Neural Computing and Applications (2024) 36:20401–20424
https://doi.org/10.1007/s00521-024-10281-4

ORIGINAL ARTICLE

Incremental federated learning for traffic flow classification in heterogeneous data scenarios

Adrian Pekar[1,2] · Laszlo Arpad Makara[1] · Gergely Biczok[1,2]

Received: 6 August 2023 / Accepted: 29 July 2024 / Published online: 12 August 2024

Abstract

This paper explores the comparative analysis of federated learning (FL) and centralized learning (CL) models in the context of multi-class traffic flow classification for network applications, a timely study in the context of increasing privacy preservation concerns. Unlike existing literature that often omits detailed class-wise performance evaluation, and consistent data handling and feature selection approaches, our study rectifies these gaps by implementing a feed-forward neural network and assessing FL performance under both independent and identically distributed (IID) and non-independent and identically distributed (non-IID) conditions, with a particular focus on incremental training. In our cross-silo experimental setup involving five clients per round, FL models exhibit notable adaptability. Under IID conditions, the accuracy of the FL model peaked at 96.65%, demonstrating its robustness. Moreover, despite the challenges presented by non-IID environments, our FL models demonstrated significant resilience, adapting incrementally over rounds to optimize performance; in most scenarios, our FL models performed comparably to the idealistic CL model regarding multiple well-established metrics. Through a comprehensive traffic flow classification use case, this work (i) contributes to a better understanding of the capabilities and limitations of FL, offering valuable insights for the real-world deployment of FL, and (ii) provides a novel, large, carefully curated traffic flow dataset for the research community.

Keywords Federated learning · Incremental learning · Centralized learning · Network traffic classification

1 Introduction

Computer networks have witnessed significant innovation over the past decades, characterized by the development of new techniques, paradigms, and protocols, as well as the proliferation of diverse services and applications [14, 17, 47, 49, 54]. Amidst these advancements, network traffic flow classification (TFC) has emerged as a cornerstone of modern network and service management. TFC involves the identification and mapping of packet flows to specific traffic types, providing rich information that enhances various management tasks, including traffic routing, congestion control, network security, and quality of service (QoS) and service level agreement (SLA) management. This task, however, is challenged by the increasing complexity and variability of network traffic patterns. Differentiating among various traffic types requires sophisticated techniques capable of capturing and interpreting intricate traffic behavior.

Despite the integration of a diverse range of techniques into TFC, such as machine learning (ML), the effective analysis and classification of different traffic types remain challenging [39, 40]. ML techniques have proven efficient in processing and categorizing complex traffic behaviors, even with massive data volumes. However, data privacy concerns and stringent data protection measures limit data sharing with researchers, professionals, and across

✉ Adrian Pekar
apekar@hit.bme.hu

Laszlo Arpad Makara
laszloarpad.makara@edu.bme.hu

Gergely Biczok
biczok@crysys.hu

[1] Department of Networked Systems and Services, Faculty of Electrical Engineering and Informatics, Budapest University of Technology and Economics, Műegyetem rkp. 3., Budapest 1111, Hungary

[2] HUN-REN-BME Information Systems Research Group, Magyar Tudósok krt. 2, Budapest 1117, Hungary

industry, businesses, and academia. Consequently, the practical usability and advancement of TFC solutions are hindered by the constrained availability of shared, labeled traffic datasets [21, 53, 67].

Federated learning (FL) offers an ingenious solution to privacy concerns in traffic flow classification. FL addresses this by allowing for a decentralized training process where the data resides locally and does not need to be transferred to a central location for processing [35]. This approach ensures that sensitive network traffic data remains within its original premises, drastically reducing the risk of data leakage and ensuring compliance with privacy regulations. Furthermore, despite the distributed nature of FL, it allows for collaborative model development, ensuring that the global model benefits from the individual insights gleaned from each local dataset.

This study examines the performance of incremental federated learning across a spectrum of data distribution scenarios, from independent and identically distributed (IID) to various levels of non-independent and identically distributed (non-IID) complexities. We introduce a novel approach to FL tailored to the dynamic nature of network traffic flow measurements. Our experiment design, involving five clients in a FL setup, is structured to closely emulate real-world conditions. We implement a distinct dataset chunk distribution strategy for both training and validation phases across ten rounds of FL, ensuring that each client receives unique data chunks per round without reuse, reflecting diverse traffic segments. This methodology prevents data overlap and promotes an incremental learning process by exposing the model to new data in each round.

The integral incremental learning approach in our FL setup ensures that the models are continually refined and updated, reflecting the dynamic nature of real-world network traffic. This approach, diverging from static learning paradigms, amplifies the practical advantages and applicability of FL in real-world settings, particularly highlighting its significance in ensuring privacy preservation and the efficient handling of heterogeneous data distributions.

We compare the FL results to an idealistic baseline obtained via centralized learning (CL). This evaluation is conducted within the context of multi-class traffic flow classification, encompassing a diverse array of network applications. For this purpose, we have developed a multi-class classification model employing a feed-forward neural network. The comprehensive evaluation of our models is carried out through several lenses: validation accuracy and loss, model performance metrics (precision, recall, and F1-score), and detailed insights drawn from normalized confusion matrices.

Our examination of FL models reveals their resilience and ability to incrementally improve across federated rounds. Specifically, in the IID scenario, our FL model demonstrated a notable increase in validation accuracy, starting at 90.38% in the initial round and reaching up to 96.65% by the tenth round. In comparison, non-IID scenarios, despite their inherent complexities, showcased the FL models' adaptability, with the FL model reaching a peak accuracy of 97.09% in the simplest scenario. This demonstrates the effectiveness of the incremental federated learning (IFL) approach even under demanding conditions. While challenges were observed in the most complex non-IID scenarios, the FL model still performed admirably and indicated potential for future improvements.

Moreover, the associated validation loss, precision, recall, and F1-score metrics, complemented by normalized confusion matrices, align with the findings from the validation accuracy analysis. These results highlight the proficiency of FL models in leveraging decentralized and varied datasets, delivering a performance that rivals those of traditional CL approaches.

Our contribution extends beyond existing studies by

(i) Identifying gaps in precursory works, particularly the inconsistencies in data collection and feature selection and the often-overlooked need for class-wise performance evaluation.
(ii) Reducing biases inherent in prior studies by maintaining stringent control over traffic flow measurement and feature computation. The result of this controlled process is a novel traffic flow dataset, which we share with the research community.
(iii) Offering a transparent and reproducible methodology, which includes the class-wise performance evaluation of network traffic flow classification, among other aspects.
(iv) Introducing an innovative incremental learning design within FL, which not only enhances the realism in FL applications but also effectively narrows the divide between theoretical machine learning constructs and their application in real-world network traffic monitoring.

The remainder of this paper is structured as follows: Sect. 2 provides a concise overview of traditional machine learning-based traffic flow classification. Section 3 delves into the challenges associated with collaborative traffic flow classification, specifically concerning data protection, and business interests. Section 4 offers a review of pertinent literature in the field. Section 5 introduces our approach to FL-based traffic flow classification. Section 6 outlines the design of our experimental approach. Section 7 presents the results obtained from the experiment. Section 8

provides a discussion of these results. Finally, Sect. 9 concludes the paper, highlighting our key findings and their implications.

2 Flow classification using machine learning

The workflow for traditional ML-based traffic flow classification, from packet capture to application categorization, is depicted in Fig. 1. The major steps of this process can be summarized as follows.

