexible on prediction timestamps and an intermediate 'unknown' flow type. The trained classifiers can dynamically estimate the minimal number of time-series data points (*e.g.,* packets) to confidently classify each candidate flow. To enhance the accuracy of our trained classifiers in detecting unknown flow types that have their statistical characteristics deviating from the known types, we develop a training data augmentation technique that compensates the commonly absent unknown flow characteristics to improve the convergence of statistical boundaries in the trained models for classifying known flow types.

The **third contribution** (§6) evaluates the classification performance of *FastFlow*. We start by comparing *FastFlow* with **state-of-the-arts methods and ablation alternatives** on three popular public network flow datasets from university and ISP labs, showcasing the performance of *FastFlow* in both simple (over 98% accuracy using on average 4 initial packets in 0.03 seconds per flow) and complex (over 86% accuracy using on average 13 initial packets in 0.6 seconds per flow) classification tasks. We then deploy *FastFlow* in a large campus network. Using flow labels from a commercial network traffic classification system as estimated ground truths, we demonstrate **deployment insights in the realistic network environment** including flow classification performance of seven popular application types and 33 content providers. On average, only 8.37 initial packets

of a flow are needed to classify its application type with an accuracy higher than 91%. Among application types, conferencing flows are detected with the highest accuracy of 99.82% using 3 to 17 initial packets in approximately 1.34 seconds. *FastFlow* also achieves superior performance in classifying content providers of each application type, an accuracy of 97% is achieved to classify video streaming providers using the first 3 to 6 packets of each flow.

2 Related Work

Classifying encrypted network flows using time-series data for application classification [1, 59] and network anomaly detection [4, 10] has been a popular topic given its importance for network operation. Despite the large number of prior works that develop or fine-tune classification models [5, 37, 42], many existing works focus on cost-effectively representing time-series flow data and developing methods for early classification of network flows *i.e.,* using the least number of time-series data points for accurate prediction, which are closely related to our work.

Time-Series Flow Data Representation: To represent a network flow that consists of a series of upstream and downstream packets arrived at different timestamps, some prior works [2, 3, 9, 11, 15, 24, 32, 51, 51, 58, 59, 61, 62, 62, 64, 65] use aggregated metrics such as packet inter-arrival time and packet length for a group of packets (such as by a fixed time interval or for a certain number of packets), which lose fine-grained packet-level statistics for accurate classification. To address this issue, other research works [14, 20, 22, 27, 45, 49, 50] choose to preserve a sequence at per-packet granularity by capturing inter-arrival time and packet size of each packet without aggregation. The classification models trained on such precise sequences can perform poorly in operational networks as packets in a flow do not always come in precise sequence due to packet drops and retransmission. This problem is especially severe for works [21, 38, 39] that classify a flow by its first few packets. In our work, we propose a flow representation process that collectively uses per-packet representation for accurate inference in general cases and time-series metrics aggregated per slot (*i.e.,* time interval) that provide resilience to candidate flows with disordered packet sequence due to packet drops and retransmission.

Time-Series Early Classification of Network Flow: A cost-effective classification of network flows in high-speed telecommunications networks often requires a decision to be taken after inspecting a small number of packets. To balance flow classification accuracy and cost, prior works use a unified number of packets or time intervals for all flows in their classification methodologies [3, 24, 38, 39, 41, 45]. For example, *GGFast* [39] uses the first 50 packets to classify each flow and *Flowpic* [45] specifies a constant 10-second time interval limit. In addition, the works in [3, 8, 17, 21, 26, 40] evaluate the performance of their flow classifiers with various numbers of packets or time intervals. Notably, in [52], the authors classify flows using their first several initial packets. However, this number of packets is a hyper parameter optimized on each training dataset, *i.e.,* deployment network, rather than on a per flow level. While such fine-tuned static number of inputs for a certain classification objective (*e.g.,* streaming video providers) in one deployment environment can perform well, such methods can be difficult to generalize for changing needs of flow classification objectives (*e.g.,* including new application types such as metaverse VR applications [29] or new content providers). Such variations in flow classification objectives are common in network operation, thus, having a static number of data points for all flows may not always provide optimal performance. In addition, the number of input data required to classify each flow can also be impacted by dropped or retransmitted packets in a practical environment. To address the limitation, our flow classification dynamically estimates the minimal number of input packets or time intervals for each candidate flow by equipping the time-series classifiers with a sequential decision-making capability trained by reinforcement learning, which is the first of such attempts (to the best of our knowledge) in network traffic classification.

Table 1. Qualitative comparison between *FastFlow* and the state-of-the-art in terms of the three key requirements of time-series flow classification for deployment in large networks.

Method	Number of time-series data points	Unknown flow type	Packet sequence disorder
[39]	a fixed input size in the method	able to detect	**not** robust
[60]	a unified input size **per TCP/UDP**	able to detect	**not** robust
[20]	a fixed input size in the method	**not** able to detect	**not** robust
[64]	a unified size **per network**	able to detect	**not** robust
[24]	a fixed input size in the method	able to detect	robust
[3]	a unified size **per network**	**not** able to detect	**not** robust
[21]	a unified size **per application type**	**not** able to detect	**not** robust
[38]	a unified size **per network**	**not** able to detect	**not** robust
[63]	a unified size **per network**	able to detect	robust
FastFlow	an estimated minimal size **per candidate flow**	able to detect	robust

3 Overview of the *FastFlow* Time-Series Flow Classification Method

This section discusses key requirements for flow classifications in large networks that are not fully addressed by state-of-the-arts (§3.1), which motivate our *FastFlow* method overviewed in §3.2.

3.1 Key Requirements for Time-Series Flow Classification in Large Networks

Network operators expect that their flow classifiers have decent prediction performance and low system costs [48]. Specifically, three major requirements can determine the practicality of flow classification methods deployed in large networks with massive throughput scaling to millions of concurrent flows: (i) early classification with the estimated minimal length of time-series packet statistics, (ii) detecting unknown flow types, and (iii) robustness to packet sequence disorders. Table 1 provides a comparison of state-of-the-art methods with respect to these three requirements.

First, operators seek to reduce computational resource consumption in processing packets that can be amplified in magnitude when flow classifiers are deployed for millions of concurrent flows in a large network. They also expect each flow to be quickly classified for precise post-prediction telemetry. Therefore, early classification that aims to provide a prediction for each flow as early as possible while maintaining classification accuracy is necessitated. This requires the estimation of an **minimal number of time-series data points** (*e.g.,* packets or slot statistics) for classifying each candidate flow. An unnecessarily large input size may reduce classification speed, whereas an insufficiently small number can lead to inaccurate classification results. State-of-the-art methods either use a fixed input size, such as first 10 packets [3], a unified input size for each deployment network (*e.g.,* [39]) or per application type regardless of the possible variations and complexities in flow profiles during practical deployment.

Second, for a flow classifier trained on a labeled dataset of known flow types such as applications (*e.g.,* video streaming or conferencing) and content providers (*e.g.,* YouTube or Zoom), processing flows that do not belong to any known types is unavoidable in practical deployments. Although classifying flows into a finite coarse-grained scope can bypass unknown flow types, such as annotating flows by their network protocols and port numbers [28], the value of classification results diminishes for large network operators that require visibility into trending applications and popular providers for network optimization and business strategies. Therefore, flow classifiers are expected to produce fine-grained application or provider-level classification while being capable of **detecting unknown flow types** rather than mislabeling them as one of the pre-defined known types. This is known as out-of-distribution classification in the machine learning community for scenarios in which it is unfeasible to enumerate all possible types. However, as listed in the third column of Table 1, such capability does not always exist due to their classifier architectures.

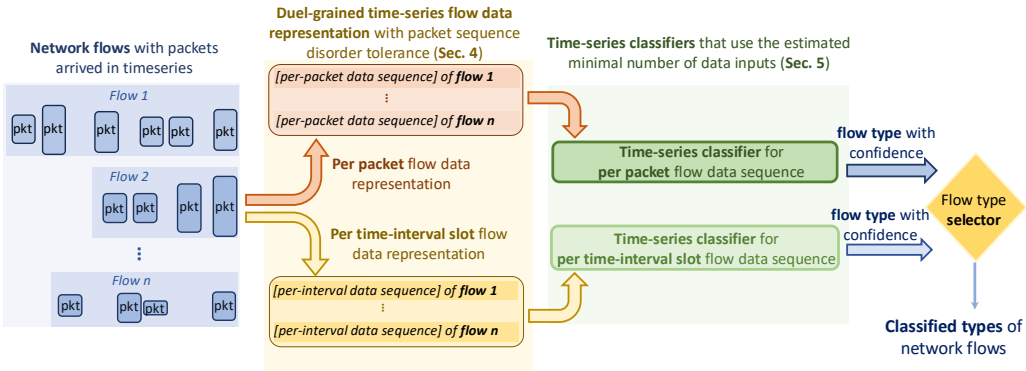


Fig. 1. Overview of our *FastFlow* flow classification process for large networks.

Third, packet sequence disorders in a flow caused by packet drops and retransmissions are commonly observed in large networks, where network conditions may not be ideal for every flow, such as the client devices served by lossy wireless connections, network congestion or bottlenecks on the routing path, or overwhelmed servers dropping packets. Therefore, flow classifiers are expected to be reasonably **robust** to deviated time-series flow patterns caused by **packet sequence disorders**. As shown in the last column of Table 1 and will be experimentally evaluated in §6, prior works that precisely match a fixed number of packets for flow classification are not inherently robust to packet sequence disorders. Some prior works are arguably robust as they use aggregated statistics over a relatively lengthy interval (*e.g.,* per 40.96 seconds [24] or the entire duration of a flow [63]), however, this inevitably sacrifices the speed of real-time flow classification.

3.2 Overview of the *FastFlow* Flow Classification Method

Aimed to develop a flow classification method that addresses the three requirements for deployment in large networks, we leverage a combination of fine-grained packet sequence with coarse-grained slot sequence to represent time-series statistics per candidate flow. Our approach also avails of time-series classifiers trained with reinforcement learning for dynamic estimations on the minimal number of data points to take before making a confident classification per flow. An overview of our method is provided in Fig. 1, showing the steps starting with a flow represented as time-series data sequence until the final classification with confidence score. Our method aims to classify the real-time flows carried by a large network into known flow types (*i.e.,* types labeled during the training process) or the unknown flow type (*i.e.,* flow types not present in the training data) that are not of interest to a network operator or need further analysis.

Starting from the left side of Fig. 1, all candidate flows that continuously have their packets flowing through the network will be represented as a time-series sequence containing statistics of the individual packets (*i.e.,* packet level) and aggregated per a given slot (*i.e.,* time interval). As will be detailed in §4, this design takes advantage of both the precise flow statistics provided by packet sequence under ideal network conditions and the robustness to packet sequence disorders introduced by interval-based aggregation. As shown in the green modules in Fig. 1 and discussed in §5.3, time-series flow data from each type of representation are fed into their respective time-series classifiers. In contrast to prior works in flow classification, each classifier in our design is trained with reinforcement learning techniques instead of supervised learning so that they are capable of making confident classification when sufficient (least number of) time-series inputs are provided for each candidate flow and identifying flows that do not belong to the known types. Two different classifiers on packet and time interval real-time sequence data can produce unsynchronized results.

For example, a flow with packet drops may not give a confident classification result until a large number of packets are received for distinguishable sequential patterns. In contrast, the data sequence aggregated by time interval can give fast prediction results with high confidence. To produce a fast and robust classification result per flow, a high-confident classification result from either one of the classifiers will first be selected.

4 Precise yet Packet Sequence Disorder Robust Time-Series Flow Data Representation

In this section, we discuss our dual-grained time-series data representation of a network flow that precisely preserve the fine-grained data sequence at (i) packet-level (§4.1) for accurate classification without the occurrence of packet sequence disorders, assisted with (ii) coarse-grained data sequence aggregated per slot (*i.e.,* time interval) that is robust to the impact of packet sequence disorders on time-series classification performance (§4.2). The classification result for a candidate flow at both granularities is selected by jointly considering their decision time and confidence §4.3 for the earliest possible yet reliable decision.