Traffic collection involves capturing network traffic data. Generally, there are two modes of traffic collection: offline mode, where packets are collected and stored in `pcap` files, providing a checkpoint for reiterating over the workflow but requiring significant storage resources; and online mode, where captured packets are forwarded directly to the feature engineering and ground-truth generation components, without storing a workflow checkpoint.

Feature engineering is a critical step in preparing the data for model training. It involves aggregating packets into flows using a preselected flow key [33], typically composed of the source and destination IP addresses, source and destination port numbers, and transport protocol identifier. Once aggregated, a set of flow features, such as mean packet inter-arrival times, variance of packet sizes, and flow duration, is computed. Basic counters and optional metrics are maintained for each flow entry in the flow cache until its termination, based on the active and inactive timeouts. Feature selection is performed to remove redundant or irrelevant features, based on correlation [65], consistency [51], and breadth-first search [66] methods.

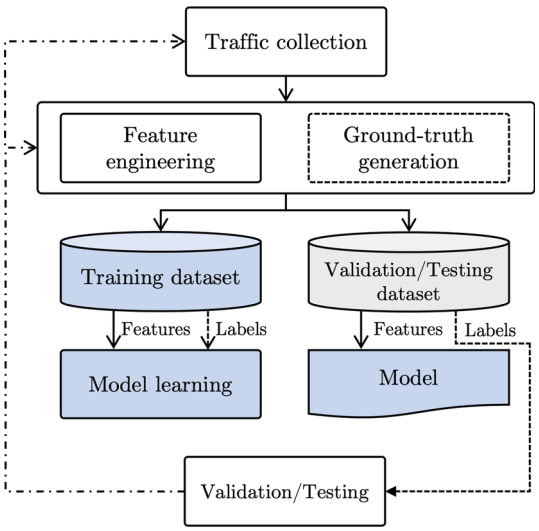


Fig. 1 Workflow for categorizing traffic flows using machine learning

The primary output of feature engineering is a training dataset for the machine learning model, which maximizes its learning accuracy.

Ground-truth generation is mandatory for supervised ML approaches and consists of labeling each flow with its corresponding class (*e.g.*, binary classification for anomaly detection or multi-classification for application identification). Based on the Garbage In, Garbage Out principle, the performance of supervised learning is intrinsically linked to the quality of labels. Thus, obtaining reliable labels is the cornerstone of the overall approach. High-quality label generation is certainly achievable on a synthetic traffic trace, (*i.e.*, a priori knowledge of synthetically generated traffic); nonetheless, it is more complicated for large-scale data collection. To overcome scalability issues, researchers leveraged deep packet inspection (DPI) engines as ground-truth generators [5, 6, 10–12, 16, 18].

Dataset splitting decomposes the collected data into training, validation, and test sets. The training set is leveraged for learning—fitting the parameters of an ML model. In contrast, the validation set is used to tune the parameters of the model. Finally, the test set provides an unbiased measure of model effectiveness as it consists of samples not used to build or tune the model. Validation and test sets are obtained using holdout or k-fold cross-validation and should follow the probability distribution of the training set. Training, validation, and testing sets can also be obtained from different network locations and/or over several time periods. This strategy allows to evaluate the temporal and spatial robustness of a model.

Model learning is the core of an ML approach built to generalize the outcome from training samples. Generalization could be simplified as the ability of a learner to predict previously unseen cases accurately. Researchers applied a broad set of ML algorithms to tackle traffic management challenges [15]. The most commonly used algorithms could be categorized as follows: supervised algorithms (*e.g.*, support vector machines [26], probabilistic neural networks [61], Bayesian neural networks [9], decision trees [8], k-nearest neighbors [56]), and unsupervised algorithms (*e.g.*, k-means [43], AutoClass [25], and DBSCAN [24]). As illustrated in Fig. 1, a supervised algorithm operates on labeled samples, whereas an unsupervised one does not rely on labels to construct a set of clusters. In the latter case, assigning a label to each extracted cluster is not a straightforward process. Hence, semi-supervised methods are also leveraged by researchers in [11, 18] where a mixture of labeled and unlabeled data is used as training input.

Validation and testing are two essential processes in an ML-based workflow. On the one hand, validation is

performed to evaluate the performance of the learner and tune its parameters. On the other hand, testing the fitted model provides an unbiased performance evaluation. Both evaluations are based on metrics such as accuracy, recall, precision, F_1-score, and the area under the precision-recall curve (AUC-PR). Metric selection depends on the nature of the problem (*e.g.*, high recall vs. high precision problem) and dataset characteristics (*e.g.*, highly unbalanced dataset).

Model deployment is the final step of the workflow. In a TFC deployment, packets captured in real-time are organized into flow records (statistics) which are subsequently fed to the classifier developed in the previous steps. The classifier (model) then indicates the likelihood of the membership of the flows. Implementations might differ to meet specific use case requirements. For example, certain implementations may update the model periodically to ensure optimal performance.

3 Data protection in the face of TFC

Network traffic flow measurement, essential for effective monitoring, service, and operation management, poses significant privacy risks [67]. While the data-gathering process inherently captures the content or characteristics of end-user communications, the common practice of using ML techniques for TFC exacerbates these concerns. ML models often require a unified dataset for training, necessitating the collection and consolidation of data from multiple network locations, typically administered by different parties. This process of sharing and transferring datasets to a central node for model training can incite issues relating to data protection and business interests [59].

Even when traffic flow measurement is conducted with legitimate objectives and without the intention to invade user privacy, the resultant dataset could contain personal information, hence falling under the purview of data protection regulations, such as the general data protection regulation (GDPR) of the European Union. While flow records only contain traffic metadata obtained through packet aggregation over a common flow key, thus avoiding full packet payload inspection [33], this metadata can provide insights into the who, when, where, and how much of our communication. Furthermore, IP addresses can serve as practical identifiers that can be linked across services, websites, and devices, representing personal information in a broader sense.

Under GDPR, establishing a lawful basis for managing personal data is paramount. In the case of sharing data with third parties, this would typically require user consent, which is impractical to obtain from all users covered by flow records. This leads to reluctance among organizations to share data for fear of potential GDPR noncompliance penalties. While anonymization techniques could potentially mitigate privacy concerns, they often compromise the usefulness of the data, creating a trade-off between privacy and utility [20, 31, 41].

Organizations also have business interests to protect. Network configuration and policy information can be inferred from flow records, and inadvertently revealing such information can compromise an organization's competitive edge. For instance, traffic patterns can be studied for topology discovery, revealing the structure of the network, or used to infer routing policies and security configurations. Similarly, details such as destination ports can hint at what services are running on the network, and behavior analysis might uncover usage patterns or security incidents. This potential disclosure of strategic information adds another layer of reluctance toward sharing traffic traces with third parties [2].

These factors contribute to the tendency to store and manage measurement data in silos under different administrative controls, hindering the development of superior ML models that could benefit from aggregated information [64]. Network traffic is inherently dynamic, and flow characteristics can vary significantly across networks and over time. Consequently, an ML model developed in one network may not perform as well when deployed in another.

However, FL provides a promising solution to these challenges by facilitating collaborative model development across multiple stakeholders without requiring direct sharing of training datasets. FL operates by sharing local model updates, keeping individual datasets under the control of their respective owners. This approach can alleviate the concerns of organizations in terms of data protection and business interests, thereby creating a more conducive environment for advancing traffic flow classification techniques.

4 Related work

Federated learning is burgeoning as a pivotal technique in collaborative learning tasks, particularly where the privacy preservation of training data is prioritized. This has garnered increasing attention from both academic and industry perspectives, thereby leading to a noticeable surge in related studies. An overview of pertinent work in the domain of network traffic classification is assembled in Table 1. This tabulated compilation allows for an efficient comparison, encapsulating the publication year, the classification type (packet or flow-based), the employed

Table 1 Overview of pertinent work in the domain of network traffic classification. The compilation encapsulates the publication year, the classification type (packet or flow-based), the employed modeling methodology, the aggregation procedure, the dataset used, the control over flow feature computation, and a critique of the application classification accomplished

ID	Year	Class. basis	Scope	Methodology	Aggreg. method	Dataset	Managed feature calculation	Class-wise analysis	IID/non-IID assessed
Majeed et al. [44]	2020	Flow	Internet TC	FL with DNN	FedAvg	CIC-ISCXVPN2016	○	○	○
Zhou [68]	2020	Packet	IIoT TC	FL with CNN	FedAvg	Modbus2014, DNP3-2017	–	◐	●
Mun and Lee [52]	2021	Packet	Internet TC	FL with CNN §	FedAvg	CIC-ISCXVPN2016	–	●	●
Zhu et al. [71]	2021	Packet	Internet TC	FL with DNN §	FedAvg	---	–	●	○
Aceto et al. [3]	2021	Flow	Internet TC	MML with DNN	N/A	CIC-ISCXVPN2016	○	●	○
Zhu et al. [70]	2022	Flow	IoT TC	Fed-SOINN with RBF	FedAvg †	CIT2005, CIC-ISCXVPN2016	○	○	●
Abbasi et al. [1]	2022	Flow	IoT TC	FL with MLP	FedAvg	CIT2005, CIC-IDS2017, CIC-Darknet2020	○	○	●
Wei et al. [63]	2022	Flow	6G Internet TC	FL with CNN	FedAvg	CIC-ISCXTor2016	○	○	○
Jin et al. [34]	2023	Flow‡	Internet TC	FSSL with CNN	FedAvg	QUIC2018	●	◐	○
Guo and Wang [32]	2023	Flow	Internet TC	FL with CNN	FedAvg	QUIC2018, CIC-ISCXVPN2016	◐	○	○
Sun et al. [60]	2023	Flow	Internet TC	FL with CNN§	N/A	USTC-TFC2016	○	○	●
This work		Flow	Internet TC	IFL with DNN	FedAvg	Author-generated	●	●	●

Symbol indication: ● present, ◐ partial, ○ lacking, – not applicable.

†Lightly modified version of the FedAvg algorithm.

‡Utilizes subflows generated by sampling from intraflow per-packet characteristics.

§Uses packet-to-vector image transformation as part of feature space preparation

modeling methodology, the aggregation procedure, the dataset used, the control over flow feature computation, and a critique of the application classification accomplished.

The reviewed works span a range of machine learning models, including deep neural networks (DNNs), convolutional neural networks (CNNs), multilayer perception (MLP), and cooperative communication networks (CCNs), implemented within a federated learning context to classify network traffic. The primary aggregation method employed is the FedAvg algorithm, though variations are used by [70] with a selective gradient communication method and [60] with a sparse parameter difference matrix. Classification tasks target the differentiation of network traffic, from binary to more granular identification of application traffic types. Studies demonstrate varying degrees of success in the federated learning model's performance, with some indicating equivalence to, if not outperforming, centralized and local models, particularly in contexts of privacy preservation and computational efficiency.

The review of recent research in the domain of network traffic classification reveals a range of techniques and approaches. Some works [44, 52, 68] have explored packet-based classification strategies. However, there is a marked preference for flow-based traffic classification as they often provide a more comprehensive picture of network traffic patterns beyond the insights derived by analyzing individual packet headers. They tend to be also more computationally efficient, given that they summarize data from numerous packets.

In the same vein, several papers [52, 60, 71] exploit packet-to-image transformation as a part of feature space preparation. Despite this methodology's potential for providing a visual understanding of packet data, it might not be optimal due to the potential loss of critical information during the transformation process. It may also introduce unnecessary complexity and computational costs. Moreover, the transformation may not preserve the inter-relationships of the data, which are vital for understanding the network traffic context.

Another notable observation drawn from the table is the heavy reliance on datasets generated by the Canadian Institute for Cybersecurity (CIC). These collections offer a broad array of labeled traffic patterns and serve as a fundamental testing ground for a variety of machine learning-based approaches. The widespread usage of CIC datasets[1] can be interpreted as a testament to their credibility within the research community. However, recent concerns surrounding the CIC-IDS2017 [57] dataset and the CICFlowMeter[2] [22, 37], a successor to the ISCXFlowMeter program, cast a shadow over this trust due to observed inconsistencies. There are documented issues pertaining to packet timestamping irregularities, the presence of duplicate packet instances, the mishandling of out-of-order packets, and an alarming lack of labeling for Port Scan attacks in these datasets [36]. Furthermore, the tool utilized for generating all the CIC datasets, CICFlowMeter, has been criticized for improper flow termination upon detecting TCP flags and inaccuracies in flag counter calculations [23]. Flow expiration issues involving active and idle times have also been raised [23].

Although a rectified version of the CICFlowMeter tool has recently been released [23], along with some corrected datasets [42], the body of related works continues to be anchored on the original, flawed datasets. It has been demonstrated that these dataset errors can significantly influence experimental outcomes, thus possibly skewing the results. Apart from one study, all others do not manage the computation of flow features themselves but rely on the prepackaged CSV files containing labeled flow records provided by CIC. This dependence could potentially introduce biases in their findings, as the preprocessed data may not necessarily reflect the specific assumptions or objectives integral to their methodologies. Consequently, this critique invites a reconsideration of the results obtained by these related works in light of these dataset inconsistencies.

With regard to the used datasets, we also noticed inconsistencies and ambiguities in certain works, prompting a call for greater clarity. For instance, Mun and Lee [52] claim that their model can classify 16 application types, including SCP, Spotify, and Netflix categories. This assertion raises questions, given that the CIC-ISCXVPN2016 [22] dataset contains only 14 application categories, with none of the three aforementioned categories directly included. The authors do not explain this discrepancy, leading to uncertainty and potential misinterpretation.

We observed similar deviations in Aceto et al. [3]. They identify 6 traffic types instead of the original 7 and 15 application types as opposed to the original 14 encompassed in the dataset. Although some preprocessing is acknowledged, details regarding the process to determine these labels are not explicitly stated, leaving room for improvement in terms of transparency.

In Zhu et al. [70], there is a lack of clarity about the selection of flow features for evaluation. Given that the CIT2005 [50] and CIC-ISCXVPN2016 datasets comprise a different number of features (248 vs. 23), a clear explanation of the selected features would eliminate potential inconsistencies resulting from divergent dataset structures.

The methodology in Abbasi et al. [1] also raises questions. It is unclear how many samples were selected from the CIT2005 dataset despite the authors' claim of combining all sets. Additionally, the selection process for the subset of application labels remains vague. Moreover, an intriguing discrepancy exists between the consistently lower training times and the considerably larger number of features in the CIT2005 dataset (248 for CIT2005 vs. 83 for CIC-IDS2017 vs. 93 for CIC-Darknet2020 [38]), casting doubts on the credibility of the presented results.

In Guo and Wang [32], there appears to be also a disparity between the number of traffic categories identified by the authors in the CIC-ISCXVPN2016 dataset and the original dataset, a discrepancy that remains unexplained.