4.1 Time-Series Flow Data Sequence at Packet Granularity

Under good conditions, network flows are expected to have lossless packet behaviors without disorder in their packet sequences caused by packet drops and retransmissions. As discussed by prior works [20, 21, 38, 39], network flows of a certain type, such as application types like video streaming or online gaming, application titles like different online games, or different user device operating systems (*e.g.,* iOS and Android), often exhibit unique sequence in their packet directions, sizes, and inter-arrival times. As revealed by prior works [29, 31, 33], this is particularly true for their initial packets that carry static content (*e.g.,* requested services or application metadata) instead of those following ones depending on user's actions.

Therefore, to precisely preserve the packet-level statistics, we define our **fine-grained** time-series packet representation of a candidate flow as:

$$\boldsymbol{p} = [\overrightarrow{p_1}, \overrightarrow{p_2}, .., \overrightarrow{p_n}] \tag{1}$$

where as $\overrightarrow{p_n}$ describes statistics of the nth packet in this flow, including packet direction (dir_n), packet size (s_n), and inter-arrival time (Δt_n), which can be expressed as:

$$\overrightarrow{p_n} = \begin{pmatrix} dir_n & s_n & \Delta t_n \end{pmatrix}^T \tag{2}$$

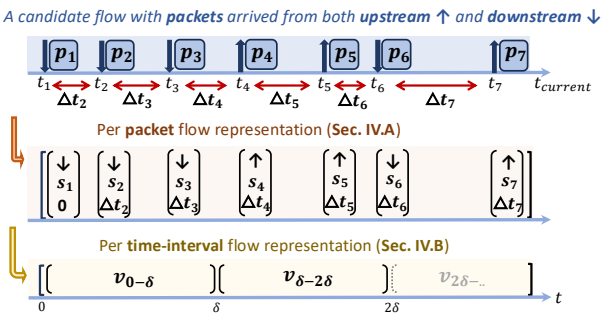


Fig. 2. A candidate flow being represented as per packet and per slot time-series data sequence.

A visual example of converting an ongoing flow into our per-packet time-series flow representation is provided in Fig. 2. From the middle layer of the figure, we can see that the statistics of every packet that has arrived until the current timestamp have been included in a runtime array for this candidate flow. Our fine-grained time-series flow representation with packet attributes in matrix format is well compatible with popular time-series classification models such as Long-Short Term Memory Cell (LSTM) without extra overhead for pre-processing. Considering computational efficiency, in our later implementation, the packet direction dir_n is set to 1 for upstream or 0 for downstream, the packet size s_n is calculated as the packet payload size excluding the network and transport layer headers in the

unit of MTU (*i.e.,* $s_n = payload_n/MTU$), and the packet inter-arrival time is taken as its log-scale conversion.

4.2 Time-Series Flow Data Sequence at Slot Granularity

Our per packet sequence $\boldsymbol{p}$ as defined in Equation 1 works well for time-series flow classification when the packet-level profiles precisely match their expected patterns learned during the training process. However, this nearly ideal assumption might not always be realistic in deployment when the expected packet sequence of a candidate flow can be disordered due to packet drops and retransmissions caused by practical factors such as network, client, and server conditions. Therefore, we introduce our **coarse-grained** time-series representation for a candidate flow with each data point aggregated on slots, which can better handle packet sequence disorders caused by less ideal network conditions by trading off visibility into each packet.

We acknowledge that prior works [3, 26] have also used aggregation per a certain number of packets instead of time intervals, as their classifications mainly use time-independent volumetric profiles such as average packet sizes. In early classification tasks, temporal information of a flow, such as the burstiness of the arrived packets, is equally important [24, 41]. Therefore, we aggregate packets that fall into each time-interval slot by their arrival times to preserve both volumetric and temporal information.

Assuming that the first packet of a flow arrived at the timestamp 0, and we have a time-interval slot of interest δ for aggregation, our slot time-series representation of a flow can be defined as:

$$\boldsymbol{v} = [\overrightarrow{v_{0-\delta}}, \overrightarrow{v_{\delta-2\delta}}, \ldots, \overrightarrow{v_{(n-1)\delta-n\delta}}] \tag{3}$$

where $\overrightarrow{v_{(n-1)\delta-n\delta}}$ represents the statistics aggregated from the packets that arrived between the timestamps of $(n-1)\delta$ and $n\delta$.

Aggregating packet statistics per time-interval slot: We now describe our methodology for obtaining the aggregation data point $\overrightarrow{v_{(n-1)\delta-n\delta}}$. For simplicity, we use $\overrightarrow{v}$ for this data point, omitting its index in the rest of this section. We aim to explicitly preserve important contextual characteristics of a network flow that are inherently impacted by its functions and types [28, 29, 33, 53], including packet directions, sizes of light and heavy packets, and dominant data transmission direction. Also, the representation process should also be lightweight in real time, *i.e.,* can be achieved by online algorithms. We therefore define the aggregated data point $\overrightarrow{v}$ in Equation 4:

$$\overrightarrow{v} = \left(\overline{\mathbf{s}_{h\uparrow}} \quad \overline{\mathbf{s}_{h\downarrow}} \quad \overline{\mathbf{s}_{l\uparrow}} \quad \overline{\mathbf{s}_{l\downarrow}} \quad \frac{\sum \mathbf{s}_{\uparrow}}{\sum \mathbf{s}_{\downarrow}}\right)^T \tag{4}$$

where the first five items $\overline{\mathbf{s}_{h\uparrow}}$, $\overline{\mathbf{s}_{h\downarrow}}$, $\overline{\mathbf{s}_{l\uparrow}}$, $\overline{\mathbf{s}_{l\downarrow}}$ and $\frac{\sum \mathbf{s}_{\uparrow}}{\sum \mathbf{s}_{\downarrow}}$ denote the average packet payload size for upstream heavy packets, downstream heavy packets, upstream light packets, downstream light packets, and ratio between total upstream and downstream packet sizes per time-interval slot δ, respectively. We use the average function for packet payload sizes as it is statistically important and can be computed using online algorithms in real-time. The light and heavy packet payload sizes are decided by a threshold value that can be set empirically for each deployment network. In our later implementation, we chose a threshold value of 1200 bytes to group packet payloads into light and heavy categories, which have been demonstrated as a reasonable separation.

4.3 Selecting Flow Classification Results at Real-Time

In our flow classification process provided in Fig. 1, time-series data sequence of a candidate flow at the granularities of both packet (§4.1) and slot (§4.2) are fed into their respective classifiers, each generates a classification result (*i.e.,* flow type) with classification confidence. Our classifiers (to be discussed in §5.1) that work on packet or slot time-series data sequences can process their confident results for one candidate flow **asynchronously**. For example, a flow without packet sequence

Algorithm 1: Selecting flow classification results produced by packet and slot classifiers nearly synchronously or asynchronously.

(1) ***Input***: $(\boldsymbol{class_p}, \boldsymbol{conf_p})$ ← classification results with confidence scores on packet flow data sequence, $(\boldsymbol{class_t}, \boldsymbol{conf_t})$ ← results on slot flow data sequence, (T_p, T_t) ← confidence thresholds for packet and slot flow data sequence, $\Delta select$ ← user-defined selection time window;
(2) ***Output***: $(class_s, conf_s)$ ← the selected flow classification result and its confidence score;
(3) ***Real-time flow classification result selection process:***
// asynchronous event: a new result from the packet sequence classifier
(5) **if** *an arrived* $(class_p, conf_p)$ ***and*** *no* $(class_t, conf_t)$ *arrive within* $\Delta select$ **then**
(6) | **if** $conf_p > T_p$ **then**
(7) | | ***return*** $(class_s, conf_s) \leftarrow (class_p, conf_p)$ // confident flow type selected and exit
(8) | **else**
(9) | | ***wait*** *for another data point to arrive*
(10) | **end**
(11) **end**
// asynchronous event: a new result from the slot sequence classifier
(13) **if** *an arrived* $(class_t, conf_t)$ ***and*** *no* $(class_p, conf_p)$ *arrive within* $\Delta select$ **then**
(14) | **if** $conf_t > T_t$ **then**
(15) | | ***return*** $(class_s, conf_s) \leftarrow (class_t, conf_t)$ // confident flow type selected and exit
(16) | **else**
(17) | | ***wait*** *for another result to arrive*
(18) | **end**
(19) **end**
// synchronous event: new results from both classifiers
(21) **if** *a* $(class_p, conf_p)$ ***and*** *a* $(class_t, conf_t)$ *arrive within* $\Delta select$ **then**
(22) | **if** $conf_p > conf_t$ ***and*** $conf_p > T_p$ **then**
(23) | | ***return*** $(class_s, conf_s) \leftarrow (class_p, conf_p)$ // flow type by packet sequence selected and exit
(24) | **else**
(25) | | ***wait*** *for another result to arrive*
(26) | **end**
(27) | **if** $conf_t > conf_p$ ***and*** $conf_t > T_t$ **then**
(28) | | ***return*** $(class_s, conf_s) \leftarrow (class_t, conf_t)$ // flow type by slot sequence selected and exit
(29) | **else**
(30) | | ***wait*** *for another result to arrive*
(31) | **end**
(32) **end**

disorders can be confidently classified by the packet sequence classifier when the fifth packet arrives at a 0.5-second timestamp. In contrast, the slot classifier may produce its results at a 3-second timestamp. Alternatively, a flow with packet sequence disorders may never be confidently classified by its packet sequence classifier, while the slot classifier may generate a confident result within several seconds. In addition, both classifiers can produce their confident results without noticeable time differences, *i.e.,* **synchronously**. Therefore, to select the most accurate classification label for a candidate flow at the earliest possible timestamp, we design a real-time algorithm (algorithm 1) to select the confident classification result for a candidate flow from its labels produced by packet and slot classifiers asynchronously or synchronously.

The algorithm takes flow classification results and their confidence scores that are generated in real-time from both packet and slot sequence classifiers. As will be discussed soon in §5.1, the packet sequence classifier generates one result when a new packet arrives, and the slot classifier generates one prediction per time interval. Those results generated at runtime are continuously checked against preset confidence thresholds T_p and T_t to select a highly confident flow type. In our implementation, we use 90th-percentile of all confidence values obtained during the training

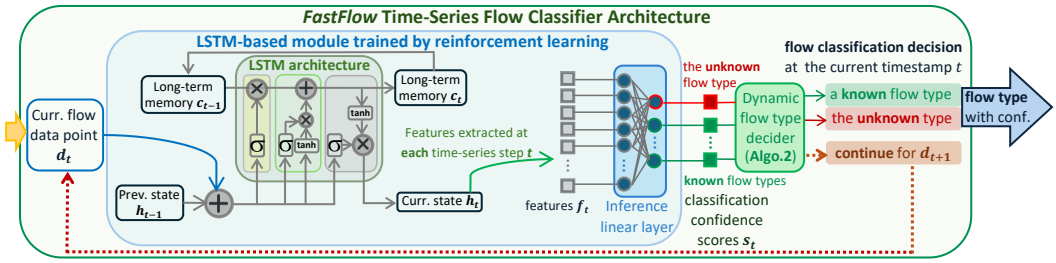


Fig. 3. The architecture of *FastFlow* time-series flow classifier realizing a sequential decision-making process.

process. We also provide a practical selection time window $\Delta select$ to recognize synchronously and asynchronously generated results by both classifiers.

For the asynchronous event, *i.e.,* a flow classification result is generated by one classifier, and no result is generated by the other one within $\Delta select$, as specified by blocks **(1)** and **(2)** in Algorithm 1, the classification confidence is checked with the respective threshold for the decision to accept the predicted flow type or wait for more results to come. For the synchronous event processed by block **(3)**, the results of both classifiers are first compared with each other before checking with their respective confidence thresholds for a decision. In our engineering implementation, a bonus confidence score will be added to the classification result if both classifiers predict the same flow type.

5 Time-Series Flow Classifier with the Estimated Minimal Length of Data Sequence

After discussing how each network flow is represented as a time-series data sequence at both packet level (for precision) and slot level (for robustness to packet disorder), we now present our time-series (LSTM-based) classifiers (§5.1) that dynamically estimate the minimal length of data sequence required for accurate classification. Our LSTM classifiers are fitted with a reinforcement learning technique (§5.2) on augmented labeled training datasets (§5.3).

5.1 Time-Series Classifier Architecture for Early Flow Classification

Our approach uses time-series classifiers (green modules in Fig. 1) to predict each flow type by its per packet or slot data sequences. After deployment, these classifiers are expected to process a large number of concurrent flows timely and accurately. An approach to increase the efficiency of the classifiers is to provide early predictions using a small number of time-series inputs (*i.e.,* packets or slots) for each flow. There is no single optimal number of time-series inputs for all flows, and a one-fits-all approach will not provide optimal results in practice.

However, existing works in flow classification typically train their models with a standard supervised approach that requires a fixed number of timesteps [41], [45], [49]. The number of timesteps is considered a model hyperparameter optimized during training. Therefore, these methods will process the same amount of input for every flow and cannot adapt to the requirements of each candidate flow. Since the optimal time-series input length required for each flow may vary according to the classification objective and minor variations in the data sequence caused by network conditions, these approaches leave room for optimization. Moreover, traditional classifiers make a "closed-world" assumption that, after deployment, they will only observe the class labels present in the training set. However, this is an unrealistic assumption given the number and diversity of flows in operational networks. Consequently, these classifiers are doomed to misclassify unknown flows, unseen in the training data, as one of the known flow types [60].

Algorithm 2: Dynamic inference on time-series flow data sequence.

***Input*:** Confidence scores of flow types $\vec{s_t}$ from the classifier, after processing the t^{th} data point in the flow data sequence; confidence threshold T_{unk}; and the maximum time steps to make prediction C_{unk}.
***Output*:** Predicted flow type l and the prediction confidence p. $\vec{c_t} \leftarrow softmax(\vec{s_t})$
if $t = C_{unk}$ **then**
    // End flow classification as the maximum number of time step C_{unk} has been reached.
    $l \leftarrow$ 'unknown';
    $p \leftarrow \vec{c_t}$['unknown']
    return l, p
end
if $\vec{c_t}$['unknown'] $\geq T_{unk}$ *or* $argmax(\vec{c_t})$ = 'unknown' **then**
    // Not yet confident enough to make a 'not unknown' prediction, wait for $\vec{s}_{t+1}$
    $l \leftarrow$ 'unknown'
    $p \leftarrow \vec{c_t}$['unknown']
    continue for the next time step $t + 1$.
else
    // Confident enough to make a 'not unknown' prediction
    $l \leftarrow argmax(\vec{c_t})$
    $p \leftarrow \vec{c_t}[l]$
    return l, p
end

Existing works on flow classification using time-series models (especially deep-learning based) are prone to misclassify unknown flow types into known types [7]. Even existing approaches that can address the open-world assumption often come with additional requirements such as additional data [64], extra computing power [60] or constraints on data distribution [24]. Our approach addresses both requirements with a uniquely elegant solution: a *sequential decision maker*. Under this paradigm, a classifier can take one of two decisions after receiving a new time-series data point: output a prediction or wait for the next data point. This decision is based on the classifier's confidence until the current data point. This approach has two major benefits: (i) the classifier issues a prediction as soon as it has gained enough confidence, and (ii) if no prediction is issued after enough data points, the classifier can issue an *unknown* class label output.

Specifically, Fig. 3 illustrates the architecture of our classifier. The blue rectangle indicates the section of the model trained with reinforcement learning. An LSTM cell provides a feature set at each time step. These features are fed into a fully connected network that outputs a class or an unknown label. The next section provides details on the training process for our trainable classifier module, which uses reinforcement learning instead of supervised learning approaches.

Our classifier leverages an LSTM [18] architecture due to its ability to process sequential data. Traditional LSTM-based classifiers extract features from a fixed number of timesteps and feed these features to a softmax layer to obtain scores for each known class label. As shown in the blue region of Fig. 3, our classifier module has two key design choices that differ from prior works. First, instead of waiting for a fixed number of timesteps, our model extracts features from the LSTM architecture every time a new data point arrives, which are then used by an inference linear layer for the classification confidence scores of each candidate class. Second, we introduce the '*unknown*' flow type in addition to the existing types. Therefore, given a new flow, our model can output a temporary *unknown* label for the initial timesteps and wait for more data points of the same flow until a confidence classification can be concluded or the maximum number of timesteps has been reached. The run-time decision on each time step is made by the dynamic flow type decider (green box in Fig. 3) as detailed in Algo. 2.

The algorithm processes the confidence of the k known classes plus the extra unknown class for each incoming time-series data point. If the confidence of the unknown class is greater than the threshold T_{unk} or if it is the most confident class, a temporary unknown classification is decided, making the classifier continue for the next time series data point. Otherwise, the algorithm produces the most confident class among the k known classes. This process continues until an output is issued or the classifier processes a maximum of C_{unk} timesteps. We note that both thresholds, T_{unk} and C_{unk}, are hyperparameters. In our training process, we tune C_{unk} to provide a balance between prediction accuracy and speed.

5.2 Training *FastFlow* Classifiers with Reinforcement Learning

We have conceived the network flow classification task as a sequential decision problem, making traditional supervised learning methods unsuitable for training our models. Traditional supervised learning defines a loss function computed for every training case. Therefore, these methods assume that every case should immediately lead to some loss after classification. However, our design introduces a postponing decision that does not immediately lead to a loss of value.

Reinforcement learning (RL) [6] is a better-suited approach to train our models. RL involves an agent (*i.e.,* sequential decision-making classifier) that observes the current environment state and chooses to take an action from a predefined action space. Based on this action, the environment will update its current state. Each time the agent/classifier selects an action, it gets a reward based on its fit to the current state. This process repeats until the environment arrives at a terminal state, where the agent cannot change the environment's state further. RL learning aims to maximize the total reward granted to the agent from the start state to the terminal state.

There are many popular RL algorithms, such as Q-learning [55], PPO [44], and A3C [35]. We employ Q-learning since our flow classification leads to a discreet action space. Q-learning is also known to be sample efficient [23], which is crucial when collecting labeled data is expensive. Finally, Q-learning had several enhancements over the initial algorithm, such as model architecture structure [54], loss function [47], and training process [43], which have improved its performance, as our experiments confirm. Q-learning trains the agent to choose the best action from the action space given a state. Given a current state s as input, the agent outputs a number (Q value) for each action potential $a \in A$ in the action space A. The Q-value $Q(s, a)$ represents the expected reward for taking action a in state s. Therefore, after training, the agent always picks the action with the maximum Q-value given an input state.

In our flow classification approach, the agent is our classifier C. The state is the first p timesteps of the flow representation (in either packet or slot). The action space for a k-way classification problem has $k+1$ actions, one for each flow type, and the $k+1th$ action for the unknown class. The terminal state occurs when the agent chooses to make a prediction or when it has seen a maximum number of timesteps, C_{unk}. The reward given to the agent is decided by the reward function that takes the current state s, the action a, and the actual class label l of the current flow.

$$\textbf{reward\_func}(s, a, l) = \begin{cases} wait\_penalty & \text{if } a == k+1 \\ positive\_reward & \text{else if } a == l \\ negative\_reward & \text{otherwise} \end{cases} \tag{5}$$

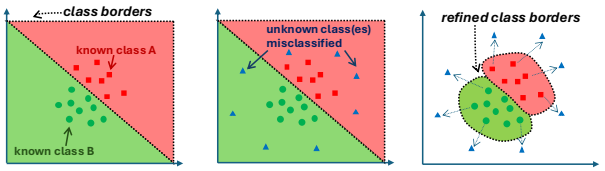
The reward function outputs a positive reward ('+1' in our training process) if the classifier outputs a correct prediction; or a negative reward ('−1' in our implementation) for an incorrect prediction. However, if the classifier decides to output 'unknown' and continues for the next timestep, a negative reward '*wait_penalty*', a hyperparameter that determines the speed of classification, will be awarded. We set the value of '*wait_penalty*' as '−.03' in our experiments after iterative selections. Lowering its value will place more importance on classification speed than accuracy and

vice versa. The model is then trained using double Q learning loss with sampling priority [43, 47] that inputs tuples of $(state, action, next\_state, reward)$.

5.3 Training Flow Data Augmentation

A final component of our solution involves the generation of flows with unknown labels. In many real scenarios, these unknown flows are readily available, consisting of all flows that do not belong to the interest classes. However, we may find some challenges in collecting these flows in practice. One situation occurs when these flows need to be manually labeled to guarantee they do not fall into one of the existing classes. Another practical situation involves training models using benchmark datasets, as most of these datasets do not bring a general unknown class.

A solution to this problem is to generate synthetic unknown flows using augmentation techniques. Augmentation is a widely used approach in deep learning to solve data deficiency [34], as several deep models have a large number of parameters that require extensive training sets. We use a strong augmentation approach that generates synthetic unknown flows away from the high-density areas that characterize each known flow type.



(a) Classifier trained on known classes. (b) Misclassifying unknown class(es). (c) Classifier with refined class borders

Fig. 4. (*a*) Under the closed-world assumption, a classifier can naively split the entire feature space according to the existing classes. (*b*) The resulting classifier misclassifies unknown instances as belonging to one of the existing classes. (*c*) We use augmentation in classifier training to generate closely outlying synthetic unknown flows that help bind the classes' decision borders.

Fig. 4 is a simplified example that illustrates using strongly augmented flows to restrain the decision border around each class, so that the trained classifier is capable of detecting flow instances belonging to the unknown types (*i.e.,* not statistically belonging to any known class) with the refined borders of each flow type.

Specifically, this synthetic unknown flow augmentation is achieved by distorting the attributes of packets before converting them into packet or slot flow data representation. We start by sampling $\alpha_{attr} \in [0, 1]$ fraction of packets in each flow. The payload size, ps_i, direction, dir_i, and timestamp, t_i for each sampled ith packet are then distorted as follows.

$$ps_i^{aug} = ps_i * \alpha_{ps} + \mathrm{U}(0, MTU) * (1 - \alpha_{ps}) \tag{6}$$

$$t_i^{aug} = t_i + ((t_{i+1} - t_i) * \mathrm{U}(0, 1) - (t_i - t_{i-1}) * \mathrm{U}(0, 1)) * \alpha_{ts} \tag{7}$$

where $\alpha_{ps} \in [0, 1]$ and $\alpha_{ts} \in [0, 1]$ are the strength of payload and timestamp distortion, respectively. $U(a, b)$ denotes the uniform distribution over the range $[a, b]$. To apply augmentation for the attribute dir_i, the direction of a flow is randomly reversed with probability α_{dir}. In practice, we set α_{ps}, α_{ts} and α_{dir} to '0.2' after experimental selections. α_{attr} is sampled from $U(0.6, 0.9)$ to generate highly augmented pseudo unknown flows. We represent this augmentation over attributes using $\mathcal{F}_{attr}$, as formally defined in formula 8.

$$\textbf{augmented\_training\_flow} = \mathcal{F}_{\text{attr}}(flow\_known, \alpha_{\text{attr}}, \alpha_{ps}, \alpha_{ts}, \alpha_{\text{dir}}) \tag{8}$$

Notably, the pseudo unknown flows used in the classifier training process will be labeled '*unknown*'. Therefore, during training, the classifier will always get a negative reward when it misclassifies these flows into one of the known types. This enhances the unknown flow detection capability of our trained classifier. We also create a weaker form of augmentation by setting α_{attr}

from $U(0, 0.2)$ to increase the size of the training data and mitigate the class imbalance [57], which are also tested as effective in our evaluation using public datasets and campus network deployment.