A broader perspective also reveals that several works [1, 63, 68, 70] use datasets developed for traditional and data center networks to validate their IoT-centric approaches. These datasets may not adequately represent the typical traffic types found in contemporary IoT networks. There exist more suitable datasets explicitly crafted for IoT contexts, such as the one mentioned in [4]. Surprisingly, the authors did not provide a rationale for selecting these datasets over more contextually appropriate alternatives. This lack of justification warrants a discussion about the adequacy of such selection of datasets and their impact on the presented results.

Next, most studies also seem to disregard the importance of class-wise performance evaluation in multi-class classification, typically presented in the form of a confusion matrix. This is a significant omission because it provides a more granular understanding of the model's performance across different classes, highlighting where

[1] https://www.unb.ca/cic/datasets/index.html.

[2] https://github.com/ahlashkari/CICFlowMeter.

the model is successful and where it fails. Without this information, one might incorrectly assume that a high overall accuracy rate indicates equal performance across all classes, which is rarely the case. In the realm of network traffic classification, where the ability to detect a variety of different traffic types accurately is vital, such analysis becomes particularly crucial.

Lastly, half of the works mentioned above do not explicitly state whether they investigated different data distribution conditions; this is a glaring omission, as a TFC mechanism could produce significantly better results in an IID scenario compared to a non-IID one. In this work, we design our experiments around four different data distribution scenarios, starting from IID and progressing toward non-IID conditions with increasing complexity; we report and compare performance figures for each of the conditions.

Amidst these observations, another trend surfaces that could inadvertently trivialize the problem of application-type classification, thereby questioning the necessity of employing machine learning. The details surrounding data preparation, feature engineering, and feature selection are often inadequately documented. Consequently, it remains ambiguous whether port numbers, typically utilized in generating flow records as part of the flow key, were incorporated into the feature space. Traditional network traffic classification associates specific port numbers with certain applications (e.g., SSH, BitTorrent, NTP, RDP), implying that these application types could be distinguished merely by scrutinizing the port numbers, thereby rendering the application of machine learning redundant. Therefore, the inclusion of port numbers within the feature space could introduce significant bias to the outcomes, thereby potentially inflating the perceived effectiveness of the model. In light of this, we emphasize the need for a more transparent and detailed account of feature selection, highlighting the impact of feature choices on the classification task and the generalization ability of the models.

In contrast to the aforementioned studies, our work consciously addresses their limitations, thereby ensuring full control over the traffic generation process. The network traffic flow characteristics in our study are calculated using NFStream [7], a tool we developed recently. This tool effectively overcomes the identified shortcomings of the CICFlowMeter, thereby eliminating the potential bias that could be introduced into the data it generates. Furthermore, we curated our own dataset for this study. We did so by measuring the network traffic flows at the uplink of the university's student dormitory network. We place a strong emphasis on describing the complete process from packet capture to feature space preparation, which serves as input for our models. This in-depth explanation bolsters the interpretability and reproducibility of our research, promoting an open and transparent approach to scientific inquiry. Recognizing the significant value of multi-class classification performance evaluation, we also provide an analysis of the effectiveness of our approach across multiple classes.

5 Methodology

This study aims to investigate the applicability and efficiency of FL in the context of traffic flow classification. Our methodology employs an incremental FL approach to network traffic flow classification, leveraging network traffic flow measurement while performing privacy-preserving machine learning. FL addresses the current challenges in sharing high-quality, labeled datasets due to data protection policies, thus overcoming barriers to effective TFC.

5.1 Flow measurement

The initial stage of the methodology involves the measurement and management of network traffic flows. This is achieved by aggregating packets that share a common key, as illustrated in Fig. 2. Flow features, derived from the IP, TCP, and UDP packet headers, form the primary basis for subsequent analysis and classification. Such features include statistical summaries (minimum, maximum, mean, and standard deviation) of packet lengths and inter-arrival times.

To standardize this flow metering across clients, we have adopted the NFStream framework from our previous work [7]. This open-source framework stores the measured flow records in pandas DataFrames and CSV files, serving as the input for local model learning. CSV and pandas DataFrames were chosen for their widespread acceptance in ML-related tasks.

5.2 Privacy-preserving machine learning

The methodology then proceeds to the application of FL to solve the learning task collaboratively, coordinated by a logically centralized server, but without the need for direct data exchange, as shown in Fig. 2. The primary computation at the client level includes model training and evaluation, while the server manages the global computation, orchestrating client activities such as client selection, configuration, gradient aggregation, federated global model evaluation, and client model update.

To facilitate this complex interaction between local and global computations, we have adopted the Flower framework [13]. This open-source FL framework enables a