6 Evaluation, Benchmarking and Deployment Insights

This section thoroughly evaluates *FastFlow* for flow classification, focusing on classification performance (accuracy and execution time) and recognition of unknown flows in line with our design objectives. This section starts by describing the experimental setup used in our experiments (§6.1), followed by a comparison with the state-of-the-art (§6.2). We also provide an ablation study evaluating the necessity of our use of packet and slot flow data sequence (§6.3), and conclude this section with deployment insights in our university campus network that mimics a residential ISP (§6.4).

6.1 Lab Evaluation Setup

Our lab experiments evaluate the performance of *FastFlow* in achieving two primary objectives: *i*) classifying network flows accurately with fast speed through the use of the estimated minimal number of packets per flow and *ii*) maintaining classification robustness to packet sequence disorders and unknown flow types commonly found in deployments. To achieve this objective we use three popular public datasets containing labeled packet captures (PCAPs) of network flows for various network types, namely VNAT [24] by MIT Lincoln lab for VPN networks, UTMobileNet [19] for mobile networks, and UNIBS [16] collected from the campus network edge of the University of Brescia. Table 2 lists the flow types in each dataset.

Table 2. Specification of the three public network flow datasets used in our evaluation.

UTMobileNet		UNIBS	
Flow type	**#flows**	**Flow type**	**#flows**
Google-Maps	2584	Browsers	18820
Netflix	1680	P2P	14175
Reddit	1295	Mail	4432
Youtube	1031	Other	2368
Facebook	1028	Skype	499
Instagram	994		
Pinterest	994	**VNAT**	
Google-Drive	709	Streaming	1628
Spotify	699	Chat	811
Twitter	682	Control	611
Gmail	400	File Transfer	456
Hangout	351		
Messenger	329		

A common limitation of these public datasets is that they are collected under nearly ideal network conditions, with no packet drop or retransmission in each labeled flow. Also, they do not contain unknown (unlabelled) flows. Therefore, we augment these datasets to introduce packet sequence disorders and randomly exclude certain flow types to mimic unknown flows in our evaluation.

To simulate packet sequence disorder, we randomly drop $\alpha_{drop} \in [0, 100)$ percentage of packets in the flow. For TCP flows, these packets are reinserted if the flow uses TCP after a variable retransmission delay. In practice, we set the retransmission delay as the RTT calculated by a three-way handshake. We chose the the drop probability α_{drop} of each packet randomly following a normal distribution with a mean value of 5% and a standard deviation of 3.5%, as suggested by the network operation community [36, 56]. This models a scenario with more realistic network conditions.

To test the capability of detecting unknown flow types, we randomly exclude around 20-25% of the flows from the training set for each training and evaluation iteration. More precisely, we remove three flow types from UTMobileNet, and one flow type from UNIBS and VNAT. We perform ten evaluation iterations for each dataset. We guarantee that every flow type is excluded from the training set at least once. During each evaluation iteration, we randomly use 70% of the data for model training and the remaining 30% data for testing.

We must clarify that these augmentations were chosen to give existing datasets more realistic data characteristics. These augmentations were *not* introduced into the data to benefit *FastFlow*, besides the fact that our method was designed to deal with packet drops and unknown flows. To certify that *FastFlow* is not favored, the packet drops are *only* introduced in the test data. Thus, neither *FastFlow* nor its competitors have information about the packet drop rate during training. We also present results of the non-augmented dataset in Appendix B, so the reader can have a

Table 3. Classification performance comparison between *FastFlow*, the state-of-the-art and baselines under packet sequence disorder and unknown flow types.

Dataset	Method	Classification performance on known flow types				Unknown types	
		Macro F1 (%)	Accuracy (%)	Packets (#)	Time (s)	FPR (%)	TPR (%)
UTMobileNet	**FastFlow**	**85.42**	86.32	12.92 ± 9.48	0.57 ± 1.61	4.56	86.94
	GGFast [39]	77.82	83.90	50	2.56 ± 0.49	**1.04**	**90.42**
	Grad-BP [60]	80.33	83.85	100(TCP) 10(UDP)	4.86 ± 1.98	4.92	61.35
	Pkt.-5	65.28	74.20	5	0.28 ± 1.07	–	–
	Pkt.-45	80.02	81.17	45	2.40 ± 1.24	–	–
	Time-int.-5	73.39	75.96	**4.39 ± 1.96**	**0.25**	–	–
	Time-int.-45	85.29	**87.05**	42.02 ± 9.23	2.25	–	–
VNAT	**FastFlow**	96.24	**98.03**	**4.16 ± 1.34**	**0.033 ± 1.43**	**0**	97.32
	GGFast [39]	70.45	86.68	50	1.08 ± 0.42	**0**	**99.47**
	Grad-BP [60]	95.64	95.91	100(TCP) 10(UDP)	1.91 ± 0.56	4.98	93.07
	Pkt.-5	63.53	71.48	5	.035 ± 0.81	–	–
	Pkt.-45	80.41	85.66	45	0.96 ± 1.59	–	–
	Time-int.-5	**96.48**	97.59	14.22 ± 5.42	0.25	–	–
	Time-int.-45	95.20	97.45	100.47 ± 23.44	2.25	–	–
UNIBS	**FastFlow**	92.30	95.21	9.92 ± 3.92	0.36 ± 0.82	2.54	98.16
	GGFast [39]	87.93	91.95	50	1.52 ± 1.87	**0.87**	**99.37**
	Grad-BP [60]	90.15	93.46	100(TCP) 10(UDP)	2.95 ± 1.97	4.93	68.45
	Pkt.-5	81.49	87.02	**5**	0.19 ± 1.18	–	–
	Pkt.-45	90.53	94.45	45	1.36 ± 1.46	–	–
	Time-int.-5	81.91	82.84	6.83 ± 2.80	**0.25**	–	–
	Time-int.-45	**92.36**	**95.55**	75 ± 17.15	2.25	–	–

complete picture of *FastFlow* performance with all combinations of presence and absence of packet drops and unknown flows.

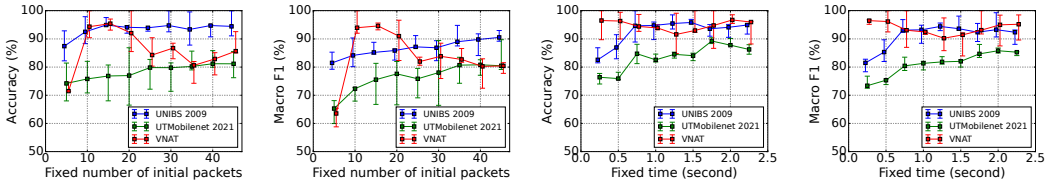
Our experiments use different performance measures. We report accuracy and macro F1 for identifying known flow types. Accuracy is a prevalent performance measure, but it is difficult to interpret in the presence of several flow types and when some are uncommon. Macro F1 is the unweighted average of the F1 measure for each flow. This measure favors classifiers that perform well for all flow types independently of the number of flows. Both measures are shown as percentages, with higher values indicating better classifiers.

We report the false positive rate (FPR) and the true positive rate (TPR) to assess the recognition of unknown flows. The FPR measures the percentage of unknown flows incorrectly assigned to one of the existing flow types. The TPR measures the percentage of unknown flows correctly recognized as such. Finally, we report the number of packets and inference time in seconds as measures of inference efficiency.

We conclude our experiment setup with some remarks about *FastFlow* hyperparameters to enable the reproducibility of our results. The hyper-parameters for the *FastFlow*' LSTM cells are set through standard tuning processes, including a hidden size of 128, Adam optimizer with a learning rate of $3e-4$, and a capped training epoch of 200. The static parameters in *FastFlow* classifier architecture are empirically tuned for a balanced classification and speed per deployment network (or dataset). For example, the time-interval slot is configured as 50ms, C_{unk} is set to 20, 30, and 15 for UNIBS, UTMobilenet and VNAT dataset, respectively. In our campus deployment, this value is set to 25 during the training process. T_{unk} and $wait\_penalty$ are consistently set to 0.8 and -0.03, respectively.

6.2 Evaluation and Comparison to the State-of-the-art

We evaluate the performance of *FastFlow* and compare it with two representative state-of-the-art network flow classification methods recently developed by the research community: *GGFast* [39] and *Grad-BP* [60] as well as some baselines. Both *GGFast* and *Grad-BP* have been practically



(a) Accuracy for using a fixed number of packets.
(b) Marco F1 for using a fixed number of packets.
(c) Accuracy for using fixed time-interval slots.
(d) Marco F1 for using fixed time-interval slots.

Fig. 5. Performance of classifiers when using time-series statistics from a fixed number of packets or time-interval slots on the three public datasets. The bounded range of each data point shows the variation of accuracy/marcoF1 across flow types in each dataset.

deployed in large networks, as discussed in their respective papers. *GGFast* classify flows by their packet length sequences and *Grad-BP* uses deep learning models to classify flow types by its packet lengths and directions in time-series sequences. The two methods are trained with their original design without our augmentation technique for unknown flow detection as discussed in §5.3. Other promising works such as CATO [48] for flow classification in large networks leveraging different approaches like multi-objective optimization, are not all extensively evaluated.

Table 3 summarizes our evaluation results[1] *FastFlow* outperforms *GGFast* and *Grad-BP* in terms of classification performance for known flow types and inference efficiency. *GGFast* performs better than *FastFlow* for detecting unknown flows. However, *GGFast*'s small decrease in FPR and increase in TPR for unknown flows comes with a significant decrease in performance for the classification of known flow types. For example, for the VNAT dataset, *GGFast* macro F1 is only 70.45% where *FastFlow* archives a respected 96.24%. Similar results can be observed for the other datasets. Compared with *FastFlow*, the performance drop of *GGFast* may be due to its dependence on packet sequence patterns that can be affected by packet drops and retransmissions. For *GradBP*, apart from its choice of using a fixed number of packets for flow classification, its performance in detecting unknown flows may be affected by not having a dataset augmentation technique for training the unknown flow detection capability.

We also include some baselines that represent families of methods in the literature. For example, several prior works classify network flows using a fixed number of time-series data points of either packets or time-interval slots [3, 24, 39, 41]. We include the performance of standard LSTM time-series classifiers that take a fixed number of initial flow data points without the ability to detect unknown flows, therefore, their respective values in Table 3 are marked as '-'. These classifiers use the first five packets (named 'Pkt.-5' in Table 3), the first forty-five packets (named 'Pkt.45'), the first five slots (named 'Time-int.-5'), and the first forty-five slots (named 'Time-int.-45').

FastFlow outperforms most of the baselines. Pkt.-5 clearly does not have enough information to make precise classifications. Pkt.-45 shows that increasing the number of packets does improve classification performance but not enough to achieve *FastFlow*'s performance. We did not investigate the performance of packet classifiers with more than 45 packets since the inference times for Pkt.-45 are already much larger than *FastFlow*. Slot classifiers perform better than packet classifiers. Overall, five slots are not enough to guarantee good performance, but with 45 intervals, the classifiers can reach similar classification performance to *FastFlow* and even slightly outperform our proposal.

[1]Additional results of our extensive evaluation are available in Appendix B. They include *i*) ideal network flow conditions on the original datasets, *ii*) with only packet sequence disorders, and *iii*) with only unknown flow types. *FastFlow* classifiers perform satisfactorily in all three scenarios.

Table 4. Ablation study comparing *FastFlow* with only one input flow data representation.

Dataset	Method	Classification performance on known flow types				Unknown types	
		Macro F1 (%)	Accuracy (%)	Packets (#)	Time (s)	FPR (%)	TPR (%)
UTMobileNet	**FastFlow**	85.42	86.32	12.92 ± 9.48	0.57 ± 1.61	4.56	86.94
	Packet seq.	79.08	79.81	8.71 ± 4.05	**0.51 ± 0.43**	**3.63**	**88.24**
	Time-int. seq.	**87.54**	**88.61**	26.30 ± 9.10	1.63 ± 0.11	4.68	83.21
VNAT	**FastFlow**	96.24	98.03	4.16 ± 1.34	0.033 ± 1.43	**0**	97.32
	Packet seq.	94.29	96.49	3.97 ± 1.73	**0.006 ± 1.18**	**0**	**99.24**
	Time-int. seq.	**99.40**	**99.61**	7.93 ± 5.06	0.093 ± 1.32	**0**	98.78
UNIBS	**FastFlow**	92.30	95.21	9.92 ± 3.92	0.36 ± 0.82	2.54	**98.16**
	Packet seq.	90.51	92.37	5.44 ± 3.29	**0.27 ± 1.20**	**0**	97.92
	Time-int. seq.	**92.60**	**96.07**	18.02 ± 9.83	0.48 ± 0.42	3.18	94.4

However, this comes with a longer inference time. We also notice that all baselines cannot handle unknown flows, so we cannot measure their performance on this task.

In addition to the two fixed numbers (*i.e.,* 5 and 45) of packets or slots for flow classification, we have also tested other fixed numbers from 5 to 45 with a step of 5. The results, including accuracy and marco F1 score, are averaged for each dataset and are provided in Fig. 5. We can see that *FastFlow* outperforms all settings with fixed number of inputs in both accuracy and marco F1 score. From Fig. 5, it can also be observed that using more initial packets or time-interval slots to classify a flow does not necessarily produce better accuracy, validating the merit of *FastFlow* method that can dynamically estimate the minimal number of packets or slots to classify each candidate flow.

6.3 Ablation Study with Only Packet or Slot Flow Data Sequence

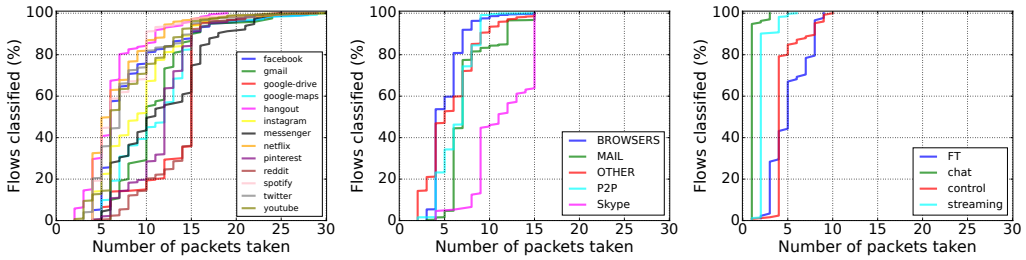
This section presents additional analyses that provide a better understanding of *FastFlow* performance. We start with an ablation study that characterizes the proposal's performance with either packet or slot representation.

FastFlow collectively uses two data representations: packet and slot data sequence. We analyze the importance of using both representations instead of one alone. This study excludes one time-series classifier (green boxes in Fig. 1) trained over one flow data representation (red or yellow boxes in the same figure). Table 4 summarizes the results, where *Packet seq.* stands for a *FastFlow* version using only a packet representation and *Time-int. seq.* using only a slot representation.

This assessment shows that the slot representation is the best representation to classify known flows, while the packet representation performs well for unknown flows. In terms of inference time efficiency, the packet representation is the best. By design, *FastFlow* inherits the best of each representation. Its known flow classification performance is inferior, but close to the slot representation. In contrast, its unknown flow identification is close to the packet representation, which performs best. Regarding inference efficiency, *FastFlow* is faster than the time representation but slower than the packet representation.

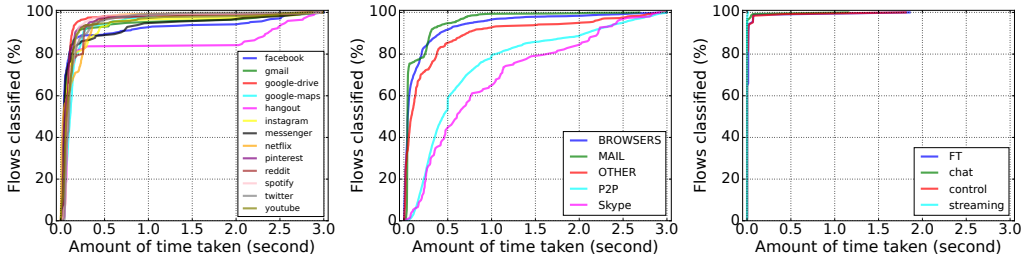
Toward this discussion, we now focus on the inference efficiency of *FastFlow*. The results in Tables 3 and 4 indicate that *FastFlow* is efficient. However, we only report the mean number of packets and inference time for all classes. Therefore, in Figs. 6 and 7, we provide details on the inference efficiency of *FastFlow* method for each flow type. Both plots show a *cumulative distribution function* (CDF). Fig. 6 shows the fraction of packets per flow type classified by *FastFlow* for a given number of packets. Fig. 7 similarly shows the fraction of packets per flow classified for a given time threshold in seconds.

Both plots show a high amount of variability across flow types and datasets. Regarding the datasets, UTMobileNet required the largest number of packets (*i.e.,* 30) to classify all packets, while UNIBS required half of this amount (*i.e.,* 15), and VNAT only needed 10 packets. We reckon that



(a) UTMobileNet (b) UNIBS (c) VNAT

Fig. 6. CDF plots for **number of packets taken** to classify each flow by *FastFlow* classifiers.



(a) UTMobileNet (b) UNIBS (c) VNAT

Fig. 7. CDF plots for the **amount of time taken** to classify each flow by *FastFlow* classifiers.

the number of packets may be correlated with the number of flow types in each dataset, as a larger number of flow types makes the recognition of individual flows more challenging.

Regarding recognizing individual flow types in each dataset, we notice that some can be readily classified, while others require more packets. This contributes to the standard deviation numbers observed in Tables 3 and 4. In both cases, these plots indicate the adaptability of *FastFlow* to recognize which flow types are more straightforward to classify.

6.4 *FastFlow* Deployment in a Live Network

Our final evaluation assesses the classification performance and inference efficiency of *FastFlow* in a realistic network deployment with a massive volume of data and a large number of unknown flows. During a one-week deployment from 1st to 7th November 2024, a copy of the traffic exchanged between our university campus border router and the Internet was streamed to our server running *FastFlow* prototype via two 10 Gbps network interfaces for inbound and outbound traffic[2], respectively. The server is configured with an 8-core Intel Xeon E5-2620 CPU and 64GB DDR4 RAM. As depicted in Fig. 1, two LSTM-based models are deployed to analyze real-time packet and slot time-series data, respectively. The number of model executions is directly determined by the number of packets and time-interval slots required to classify a candidate flow, as will be reported in the 'Packet (#)' and 'Time (s)' columns of Table 6 and 7. Given the simplified design of the classifier structure, the small dimension of the inference linear layer, and our engineering optimization including huge memory pages, large batch processing sizes, and concurrency, the prototype can be run on the server for our campus deployment (for a peak number of concurrent flows of around 50K and an average of 8.37 packets per flow) without performance issues being

[2]We have obtained ethical clearance approval as detailed in Appendix A, which allows us to analyze campus traffic for Internet application classifications without collecting information that can identify users such as university IDs and names.

observed. To train our classifiers, we have collected flows labeled by a commercial network traffic classification system from Canopus Networks Pty Ltd one week before the deployment, indicated by the '# train. flow' column in Table 5. With the about 1 million labeled flows, our classifiers are trained by a single NVIDIA-4090 GPU. The training process took about 6 hours.

Shown as the '# live flow' column, during the one-week live deployment, our prototype processed 22.9 million flows that belong to seven application types served by 32 content providers labeled by the commercial system deployed in parallel to our *FastFlow* prototype.

Table 5. Summary of the deployment data with the number of flows per application and content provider.

Application	Provider	#live flow	#train. flow
Video Streaming	MS Stream	1835585	136358
	Youtube	1835508	124088
	QQ	592163	12424
	WeChat	387700	31024
	Fastly	96750	15339
Software Update	Adobe	1814389	101212
	Windows	310918	34429
	Apple	820493	31090
	Ubuntu	10921	1763
Conferencing	Discord	337876	38945
	WhatsApp	30959	2589
	GoogleMeet	30034	1415
	MS Teams	154832	2010
	Facetime	9866	815
	Zoom	5424	361
Social Media	TikTok	1492556	115313
	Instagram	1344128	73665
	Facebook	759719	42566
	LinkedIn	185968	16363
	Reddit	154804	10566
	Twitter	126376	6679
File Storage	Apple iCloud	908163	13808
	MS Sharepoint	679058	25600
	Dropbox	112415	11778
	Google Drive	253460	8327
	OneDrive	59111	3485
Download	AmazonAWS	1212689	35159
	GoogleServices	882736	37417
	Google	755756	21928
	MS DotNET	10880	13701
Mail	Microsoft	738633	26950
	Google	87650	4216
Unknown	–	4075187	–

We analyze the performance of *FastFlow* for application types and their content providers. Table 6 shows the results for the application type. On average, *FastFlow* achieves an accuracy of 91.45%, and Macro F1 of 90.23%. The average number of packets is 8.37, and the average time to classify a flow is 0.5 seconds.

Among the seven **application types**, *FastFlow* achieves very high classification accuracy for conferencing (over 99%), mail (over 95%), and software update (over 93%), all of which have quite deterministic initial packet patterns compared to other application types. For mail and software, such patterns become confidently clear within the first eight packets with slight variations within 3 or 4 packets. In comparison, conferencing flows require about 3 to 18 packets for confident classification by *FastFlow* due to the diversified flow functionalities such as for video, audio, chats, and screen sharing as discussed in [33], each can have a unique initial packet sequence for function-specific requests.

The other four application types have decent classification performance with less than 10 packets and 1 to 2 seconds for over 85% accuracy or macro F1 compared to existing literature. The exception is social media flows, which are only classified with about 82% accuracy using up to 20 packets in 4 seconds. As we can see in the provider list of the social media application type in Table 5, the flows labeled as social media by the commercial system belong to providers offering a mixed set of services such as short videos (*e.g.,* Tiktok), picture sharing (*e.g.,* Instagram), online social platforms (*e.g.,* Facebook) and forums (*e.g.,* Reddit), which are inherently different in the content delivered via network flows compared to each other. With a maximum time step threshold C_{unk} value tuned as 25 packets, our classifiers can accurately detect 91.94% unknown flows that are labeled by the commercial system. The remaining unknown flows are classified as mostly 'social media' and 'conferencing', followed by 'download', as reported in the 'Unknown FPR' column of Table 6. We acknowledge that those false positives may not be truly misclassification as we use the labels provided by the commercial system as an estimation of ground truths, which has no means of always being correct, particularly for the applications (*e.g.,* social media and software update) that have providers not included in the ground-truth labels.

Table 6. Performance metrics of flow classification by *FastFlow* for application type.

Application	Macro F1 (%)	Accuracy (%)	Packets (#)	Time (s)	Unknown FPR (%)
Video Streaming	89.90	88.73	8.31 ± 4.38	0.21 ± 1.32	0.00
Software Update	92.50	93.18	7.22 ± 4.50	0.10 ± 1.06	0.00
Conferencing	98.63	99.82	10.57 ± 7.27	1.34 ± 3.89	4.34
Social Media	81.91	82.42	11.89 ± 5.65	1.59 ± 2.38	5.17
File Storage	89.13	84.75	8.83 ± 4.11	0.23 ± 2.17	0.00
Download	83.60	87.97	10.10 ± 6.22	0.20 ± 1.21	1.43
Mail	93.66	95.20	6.12 ± 3.33	1.68 ± 9.58	0.00
Unknown	–	91.94	20 ± 0.00	0.77 ± 2.69	–
Average	90.23	91.45	8.37	0.5	–

We also developed and deployed flow classifiers to determine the **content provider** of each flow with its identified application type. Given that classifying content providers within each application type reduces the classification scope and possible variations within the same flow type, we observe better performance for both accuracy and speed in all subsequent content provider classifiers compared to its preliminary application type classifiers. We show the classification performance of our content provider classifiers for two representative application types, including video streaming and software update in Table 7.