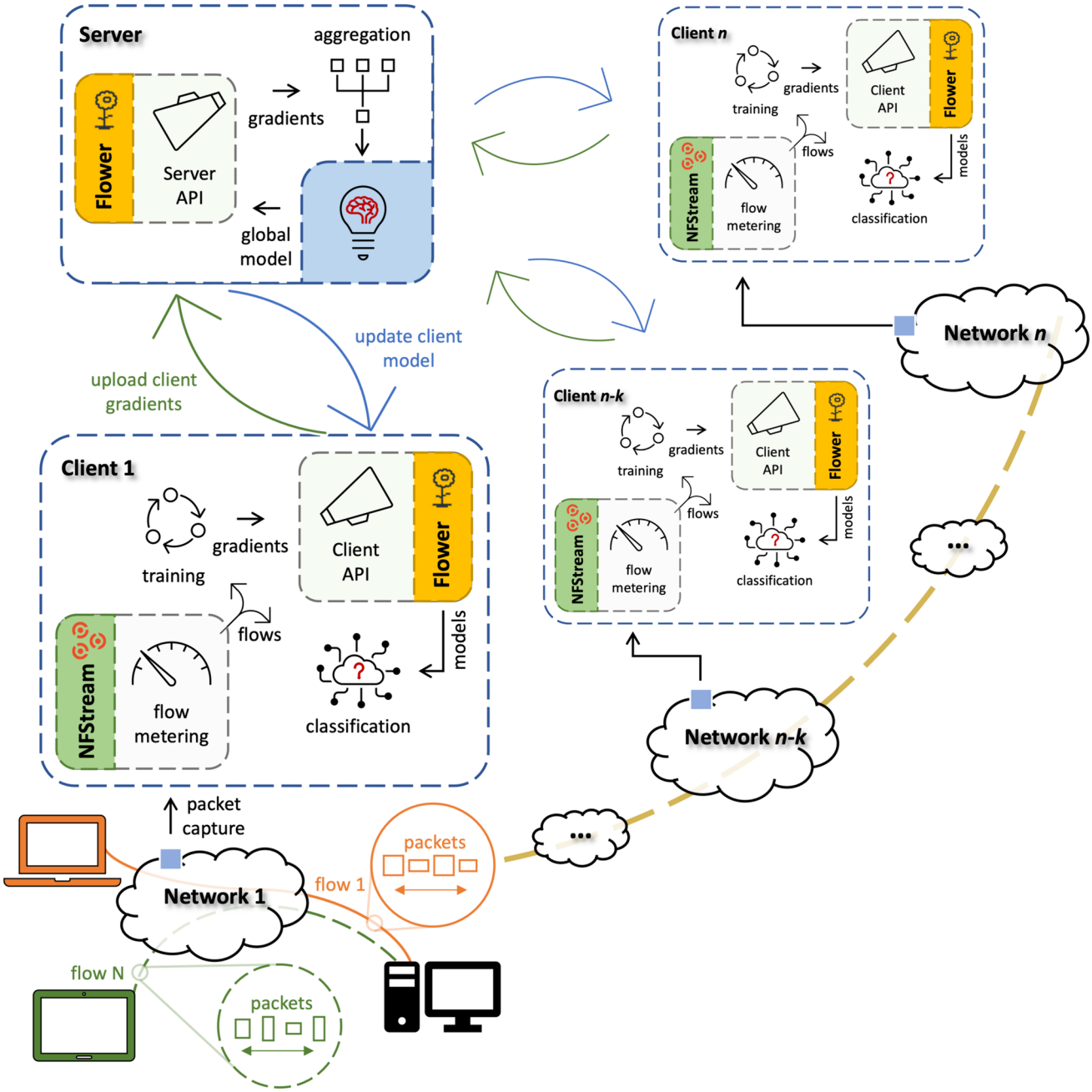


Fig. 2 High-level architecture of the methodology. Records that carry traffic flow information collected by the clients in each network via NFStream [7] serve as input for TFC. Eligible participating clients first connect to the server, download the current model, and learn about the training strategy. Then, the clients start computing the training gradients using local data. The server systematically and periodically collects the gradients from the clients to update the current model via Flower [13]. Subsequently, the participating entities update their model. Repeating these steps realizes the cross-silo learning task through a loose federation of nodes participating in the learning process coordinated by the central server. The data are observed locally and remains put without any other clients or the server having access to it

flexible, extendable, and language-, communication-, and device-agnostic solution. Flower aims to streamline the transition of existing ML pipelines into an FL setup and fosters realistic-scale research.

5.3 Workflow

The workflow for our incremental FL-based methodology for traffic flow classification involves a series of interactions between the server and the client components.

5.3.1 Server initialization

The FL process mandates the server to be fully operational before beginning. This involves setting up secure communication channels with the client devices. In parallel, the server undergoes global model initialization, which includes setting up a model with a pre-defined architecture and specific parameters.

Notably, this phase does not necessitate the use of any dataset, reflecting the server's role in FL as a coordinator rather than a direct learner. Instead, its function is to compile and aggregate updates—such as model weights or gradients—submitted by various clients, each of which trains the model locally on its unique (possibly self-curated) dataset. Following these initial steps, the server invokes the Flower framework to facilitate the management of the FL process, thereby ensuring efficient coordination and seamless integration of updates from all participating clients.

5.3.2 Client initialization

Each client within the federated network initiates the necessary processes for active participation. This process starts with the establishment of a secure communication channel with the server, consistent with the server-side configuration. Subsequently, clients initialize their local models, typically aligning with the initial model architecture and parameters set by the server. This standardization is crucial for uniform starting conditions across clients, facilitating effective model update aggregation on the server side. In our FL framework, the initial model setup can be directly defined and distributed by the server at the training's outset, ensuring uniformity across all client models.

Upon completing these initial steps, clients signal their readiness for training to the server, marking the commencement of the federated learning cycle.

In our framework, the input dataset for local training can come from diverse sources, such as live packet captures, PCAP files, or structured datasets (e.g., CSV or Parquet formats). To ensure consistency and compatibility throughout the federated learning process, clients are required to format their local datasets according to the NFStream feature syntax[3].

An essential aspect of data preparation involves unifying target label encoding across all participating clients, crucial for the consistency and interpretability of the model's outputs. In our design, this uniformity is ensured by encoding the target labels before employing our unique dataset chunking methodology, detailed in Sect. 6.4. However, the comprehensive examination of encoding strategies and their implications in varied deployment contexts is beyond the scope of this current work and earmarked for future investigation.

Each client in our FL experiment design represents a distinct network segment and is continually exposed to new, unseen network traffic flows. This exposure is facilitated through our unique dataset chunking methodology, as outlined in Sect. 6.4, which enables an incremental learning process. This methodology allows local models to be progressively refined with each new data iteration, enhancing the realism of our FL setup and providing valuable insights into real-world traffic analysis applications. By adopting this structured approach, we not only aid in maintaining label consistency across the federated network but also enhance the overall efficiency and applicability of the learning process, embodying the principles of incremental learning in a federated context.

5.3.3 Federated learning process

The FL process in our methodology utilizes federated averaging (FedAvg) [45], a privacy-preserving machine learning algorithm widely recognized for enabling the collaborative development of a unified global model in distributed learning environments.

The selection of clients for model training and the evaluation of the global model are fully configurable within our adapted framework. While client selection is typically random when multiple clients are available, our configuration does not preclude the engagement of all available clients if deemed necessary or practical. In our specific experiment design, we have opted for the involvement of all five clients, facilitating a fair comparison to the idealistic CL case.

In each learning round, clients initiate the process by downloading the latest iteration of the global model, which entails retrieving the current weights (parameters) from the server. These weights are synthesized from the cumulative learning from all clients that have previously contributed to the model. Following this, clients proceed to locally train an instance of this global model using their private datasets. Upon the completion of an FL round, the server aggregates the model updates—specifically, the gradients—from all participating clients. The server then updates the global model by averaging these collected parameters, thereby finalizing the cycle and setting the stage for the subsequent FL round.

5.3.4 Model evaluation

Typically, after developing a new global model, the server may send it to a randomly selected subset of clients (or to

[3] https://www.nfstream.org/docs/api.

all clients, depending on the configuration) for evaluation. Following this evaluation phase, the server aggregates the performance metrics returned by the clients.

In contrast, our experiment design opts for validation of the global models directly on the server side. We have established dedicated validation sets for each federated round, as outlined in Sect. 6.4. Clients employ these distinct validation sets to evaluate their model's performance following every local training epoch within each round. This evaluation mechanism is implemented via a Python Keras Callback, facilitating real-time performance assessment during the training phases. In parallel, the server leverages these same validation sets specific to the round to assess the efficacy of the global model after each round's aggregation phase.

This approach allows for a detailed comparison between the local client models and the server's aggregated global model. By employing this method, we gain valuable insights into the progression of individual client models relative to the overall advancements of the global model, enhancing our understanding of FL dynamics.

5.3.5 Iteration and adaptability

Once the performance of the new model has been evaluated, the system is ready to commence the subsequent FL round. The workflow is designed to accommodate an arbitrary number of rounds, offering significant flexibility. This design allows clients to switch modes and join or exit the federated network as required, thereby ensuring the system's adaptability to dynamic operational environments. This iterative process facilitates continuous learning and model refinement in response to changing data landscapes and client configurations.

6 Experiment design

Our evaluation consists of a comparative analysis of the incremental FL-based approach versus an idealistic CL mechanism. To achieve a robust evaluation of network traffic flow classification using FL, our experiment design incorporates NFStream v6.5.4 for flow measurement and Flower v1.4.0 for federated network communication, following the architecture illustrated in Fig. 2. In this section, we delve into the key aspects of our experiment design, starting with dataset preparation and the establishment of ground truth, followed by our approach to training setup and model configuration.

6.1 Dataset preparation and ground-truth establishment

We established a ground truth to assess the comparative performance of FL and CL approaches based on the flow measurements from a university's student dormitory network. Flow application labeling was performed using the nDPI library integrated within NFStream, ensuring precise identification and categorization of flow applications.