For video streaming content providers, our classifiers achieve superior performance in the three content providers primarily providing streaming services, including Microsoft Stream (used by our university for organizational video content such as lecture recording), YouTube, and Fastly. Over 97% accuracy is achieved with about 3 to 4 initial packets (less than 0.15 seconds) per flow. The video flows supported by QQ and WeChat, mainly known as social media applications with video streaming as their side features, require an average of 5 to 8 packets (less than 0.2 seconds) for classification accuracy over 89%. This shows the complexity of flow patterns for content providers that offer services across application types. Notably, all flows labeled (by the commercial system) as video flows get confidently classified by their provider types instead of being classified as the 'unknown' type shown in the last column of Table 7.

Software update has all its popular content providers accurately (over 93%) predicted with about six initial packets (less than 0.1 seconds) per flow. Windows and Apple, which have a diversified firmware catalog potentially supported by different product teams, are classified with lower accuracies (94.20% and 93.12%) compared to Adobe and Ubuntu (98.24% and 99.38%), which are with unified firmware/software portals for updates. From the last column of Table 7, a minority (less than 0.48%) of Adobe and Windows flows labeled by the commercial system are misclassified as 'unknown' by our *FastFlow* classifier, suggesting that further improvements on the fine-grained labeled dataset are needed to train a more accurate classifier on those application/provider types.

As discussed earlier for the result of application types, social media flows are inherently diversified in their functionalities, which also leads to a mediocre classification performance (80% to 90% in accuracy and Marco F1) to classify the content provider of each flow. This necessitates a ground-truth dataset with well-engineered flow types within the application type for training purposes, which is not within the scope of this paper. Similar insights are observed for other application types and are not explicitly discussed.

System deployment considerations: Here we discuss two practical considerations when deploying *FastFlow* in a live network. First, according to our performance benchmarking, around 30K concurrent flows can be processed by a time-series classifier on a single Intel Xeon E5-2620 CPU core of the server. Therefore, our 8-core CPU is sufficient for processing our live campus traffic with up to 50K concurrent flows. While not used in our current deployment, we have also benchmarked on a NVIDIA-4090 GPU that the two classifiers in *FastFlow* can process up to 200K concurrent flows.

Table 7. Performance of flow classification by *FastFlow* for representative application providers.

Application	Provider	Macro F1 (%)	Accuracy (%)	Packet (#)	Time (s)	Unknown FPR (%)
Video Streaming	Microsoft	98.70	99.29	3.25 ± 2.32	0.05 ± 0.44	0.00
	YouTube	97.98	97.14	4.25 ± 3.98	0.15 ± 1.51	0.00
	QQ	89.77	86.32	8.32 ± 4.68	0.19 ± 0.36	0.00
	WeChat	91.01	91.56	5.78 ± 5.43	0.11 ± 0.40	0.00
	Fastly	99.05	99.33	4.76 ± 1.56	0.01 ± 0.02	0.00
Software Update	Adobe	98.09	98.24	4.75 ± 2.44	0.02 ± 0.09	0.48
	Windows	93.84	94.20	6.40 ± 2.97	0.08 ± 0.51	0.38
	Apple	95.20	93.12	6.61 ± 4.16	0.06 ± 0.46	0.00
	Ubuntu	98.77	99.38	6.63 ± 1.55	0.13 ± 0.15	0.00
Conferencing	Discord	98.74	99.70	1.20 ± 1.59	0.04 ± 0.03	0.00
	Whatsapp	99.22	99.38	2.99 ± 2.30	0.39 ± 2.45	0.00
	GoogleMeet	98.41	96.87	4.00 ± 4.09	0.13 ± 0.05	0.03
	MS Teams	98.68	99.20	2.51 ± 1.45	0.57 ± 2.51	0.00
	FaceTime	97.77	95.65	3.65 ± 3.60	0.38 ± 1.80	0.00
	Zoom	97.41	98.62	4.12 ± 4.45	0.99 ± 3.62	0.07
Social Media	TikTok	83.51	86.36	6.31 ± 3.52	0.13 ± 0.18	0.00
	Instagram	85.17	88.97	11.16 ± 5.57	0.29 ± 1.39	0.00
	Facebook	82.35	82.12	9.90 ± 5.23	0.06 ± 0.28	0.05
	LinkedIn	81.16	89.09	10.52 ± 5.26	0.27 ± 1.62	0.00
	Reddit	84.61	88.85	9.09 ± 4.62	0.67 ± 4.10	0.00
	Twitter	84.06	88.14	3.85 ± 4.11	0.02 ± 0.10	0.00
File Storage	Apple iCloud	96.44	95.00	11.89 ± 4.62	0.05 ± 0.10	4.76
	MS Sharepoint	91.94	95.65	7.35 ± 4.40	0.05 ± 0.27	2.42
	Dropbox	96.42	97.29	8.12 ± 2.63	0.08 ± 0.12	0.00
	Google Drive	97.77	96.24	3.85 ± 4.11	0.02 ± 0.10	1.48
	OneDrive	88.37	84.73	9.63 ± 3.95	0.03 ± 0.07	0.00

For larger workloads, we suggest two **scaling-up options**, including leveraging GPU servers that are more efficient in executing neural network models and setting up parallel computing nodes, each processing a subnet of the monitored network. Second, the classification performance of machine learning classifiers can be significantly impacted by the **quality of the training data**. For *FastFlow*, its performance for both classification of known flow types and unknown flow detection is directly determined by the quality of the flow data used in the training process. In our proof-of-concept research prototype, we used a commercial network traffic classification system for flow labels. In industrial practice, such localized training data are often obtained from ISP digital twins or by the service team of a network observability platform.

7 Conclusion

In this paper, we present *FastFlow*, a time-series early flow classification method that can be practically deployed in large networks such as ISPs at runtime. By developing a dual-grained time-series flow representation scheme and innovating a time-series flow classifier architecture trained with reinforcement learning techniques, *FastFlow* is the first of its kind that addresses three key deployment challenges in large networks, including accurate classification with the estimated minimal number of initial packets in each candidate flow, robust to packet sequence disorders, and capable of detecting unknown flow types. We extensively validate the classification performance of *FastFlow* using public datasets and compare its performance with ablation alternatives and state-of-the-art methods. *FastFlow* is implemented and deployed in a large campus network to classify application types and content providers of network flows. The deployment insights showcase that *FastFlow* classifiers can accurately classify flows for their applications and content providers with only about 10 initial packets of each flow in less than one or two seconds, and are able to detect flows that do not belong to a known type.

Acknowledgement

We thank our shepherd Francesco Bronzino and the anonymous reviewers for their insightful feedback. This work is supported by the Australian Government's Cooperative Research Centres Projects (CRC-P) Grant CRCPXIV000099.

References

[1] Giuseppe Aceto, Domenico Ciuonzo, Antonio Montieri, and Antonio Pescapé. 2020. Toward effective mobile encrypted traffic classification through deep learning. *Neurocomputing* 409 (Apr 2020), 306–315.

[2] Iman Akbari, Mohammad A. Salahuddin, Leni Ven, Noura Limam, Raouf Boutaba, Bertrand Mathieu, Stephanie Moteau, and Stephane Tuffin. 2021. A Look Behind the Curtain: Traffic Classification in an Increasingly Encrypted Web. *Proceedings of the ACM on Measurement and Analysis of Computing Systems* 5, 1, Article 04 (Feb 2021), 26 pages.

[3] Hassan Alizadeh, Harald Vranken, André Zúquete, and Ali Miri. 2020. Timely Classification and Verification of Network Traffic Using Gaussian Mixture Models. *IEEE Access* 8 (May 2020).

[4] Omar Almomani, Mohammed Amin Almaiah, Adeeb Alsaaidah, Sami Smadi, Adel Hamdan Mohammad, and Ahmad Althunibat. 2021. Machine learning classifiers for network intrusion detection system: comparative study. In *Proc. International Conference on Information Technology*. 440–445.

[5] Ahmad Azab, Mahmoud Khasawneh, Saed Alrabaee, Kim-Kwang Raymond Choo, and Maysa Sarsour. 2024. Network Traffic Classification: Techniques, Datasets, and Challenges. *Digital Communications and Networks* 10, 3 (Jul 2024).

[6] Andrew Gehret Barto, Richard S Sutton, and CJCH Watkins. 1989. *Learning and Sequential Decision Making*. Vol. 89. University of Massachusetts Amherst, MA.

[7] David Berend, Xiaofei Xie, Lei Ma, Lingjun Zhou, Yang Liu, Chi Xu, and Jianjun Zhao. 2021. Cats are not fish: deep learning testing calls for out-of-distribution awareness. In *Proc. Automated Software Engineering*. Virtual Event, Australia, 1041–1052.

[8] Laurent Bernaille, Renata Teixeira, Ismael Akodkenou, Augustin Soule, and Kave Salamatian. 2006. Traffic Classification on the Fly. *ACM SIGCOMM Computer Communication Review* 36, 2 (Apr 2006), 23–26.

[9] Jakub Michał Bilski and Agnieszka Jastrzębska. 2023. CALIMERA: A new early time series classification method. *Information Processing and Management* 60, 5 (Jul 2023), 103465–103498.

[10] Agathe Blaise, Mathieu Bouet, Vania Conan, and Stefano Secci. 2020. Detection of zero-day attacks: An unsupervised port-based approach. *Computer Networks* 180 (Oct 2020), 107391.

[11] Francesco Carrera, Vincenzo Dentamaro, Stefano Galantucci, Andrea Iannacone, Donato Impedovo, and Giuseppe Pirlo. 2022. Combining unsupervised approaches for near real-time network traffic anomaly detection. *Applied Sciences* 12, 3 (Jan 2022), 1759.

[12] Hyunseok Chang, Matteo Varvello, Fang Hao, and Sarit Mukherjee. 2021. Can You See Me Now? A Measurement Study of Zoom, Webex, and Meet. In *Proc. ACM IMC*. Virtual Event.

[13] Albert Choi, Mehdi Karamollahi, Carey Williamson, and Martin Arlitt. 2022. Zoom Session Quality: A Network-Level View. In *Proc. PAM*. Virtual Event.

[14] Murat Dener, Samed Al, and Gokce Ok. 2023. RFSE-GRU: Data Balanced Classification Model for Mobile Encrypted Traffic in Big Data Environment. *IEEE Access* (Jan 2023).

[15] Gerard Draper-Gil, Arash Habibi Lashkari, Mohammad Saiful Islam Mamun, and Ali A Ghorbani. 2016. Characterization of Encrypted and VPN Traffic Using Time-Related Features. In *Proc. International Conference on Information Systems Security and Privacy*. Rome, Italy, 407–414.

[16] Alice Este, Francesco Gringoli, and Luca Salgarelli. 2011. On-Line SVM Traffic Classification. In *Proc. International Wireless Communications and Mobile Computing Conference*. Istanbul, Turkey.

[17] Johan Garcia and Topi Korhonen. 2018. Efficient Distribution-Derived Features for High-Speed Encrypted Flow Classification. In *Proc. SIGCOMM Workshop on Network Meets AI and ML*. Budapest, Hungary, 21–27.

[18] Alex Graves and Alex Graves. 2012. Long short-term memory. *Supervised sequence labelling with recurrent neural networks* (Feb 2012), 37–45.

[19] Yuqiang Heng, Vikram Chandrasekhar, and Jeffrey G. Andrews. 2021. UTMobileNetTraffic2021: A Labeled Public Network Traffic Dataset. *IEEE Networking Letters* 3, 3 (Dec 2021), 156–160.

[20] Jordan Holland, Paul Schmitt, Nick Feamster, and Prateek Mittal. 2021. New Directions in Automated Traffic Analysis. In *Proc. ACM CCS*. Virtual Event, Republic of Korea, 3366–3383.

[21] Nen-Fu Huang, Gin-Yuan Jai, Han-Chieh Chao, Yih-Jou Tzang, and Hong-Yi Chang. 2013. Application Traffic Classification at the Early Stage by Characterizing Application Rounds. *Information Sciences* 232 (May 2013), 130–142.

[22] Auwal Sani Iliyasu and Huifang Deng. 2020. Semi-Supervised Encrypted Traffic Classification With Deep Convolutional Generative Adversarial Networks. *IEEE Access* 8 (May 2020).

[23] Chi Jin, Zeyuan Allen-Zhu, Sebastien Bubeck, and Michael I Jordan. 2018. Is Q-learning provably efficient? *Advances in Neural Information Processing Systems* 31 (Dec 2018).
[24] Steven Jorgensen, John Holodnak, Jensen Dempsey, Karla de Souza, Ananditha Raghunath, Vernon Rivet, Noah DeMoes, Andrés Alejos, and Allan Wollaber. 2024. Extensible Machine Learning for Encrypted Network Traffic Application Labeling via Uncertainty Quantification. *IEEE Transactions on Artificial Intelligence* 5, 1 (Jan 2024), 420–433.
[25] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. 2016. Continuous Control with Deep Reinforcement Learning. In *Proc. ICLR*. San Juan, Puerto Rico.
[26] Yang Liu, Jinfu Chen, Peng Chang, and Xiaochun Yun. 2017. A novel algorithm for encrypted traffic classification based on sliding window of flow's first N packets. In *Proc. IEEE International Conference on Computational Intelligence and Applications*. Beijing, China, 463–470.
[27] Manuel Lopez-Martin, Belén Carro, Antonio Sanchez-Esguevillas, and Jaime Lloret. 2017. Network Traffic Classifier With Convolutional and Recurrent Neural Networks for Internet of Things. *IEEE Access* (Sep 2017).
[28] Minzhao Lyu, Sharat Chandra Madanapalli, Arun Vishwanath, and Vijay Sivaraman. 2024. Network Anatomy and Real-Time Measurement of Nvidia GeForce NOW Cloud Gaming. In *Proc. PAM*. Virtual Event, 61–91.
[29] Minzhao Lyu, Rahul Dev Tripathi, and Vijay Sivaraman. 2023. MetaVRadar: Measuring Metaverse Virtual Reality Network Activity. *Proceedings of the ACM on Measurement and Analysis of Computing Systems* 7, 3 (Sep 2023), 1–29.
[30] Kyle MacMillan, Tarun Mangla, James Saxon, and Nick Feamster. 2021. Measuring the Performance and Network Utilization of Popular Video Conferencing Applications. In *Proc. ACM IMC*. Virtual Event.
[31] Sharat Chandra Madanapalli, Hassan Habibi Gharakheili, and Vijay Sivaraman. 2022. Know Thy Lag: In-Network Game Detection and Latency Measurement. In *Proc. PAM*. Virtual Event, 395–410.
[32] Anthony McGregor, Mark Hall, Perry Lorier, and James Brunskill. 2004. Flow Clustering Using Machine Learning Techniques. In *Proc. PAM*. Antibes Juan-les-Pins, France, 205–214.
[33] Oliver Michel, Satadal Sengupta, Hyojoon Kim, Ravi Netravali, and Jennifer Rexford. 2022. Enabling Passive Measurement of Zoom Performance in Production Networks. In *Proc. ACM IMC*. Nice, France.
[34] Agnieszka Mikołajczyk and Michał Grochowski. 2018. Data Augmentation for Improving Deep Learning in Image Classification Problem. In *Proc. IEEE International Interdisciplinary PhD Workshop*. Gdansk, Poland, 117–122.
[35] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. 2016. Asynchronous Methods for Deep Reinforcement Learning. In *Proc. ICML*. New York City, USA, 1928–1937.
[36] Obkio. 2024. What is Acceptable Packet Loss? 10% Packet Loss = 100x Slower. https://obkio.com/acceptable-packet-loss/ Accessed: 2024-11-08.
[37] Fannia Pacheco, Ernesto Exposito, Mathieu Gineste, Cedric Baudoin, and Jose Aguilar. 2019. Towards the Deployment of Machine Learning Solutions in Network Traffic Classification: A Systematic Survey. *IEEE Communications Surveys & Tutorials* 21, 2 (Jun 2019), 1988–2014.
[38] Lizhi Peng, Bo Yang, and Yuehui Chen. 2015. Effective Packet Number for Early Stage Internet Traffic Identification. *Neurocomputing* 156 (May 2015), 252–267.
[39] Julien Piet, Dubem Nwoji, and Vern Paxson. 2023. GGFAST: Automating Generation of Flexible Network Traffic Classifiers. In *Proc. ACM SIGCOMM*. New York, USA, 850–866.
[40] Buyu Qu, Zhibin Zhang, Li Guo, and Dan Meng. 2012. On Accuracy of Early Traffic Classification. In *Proc. IEEE International Conference on Networking, Architecture, and Storage*. Xiamen, China, 348–354.
[41] Sangita Roy, Tal Shapira, and Yuval Shavitt. 2022. Fast and lean encrypted Internet traffic classification. *Computer Communications* 186 (May 2022), 166–173.
[42] Ola Salman, Imad H Elhajj, Ayman Kayssi, and Ali Chehab. 2020. A Review on Machine Learning–Based Approaches for Internet Traffic Classification. *Annals of Telecommunications* 75, 11 (Nov 2020), 673–710.
[43] Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. 2016. Prioritized Experience Replay. In *Proc. ICLR*. San Juan, Puerto Rico.
[44] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal Policy Optimization Algorithms. *arXiv preprint arXiv:1707.06347* (Jul 2017).
[45] Tal Shapira and Yuval Shavitt. 2021. FlowPic: A Generic Representation for Encrypted Traffic Classification and Applications Identification. *IEEE Transactions on Network and Service Management* 18, 2 (Sep 2021), 1218–1232.
[46] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. 2016. Mastering the Game of Go with Deep Neural Networks and Tree Search. *Nature* 529, 7587 (Jan 2016), 484–489.
[47] Hado van Hasselt, Arthur Guez, and David Silver. 2016. Deep Reinforcement Learning with Double Q-learning. In *Proc. AAAI Conference on Artificial Intelligence*. Phoenix, USA, 2094–2100.
[48] Garry Wan, Shinan Liu, Francesco Bronzino, Nick Feamster, and Zakir Durumeric. 2025. CATO: End-to-end Optimization of ML Traffic Analysis Pipelines. In *Proc. USENIX NSDI*. Philadelphia, PA, USA.

[49] Wei Wang, Y. Sheng, Jinlin Wang, Xuewen Zeng, Xiaozhou Ye, Yongzhong Huang, and Ming Zhu. 2017. HAST-IDS: Learning Hierarchical Spatial-Temporal Features using Deep Neural Networks to Improve Intrusion Detection. *IEEE Access* (Dec 2017).
[50] Wei Wang, Ming Zhu, Jinlin Wang, Xuewen Zeng, and Zhongzhen Yang. 2017. End-to-end encrypted traffic classification with one-dimensional convolution neural networks. In *Proc. IEEE Intelligence and Security Informatics*. Beijing, China.
[51] Yu Wang, Chao Chen, and Yang Xiang. 2015. Unknown Pattern Extraction for Statistical Network Protocol Identification. In *Proc. IEEE LCN*. Clearwater Beach, Florida, 506–509.
[52] Yipeng Wang, Huijie He, Yingxu Lai, and Alex X. Liu. 2022. A Two-Phase Approach to Fast and Accurate Classification of Encrypted Traffic. *IEEE/ACM Trans. Netw.* 31, 3 (Jun 2022), 1071–1086.
[53] Yifan Wang, Minzhao Lyu, and Vijay Sivaraman. 2024. Characterizing User Platforms for Video Streaming in Broadband Networks. In *Proc. ACM IMC*. Madrid, Spain.
[54] Ziyu Wang, Tom Schaul, Matteo Hessel, Hado Hasselt, Marc Lanctot, and Nando Freitas. 2016. Dueling Network Architectures for Deep Reinforcement Learning. In *Proc. ICML*. New York City, USA, 1995–2003.
[55] Christopher J. C. H. Watkins and Peter Dayan. 1992. Q-learning. *Machine Learning* 8, 3 (Jul 1992), 279–292.
[56] WebSentra. 2024. Packet Loss: Understanding, Diagnosing & Fixing in Networks. https://www.websentra.com/packet-loss-understanding-diagnosing-fixing-in-networks/ Accessed: 2024-11-08.
[57] Sebastien C Wong, Adam Gatt, Victor Stamatescu, and Mark D McDonnell. 2016. Understanding Data Augmentation for Classification: When to Warp?. In *Proc. International Conference on Digital Image Computing: Techniques and Applications*. Gold Coast, Australia.
[58] Charles V Wright, Fabian Monrose, and Gerald M Masson. 2006. Using visual motifs to classify encrypted traffic. In *Proc. International Workshop on Visualization for Computer Security*. Alexandria, VA, USA, 41–50.
[59] Baris Yamansavascilar, M. Amac Guvensan, A. Gokhan Yavuz, and M. E. Karsligil. 2017. Application Identification via Network Traffic Classification. In *Proc. International Conference on Computer Networks and Communications*. Silicon Valley, USA, 843–848.
[60] Lixuan Yang, Alessandro Finamore, Feng Jun, and Dario Rossi. 2021. Deep Learning and Zero-Day Traffic Classification: Lessons Learned From a Commercial-Grade Dataset. *IEEE Transactions on Network and Service Management* 18, 4 (Sep 2021), 4103–4118.
[61] Ruixi Yuan, Zhu Li, Xiaohong Guan, and Li Xu. 2010. An SVM-based Machine Learning Method for Accurate Internet Traffic Classification. *Inf. Syst. Front.* 12 (Mar 2010), 149–156.
[62] Sebastian Zander, Thuy Nguyen, and Grenville Armitage. 2005. Automated Traffic Classification and Application Identification Using Machine Learning. In *Proc. IEEE LCN*. Sydney, Australia, 250–257.
[63] Jun Zhang, Chao Chen, Yang Xiang, Wanlei Zhou, and Athanasios V. Vasilakos. 2013. An Effective Network Traffic Classification Method with Unknown Flow Detection. *IEEE Trans. Netw. Serv. Manag.* 10, 2 (Jun 2013), 133–147.
[64] Jun Zhang, Xiao Chen, Yang Xiang, Wanlei Zhou, and Jie Wu. 2015. Robust Network Traffic Classification. *IEEE/ACM Trans. Netw.* 23, 4 (Aug 2015), 1257–1270.
[65] Zhuang Zou, Jingguo Ge, Hongbo Zheng, Yulei Wu, Chunjing Han, and Zhongjiang Yao. 2018. Encrypted traffic classification with a convolutional long short-term memory neural network. In *Proc. IEEE HPCC*. Exeter, UK, 329–334.

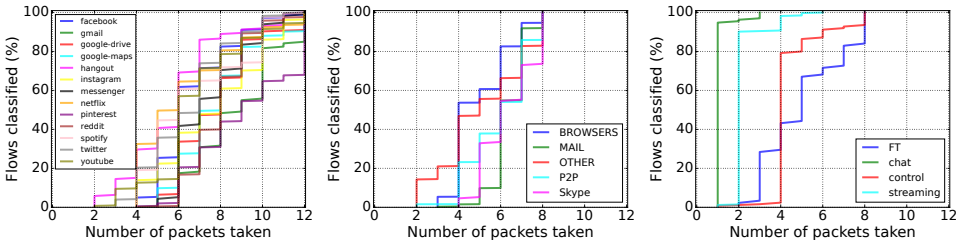
A Ethics

We have obtained ethical clearance from our university ethics board (UNSW Human Research Ethics Advisory Panel approval number HC211007) which allows us to analyze campus traffic for Internet applications without being able to access user identities such as ID numbers and names. In our campus deployment, insights into application types and providers were reported in an aggregated manner, preserving anonymity rather than identifying specific users. In our analysis, no attempt was made to associate network flows with personal identities.