During the generation of flow records, priority was given to the effective management of flow expiration. We set the idle timeout to 120 s and the active timeout to 1800 s, with a specific emphasis on the expiration of TCP flows. Our TCP flow expiration policy is designed to remove expired flows based on specific conditions involving TCP flags. A flow immediately expires upon the detection of a TCP RST flag, ensuring rapid termination of the flow in cases of abrupt connection closure. For the FIN flag, the policy is more nuanced: a flow expires only after the detection of two bidirectional FIN packets, which must be followed by an ACK packet. This ensures that the flow is observed through the complete process of a graceful TCP connection termination in accordance with TCP flow termination conventions. In addition, it is important to note that the byte counts reported by NFStream in our setup were configured at the link layer level, ensuring that the reported sizes reflect the entire packet size, including link layer headers, rather than just the IP payload size.

The data preparation process involved several pivotal steps to ensure the reliability and accuracy of our study. We started by eliminating all flows where nDPI did not assign the label with the highest level of confidence. This precaution was taken to enhance the accuracy of the identified classes while reducing the possibility of misclassification, which could otherwise distort the analysis results.

In addition, we dismissed flows with a count below a certain packet threshold. Such flows might represent ephemeral, sporadic or faulty connections, contributing insignificant or unreliable information to our dataset. Setting a minimum packet count requirement ensured that the flows included in our study were substantial enough to yield meaningful data. In our data preparation, we defined this minimum number as 10 packets.

The filtered dataset consisted of 451 distinct applications. For the purpose of this study, however, we decided to concentrate exclusively on a subset of these applications to streamline interpretation and management. The selected applications included 'TLS.TikTok,' 'QUIC.YouTube,' 'BitTorrent,' 'TLS.Facebook,' 'HTTP,' 'Discord,' 'STUN,' 'QUIC.Instagram,' 'RDP,' and 'WhatsApp.' This selection ensured a balanced representation of both

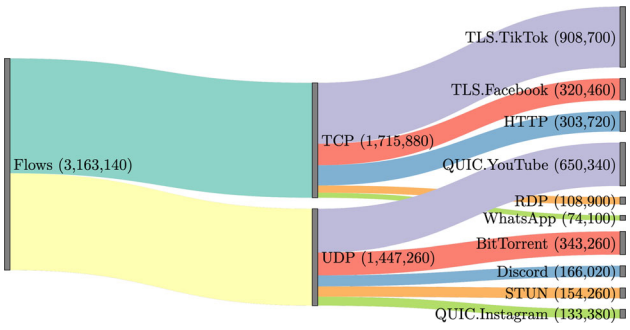


Fig. 3 Distribution of network traffic across different protocols and the selected ten applications

applications with distinct patterns and popular applications frequently used by students.

The splitting process to prepare the dataset for the IID scenario was executed in a stratified manner. This ensured that the samples from each application were uniformly distributed among the chunks. However, in some instances, data cannot be evenly divided due to an insufficient number of samples for certain applications or when the number of samples for some applications is not a multiple of the number of chunks. In our study, this discrepancy resulted in a residual amount of 322 samples. These residual samples were removed from the dataset to maintain consistency and ensure a fair allocation.

Our process concluded with a final dataset comprising 3, 163, 140 flows, furnishing a robust foundation for our subsequent analysis. The distribution of this dataset across TCP and UDP, as well as among the ten chosen application types, is depicted in Fig. 3. The specific distribution for each application type is as follows: 'TLS.TikTok,' 28.73% 'QUIC.YouTube,' 20.56% 'BitTorrent,' 10.85% 'TLS.FaceBook,' 10.13% 'HTTP,' 9.60% 'Discord,' 5.25% 'STUN,' 4.88% 'QUIC.Instagram,' 4.22% 'RDP,' 3.44% and 'WhatsApp' 2.34%.

To uphold the principles of reproducibility and open scientific collaboration, we are sharing this dataset[4].

6.2 Data feature reduction and selection

Our final dataset initially encompassed a diverse range of features, including both numerical and categorical types. Among the categorical features, we specifically retained the application names, which served as the target for our model training. These application names were processed using label encoding to convert them into a numerical format.

However, to reduce model complexity and mitigate the risk of overfitting, other categorical features such as IP addresses and client and server fingerprints were removed. High-cardinality categorical features can often introduce noise and distort the learning process of the model. By focusing on numerical and directly quantifiable features alongside the label-encoded application names, we refined our dataset to 65 features. This dataset included forward (src-to-dst), backward (dst-to-src), and bidirectional measures.

To streamline the feature space further for more efficient learning, we employed the Extra Trees [30] feature selection method. This approach assigns a significance score to each feature. By setting a threshold of 0.02, we identified and selected the most impactful features, narrowing them down to 14 key flow features. These selected features include the minimum, mean, standard deviation, and maximum packet sizes (PS) in both the bidirectional and backward directions. In the forward direction, we included the standard deviation and maximum packet sizes. Additionally, our reduced feature space also included time-related features, specifically the bidirectional standard deviation and maximum of packet inter-arrival times (PIAT). Furthermore, the TCP flags were considered by including the RST (reset) packets measured bidirectionally. This methodical approach led to a compact yet comprehensive feature set, ideally suited for an efficient and robust analysis.

6.3 Neural network architecture

Our experiment for both FL and CL employed a feed-forward neural network (FNN) with a conventional architecture comprising an input layer, two hidden layers, and an output layer. The network was structured as follows:

- The input layer was designed with 14 neurons, aligning with the number of features in our dataset.
- Hidden layer 1 contained a calculated number of neurons: two-thirds the size of the input layer (rounded to the nearest whole number, resulting in 9 neurons) plus the size of the output layer (10 neurons). This configuration yielded 19 neurons for the first hidden layer.
- Hidden layer 2 was configured to match the size of the input layer, comprising 14 neurons.
- The output layer was equipped with 10 neurons, corresponding to the number of classes in our classification task.

For both hidden layers, we utilized the rectified linear unit (ReLU) activation function. ReLU is favored for its ability to introduce nonlinearity, enhancing the model's capacity to learn complex patterns without significant computational complexity. The output layer employed the Softmax activation function, which is effective in multi-class

[4] https://github.com/FlowFrontiers/IFLforTFC/blob/main/datasets/dataset.parquet.

classification tasks. Softmax converts the output layer's raw scores (logits) into probabilities that sum to one, ideal for classifying into multiple categories.

The model was compiled using the Adam optimizer, renowned for its efficiency in handling sparse gradients and its adaptability with varying learning rates. We specifically chose a learning rate of 0.0001, a relatively low value, to ensure gradual and more stable convergence during training. This fine-tuned learning rate helps in navigating the model through complex landscapes of high-dimensional data, aiming to find the optimal set of weights with minimal risk of overshooting the minimum loss.

6.4 Data handling strategy in FL and CL setups

Our experiment was structured to simulate the dynamic and evolving nature of network traffic flow measurements involving five clients in a cross-silo FL setup. To effectively emulate real-world network traffic, we devised a unique dataset chunk distribution strategy. Each client participated in 10 rounds of FL, and for each round, we allocated specific data chunks for training and validation:

- *Training Dataset Chunks:* We provided each client with ten distinct training dataset chunks, one for each round, containing flows representing different network traffic segments. These chunks were exclusive to each client and each round, with no overlap of flow samples between chunks, and were used only once. This approach was chosen to mimic the progressive nature of network traffic flow measurements.
- *Validation Dataset Chunks:* Alongside, for each round of FL, we reserved a separate validation dataset chunk. These differed from the training chunks and were shared among all clients and the server within the specific round. This ensured that validation was consistent and comprehensive.

In FL, each client conducted local training over 10 epochs for each round using a batch size of 32. The clients performed validation facilitated by Keras Callbacks after each of the 10 epochs of local training using the dedicated chunk. This setup enabled us to assess how well the local models generalized to data outside their own distribution (client-specific data). After the completion of each round, the server aggregated the local model gradients using FedAvg and validated the updated global model using the same dedicated validation chunk. This end-of-round validation assessed the performance of the updated global model.

This approach introduces a unique design setup in federated learning that is particularly suited for environments with dynamic and ever-evolving data, such as network traffic. Unlike most existing methods where data might be reused or iteratively cycled through FL rounds, our methodology ensures that each data sample (flow) is used only once for training (excluding repetitions within epochs) or validation. This exclusive utilization of data samples embodies **an incremental learning process** wherein models are systematically updated and refined with the influx of new, unique data. This strategy not only adds a layer of realism to our FL model but also provides more accurate insights into how these models would perform in real-world network traffic analysis. Consequently, our design represents a significant advancement, effectively narrowing the gap between theoretical machine learning constructs and their practical applications in real-world settings.

For CL, we aimed to mirror the validation proportion seen in the FL setup. Our total dataset consisted of 3,163,140 samples. Considering the division of the dataset into 60 chunks (50 for training and 10 for validation, across all clients and rounds), each chunk contained approximately 52,719 samples. Therefore, to have a comparable validation set size in CL as in FL, we calculated a train-test split of approximately 16.67%. In CL, we maintained a consistent total number of training epochs (100) to align with the cumulative epochs in FL (10 epochs across 10 rounds), and adhered to a training batch size of 32.

6.5 IID and non-IID data partitioning and assignment

Traditional machine learning algorithms typically operate under the presumption that data is IID, implying that all samples stem from the same distribution, and each sample is independent from others. This IID assumption holds significant relevance in FL, which involves learning from distributed data sources, also referred to as 'participants.' Considering the potential for each participant's data to exhibit a non-IID distribution, it is critical to verify the robustness and efficiency of our learning algorithm under both IID and non-IID conditions.

In our study, we therefore considered a variety of conditions contributing to the creation of different IID and non-IID scenarios. From Table 2, we find that by carefully partitioning and assigning data according to these conditions, we craft scenarios that test the limits of federated learning, from idealized conditions where data are evenly distributed, to complex situations that challenge the algorithms with varied data distributions among the participants. The description of these scenarios is as follows:

1. *IID*: In the IID scenario, every data chunk has an identical sample size, ensuring that each participant receives an equal number of samples. This mirrors the

assumption that data is evenly distributed among participants.

2. *non-IID-A*: For the non-IID-A condition, chunks differ in sample size, reflecting a more realistic situation where some participants may have more data than others.
3. *non-IID-B*: In the non-IID-B scenario, while each chunk has an identical application distribution, the sample sizes differ. This setup allows us to study the impact of different amounts of data per participant while maintaining consistent application diversity.
4. *non-IID-C*: The non-IID-C scenario is characterized by identical application counts across chunks but varying application distribution and sample sizes. This represents a common real-world challenge where the number of samples per application may be consistent, but the overall composition of the data can vary significantly.

Each of these scenarios presents unique challenges for federated learning algorithms, and assessing performance under each condition provides insight into their robustness and adaptability.

The stacked bar plots in Fig. 4 illustrate the per-chunk application type distribution across the four scenarios. For the IID scenario, as shown in Fig. 4a, we ensured a uniform distribution-identical to that in the unpartitioned dataset-within each chunk, thereby fulfilling the 'identically distributed' requirement of the IID condition. As a result, each chunk featured a proportionate representation of every application type.

In contrast, for the non-IID scenarios A–C (refer to Fig. 4b–d), we preserved a uniform distribution of application types in the validation chunks (allocated every sixth chunk, as indicated in the plots), mirroring the distribution found in the IID scenario chunks. This was intended to ensure the representation of each application class within the validation chunks, providing a robust basis for evaluating the global model's adaptability and generalizability. Following this, we systematically distributed the remaining samples to the training chunks using three distinct approaches, incrementally increasing the degree of non-IIDness as detailed in Table 2. This allowed us to explore scenarios in which some clients may not encounter certain application types during local training, offering a methodical examination of the models under diverse distribution conditions and yielding insight into the practicality and efficacy of FL in authentic environments.

7 Results

This section offers a comprehensive analysis of the results derived from our experimental evaluation. To enhance comprehensibility, we have divided our findings into several categories, each focusing on a different facet of the performance exhibited by our FL approach compared to CL.

We begin with an evaluation of model performance, focusing on validation accuracy and loss. Although training accuracy and loss are commonly assessed, we emphasize validation metrics because the global model, which is an aggregation of local model gradients, does not provide training accuracy and loss. This precludes a direct comparison between local and global models, hence our focus on validation metrics.

Next, we examine a range of well-established machine learning performance indicators: precision, recall, accuracy, and the F_1-score. Precision measures the fraction of samples in a class that were classified correctly. Recall calculates the proportion of actual positives that were classified correctly (accounting for false negatives). F_1-score is defined as the harmonic mean of precision and recall, balancing both previous characteristics. Finally, accuracy establishes the number of correct classifications over all attempts. These metrics provide a comprehensive view of our approach's effectiveness. For both FL and CL, we computed multi-class versions of these metrics using the macro-average method. This method treats each class with equal importance, which is essential given our dataset's class imbalances. While macro averaging may not yield the highest overall accuracy, it promotes fairness and ensures high performance across all classes.

Finally, we present normalized confusion matrices. These matrices offer detailed insights into the classification accuracy for different application types, highlighting the models' discriminative abilities. Together, these evaluations contribute to a thorough understanding of our FL approach's performance.

7.1 Validation accuracy and loss

Figure 5 compares the validation accuracy and loss of the CL model with that of the FL models under the four distinct data distribution scenarios. This analysis is based on data collected over 10 FL rounds. Each round includes 10 epochs of local training by the clients, followed by a server-side aggregation of gradients, denoted as FedAvg in the plots (bottom x axis). For better clarity in the visual representation, the plots for the FL models depict results from every second epoch (bottom x axis), whereas the CL

Table 2 Overview of IID and non-IID data conditions in the designed FL scenarios

Condition	IID	non-IID-A	non-IID-B	non-IID-C
Each chunk has identical sample size	✓	×	×	×
Each chunk has identical application distribution	✓	✓	×	×
Each chunk has identical application count	✓	✓	✓	×