B Additional Lab Evaluation Results of *FastFlow* Classification Performance

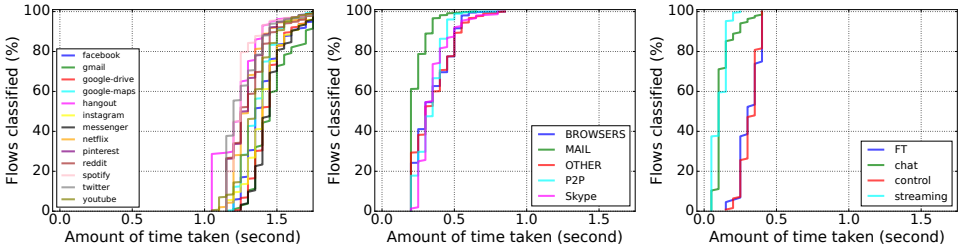
B.1 Speed of *FastFlow* Classifiers with Only Packet or Slot Data Sequence

B.1.1 Using Only Packet Data Sequence. Fig. 8 contains CDF plots showing the number of packets taken to classify flows by the classifiers that only consume the packet data sequence. For the three datasets, the number of packets required for flow classification is usually less than 8. However, as discussed before, our classifiers that only use packet data sequence cannot achieve decent performance when there are packet drops and retransmissions in a candidate flow.



(a) UTMobilenet (b) UNIBS (c) VNAT

Fig. 8. CDF plots for number of packets taken by *FastFlow* classifiers on **only packet** data sequence.



(a) UTMobileNet (b) UNIBS (c) VNAT

Fig. 9. CDF plots for amount of time taken by *FastFlow* classifiers on **only slot** data sequence.

Table 8. Classification performance of *FastFlow* on ideal network flow dataset without packet sequence disorder or unknown flow type.

Dataset	Method	Macro F1 (%)	Accuracy (%)	Packets Taken (#)	Time Taken (s)
UTMobileNet	**FastFlow**	89.44	90.54	13.96 ± 5.94	0.70 ± 2.45
	Packet seq.	90.83	92.20	9.07 ± 3.91	0.47 ± 2.73
	Time-int. seq.	88.74	91.50	26.50 ± 12.95	1.32 ± 3.80
	GGFast [39]	88.04	90.56	50	2.46 ± 1.98
	Grad-BP [60]	89.08	88.11	100(TCP) 10(UDP)	4.77 ± 2.31
	Pkt.-5	69.97	74.33	5 ± 0.00	0.24 ± 1.95
	Pkt.-45	89.76	91.06	45 ± 0.00	2.30 ± 1.47
	Time-int.-5	73.41	74.02	4.91 ± 2.72	0.25
	Time-int.-45	86.47	91.10	44.92 ± 7.21	2.25
VNAT	**FastFlow**	95.00	96.31	4.36 ± 6.35	0.02 ± 1.29
	Packet seq.	95.22	96.60	2.94 ± 2.31	.007 ± 0.15
	Time-int. seq.	92.25	95.17	10.57 ± 4.20	0.15 ± 0.24
	GGFast [39]	77.56	90.81	50	1.07 ± 1.35
	Grad-BP [60]	93.70	95.92	100(TCP) 10(UDP)	1.92 ± 1.35
	Pkt.-5	95.76	96.78	5 ± 0.00	0.03 ± 0.62
	Pkt.-45	87.87	87.92	45 ± 0.00	0.90 ± 2.86
	Time-int.-5	90.10	92.17	14.89 ± 5.24	0.25 ± 0.00
	Time-int.-45	80.71	82.09	107.45 ± 18.19	2.25 ± 0.00
UNIBS	**FastFlow**	95.87	98.34	5.42 ± 2.19	0.28 ± 0.60
	Packet seq.	95.87	98.34	5.42 ± 2.19	0.28 ± 0.60
	Time-int. seq.	93.57	97.05	10.26 ± 8.66	0.43 ± 1.68
	GGFast [39]	91.97	95.03	50	1.79 ± 3.40
	Grad-BP [60]	93.87	97.22	100(TCP) 10(UDP)	3.28 ± 2.05
	Pkt.-5	86.68	93.82	5 ± 0.00	0.26 ± 0.46
	Pkt.-45	92.30	92.44	45 ± 0.00	1.56 ± 3.06
	Time-int.-5	72.41	76.32	4.84 ± 2.37	0.25
	Time-int.-45	91.60	93.39	64.60 ± 9.75	2.25

Table 9. Classification performance of *FastFlow* with packet sequence disorders in each candidate flow.

Dataset	Method	Macro F1 (%)	Accuracy (%)	Packets Taken (#)	Time Taken (s)
UTMobileNet	**FastFlow**	85.32	87.21	12.83 ± 6.74	0.93 ± 1.67
	Packet seq.	80.30	82.05	9.29 ± 4.00	0.54 ± 0.76
	Time-int. seq.	86.69	88.20	18.21 ± 9.96	1.37 ± 2.36
	GGFast	72.43	78.49	50	2.49 ± 1.61
	Grad-BP	82.59	84.25	100(TCP) 10(UDP)	5.02 ± 1.85
	Pkt.-5	64.72	71.73	5 ± 0.00	0.28 ± .202
	Pkt.-45	81.20	82.84	45 ± 0.00	2.35 ± 1.94
	Time-int.-5	70.28	72.62	3.92 ± 2.78	.25
	Time-int.-45	85.90	89.04	44.09 ± 8.53	2.25
VNAT	**FastFlow**	91.32	94.82	4.88 ± 6.79	.061 ± 0.43
	Packet seq.	89.42	93.71	3.03 ± 2.59	.021 ± 0.33
	Time-int. seq.	91.82	94.89	10.08 ± 7.2	0.19 ± 0.24
	GGFast	65.22	76.48	50	1.28 ± .641
	Grad-BP	88.62	90.33	100(TCP) 10(UDP)	2.47 ± 0.94
	Pkt.-5	88.91	92.34	5 ± 0.00	0.04 ± .399
	Pkt.-45	82.10	84.20	45 ± 0.00	1.11 ± 2.94
	Time-int.-5	89.46	91.89	12.23 ± 6.02	.25
	Time-int.-45	80.27	82.11	86.77 ± 17.02	2.25
UNIBS	**FastFlow**	92.04	97.28	6.99 ± 4.17	0.40 ± 0.91
	Packet seq.	89.06	94.86	5.59 ± 2.38	0.39 ± 2.27
	Time-int. seq.	92.33	96.23	10.09 ± 10.02	0.56 ± 1.16
	GGFast	74.31	79.58	50	1.82 ± 0.92
	Grad-BP	90.62	93.48	100(TCP) 10(UDP)	3.71 ± 1.33
	Pkt.-5	80.31	88.85	5 ± 0.00	0.32 ± 0.96
	Pkt.-45	89.55	95.7	45 ± 0.00	1.61 ± 0.81
	Time-int.-5	69.76	73.80	2.12 ± 2.42	.25
	Time-int.-45	91.29	92.94	63.96 ± 7.97	2.25

B.1.2 Using Only Time-Interval Slot Data Sequence. Fig. 9 contains CDF plots for the amount of time needed for classifiers that only uses time-interval slot data sequence. Compared to the CDF plots in Fig. 7 for *FastFlow* classifiers using both packet and slot data sequences, we can conclude that *FastFlow* performs significantly faster than the classifiers that only use slot data sequence.

B.2 Flow Classification Performance

B.2.1 Ideal Conditions without Packet Sequence Disorder and Unknown Flow Type. Table 8 shows the classification results without augmenting the three public datasets. As expected, the macro-F1 and accuracy scores for all methods are higher compared to those with packet sequence disorder introduced to the datasets. *FastFlow* performs slightly better than two state-of-the-art methods (*i.e., GGFast* and *Grad-BP*) on the three datasets. Also, *FastFlow* requires much smaller number of packets on average to classify flows compared to the state-of-the-art methods. For the ablation studies, on the datasets without introduced packet sequence disorder, *FastFlow* classifiers that only use packet data (*i.e.,* 'Packet seq.') achieved equivalent accuracies on the three datasets compared to the classifiers that use both packets and time-interval slots. Classifiers that only use time-interval slot data (*i.e.,* 'Time-int. seq.') are less accurate in such scenarios. *FastFlow* also outperforms all classifiers using fixed length of input data.

B.2.2 With Packet Sequence Disorders but not Unknown Flow Type. Table 9 shows the evaluation results when the datasets are augmented with packet sequence disorders. It can be seen that *FastFlow* achieves a balanced performance in both classification accuracy and speed compared to all methods. Classifiers that only use packet data (*i.e.,* 'Packet seq', 'Pkt.-5' and 'Pkt.-45') are not robust to packet sequence disorders. Classifiers that only use time-interval slot data (*i.e.,* 'Time-int

Table 10. Classification performance of *FastFlow* with unknown flow type.

Dataset	Method	Classification Performance				Unk. flow detection	
		Macro F1 (%)	Accuracy (%)	Packets (#)	Time (s)	FPR (%)	TPR (%)
UTMobileNet	**FastFlow**	89.21	90.2	12.74 ± 5.99	0.59 ± 1.07	4.16	88.21
	Packet seq.	90.07	90.7	8.51 ± 3.63	0.41 ± 0.23	2.81	90.01
	Time-int. seq.	89.33	89.72	29.63 ± 10.29	1.68 ± 3.79	4.59	85.38
	GGFast	90.35	91.49	50	2.92 ± .96	4.97	92.81
	Grad-BP	91.21	91.68	100(TCP) 10(UDP)	5.73 ± 1.49	4.88	69.87
	Pkt.-5	72.95	77.78	5	0.15 ± 1.99	–	–
	Pkt.-45	88.96	90.07	45	2.61 ± 0.79	–	–
	Time-int.-5	69.51	74.93	6.49 ± 2.94	0.25	–	–
	Time-int.-45	88.85	89.49	39 ± 6.04	2.25	–	–
VNAT	**FastFlow**	99.33	99.46	3.89 ± 1.61	.06 ±0.74	0	99.39
	Packet seq.	99.33	99.46	3.89 ± 1.61	.06 ±0.07	0	99.20
	Time-int. seq.	99.70	99.80	7.85 ± 4.35	.088 ± 1.64	0	98.93
	GGFast	84.28	90.72	50	0.95 ± .62	2.79	99.42
	Grad-BP	99.77	99.79	100(TCP) 10(UDP)	1.97 ± 0.84	4.98	99.30
	Pkt.-5	92.01	96.50	5	0.03 ± 0.83	–	–
	Pkt.-45	94.72	98.67	45	0.84 ± 0.59	–	–
	Time-int.-5	93.42	98.96	15.79 ± 3.07	0.25	–	–
	Time-int.-45	90.43	93.50	93.52 ± 11.96	2.25	–	–
UNIBS	**FastFlow**	94.56	97.80	6.32 ± 2.68	0.28 ± 1.30	1.33	97.32
	Packet seq.	95.10	97.79	5.19 ± 3.02	0.23 ± 1.16	0	98.29
	Time-int. seq.	92.62	96.18	11.41 ± 7.81	0.39 ± 0.83	4.18	98.40
	GGFast	93.12	97.45	50	1.35 ± 1.31	3.43	99.03
	Grad-BP	89.09	96.11	100(TCP) 10(UDP)	2.53 ± 1.63	4.90	73.90
	Pkt.-5	85.08	89.81	5	0.23 ± 0.93	–	–
	Pkt.-45	93.56	97.54	45	1.22 ± 1.84	–	–
	Time-int.-5	82.45	87.01	5.62 ± 1.92	0.25	–	–
	Time-int.-45	92.73	95.63	86.01 ± 21.02	2.25	–	–

seq.', 'Time-int.-5' and 'Time-int.-45') are either not having good accuracy with small numbers of inputs or requiring long time (*e.g.*, over 45 seconds) for flow classification.

B.2.3 With Unknown Flow Types but not Packet Sequence Disorder. Table 10 reports the evaluation results on the three datasets augmented with unknown flows. Compared to the two state-of-the-arts methods and *FastFlow* classifiers with only packet or time-interval slot data sequences, *FastFlow* achieves decent performance in both classification accuracy and speed for both known and unknown flows.

Received January 2025; revised April 2025; accepted April 2025