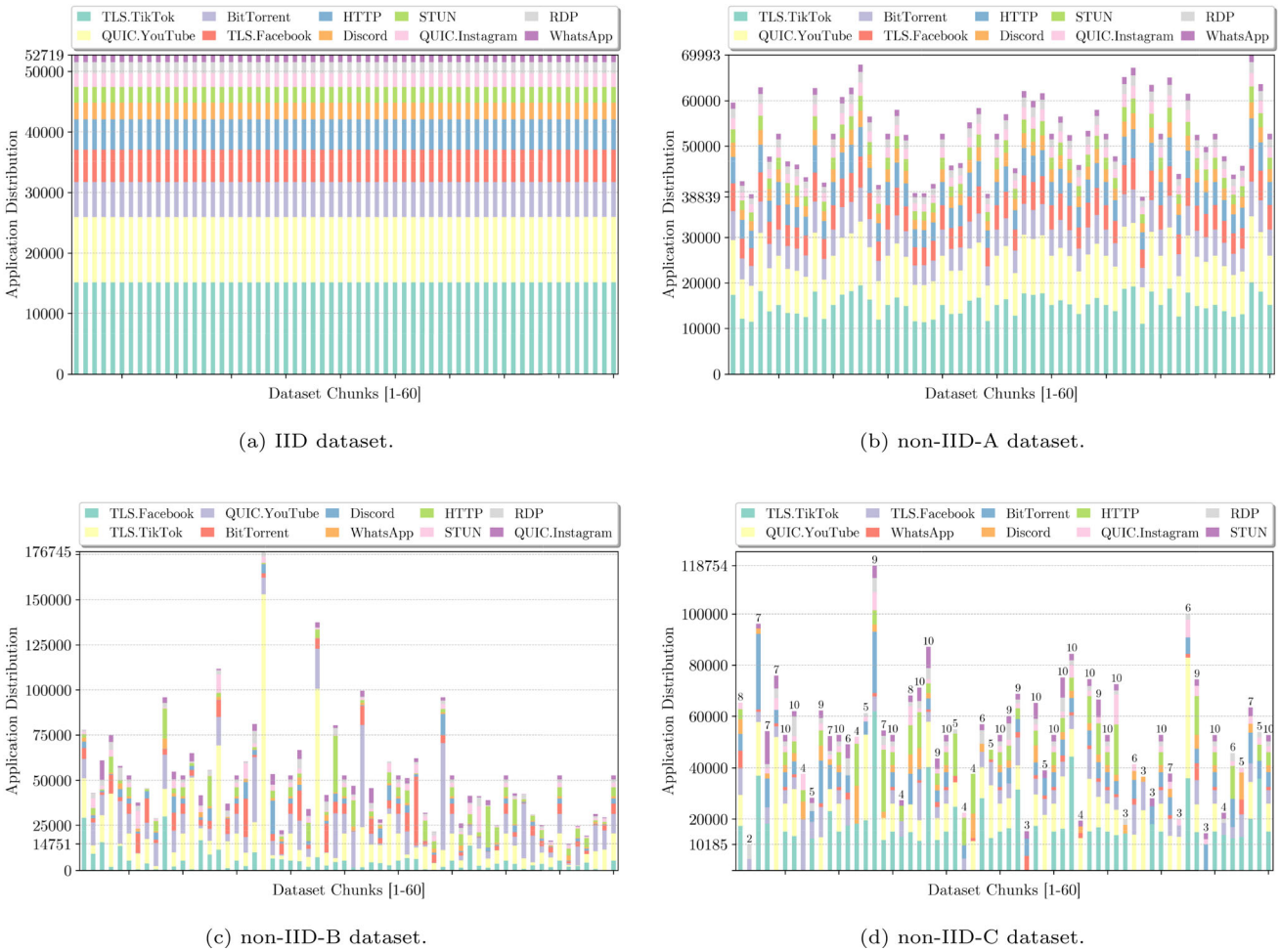


Fig. 4 Stacked bar plots representing the distribution of application types within the dataset chunks under IID and non-IID conditions. The *x* axis enumerates the dataset chunks from 1 to 60, and the *y* axis quantifies the counts of each application type. Distinct colors in the bar plots correspond to different application types, as identified in the legend. For non-IID scenarios, validation chunks (every sixth chunk) feature a uniform application type distribution to match that of the IID chunks, ensuring consistency for model validation

model's results are presented for every fifth epoch (top *x* axis).

7.1.1 IID scenario

The comparison of validation accuracy between the FL models in the IID scenario and the CL model is depicted in Fig. 5a. This comparison reveals a wide range of accuracy levels among different clients in the FL model, highlighting the variability in local model performances. Despite these differences, the aggregated global FL model shows a general trend of increasing accuracy across the rounds, albeit with some fluctuations.

The most significant of these fluctuations occurs in round 6, where the FL model experiences a temporary drop in validation accuracy, followed by a recovery and subsequent improvement. Key factors contributing to this fluctuation include the interplay of the distinct characteristics of Client 2's training set compared to the validation set used in round 6, potential temporal overfitting due to

intensive learning on specific, non-generalizable patterns, the inherent stochasticity stemming from mini-batch gradient descent, and the adaptive learning rate mechanism of the Adam optimizer. These elements collectively contribute to the complex, nonlinear, and somewhat unpredictable nature of neural network training. Such a dynamic process can lead models through periods of suboptimal performance before they achieve improved generalization.

Notably, the general trend of the FL model, while not exceeding, remains close to the CL model's performance, underscoring the potential of FL's collaborative training to achieve results comparable to CL. The oscillations in accuracy among individual FL clients highlight the variability of local data, yet the global FL model's ability to maintain close proximity to the CL model's accuracy suggests effective incremental learning integration. Quantitatively, the FL model's validation accuracy ranged from a minimum of 90.38% in round 1 to a maximum of 96.65% in round 10. This is in comparison to the CL model, which achieved an accuracy of 98.50% at its final epoch. This showcases the feasibility of FL in scenarios, where data is not only distributed evenly but also continuously evolving. Such adaptability makes FL particularly relevant in contexts where data privacy is crucial and data sources are decentralized, confirming its applicability in preserving data integrity while still benefiting from collaborative learning dynamics.

The validation loss showed in Fig. 5b aligns with the insights gleaned from the accuracy analysis, showcasing the FL model's trend toward improved generalization as it advances through the federated rounds. While there are episodes of increased loss corresponding to specific epochs, particularly during round 6 as noted above, these do not disrupt the overall decreasing trend.

Despite the variability in loss among individual FL clients, the FL model as a whole maintains a downward trajectory in validation loss across the rounds. It approaches the level of loss exhibited by the CL model, indicative of the robust nature of the FL framework. This ability to recover from loss spikes and mirror the CL model's loss profile underscores the effectiveness of incremental collaborative training within the FL paradigm. Although the FL model does not outperform the CL model in terms of lower loss, its consistent proximity to the CL model's performance suggests that FL is well-poised to achieve degrees of generalization and stability comparable to centralized approaches.

7.1.2 Non-IID-A scenario

The non-IID-A scenario, characterized by variable sample sizes across chunks while preserving the distribution of application types, presents a unique challenge that surprisingly results in a more stable learning curve for the FL model, as evidenced by validation accuracy (Fig. 5c) and loss (Fig. 5d).

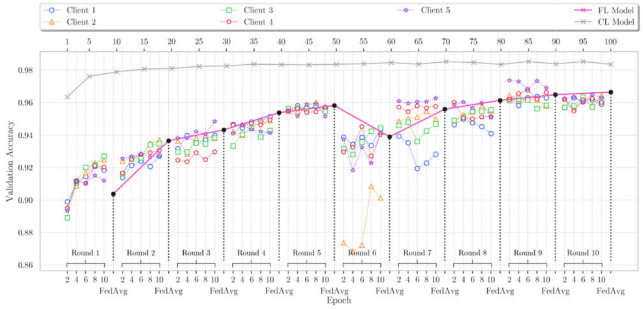
Unlike the IID scenario, where larger accuracy fluctuations were observed, the non-IID-A scenario exhibits a smoother improvement trajectory in validation accuracy across federated rounds. This stems from a marked reduction in variability among clients, especially noticeable in later rounds. A similar pattern of stability is evident in the validation loss.

This unexpected stability, with significant fluctuations like the dip observed in round 6 of the IID scenario notably absent, suggests a potentially beneficial impact of non-IID conditions on FL dynamics. The methodical approach to generating non-IID chunks, employing systematic sampling techniques with tools such as Python's Pandas |sample()| function and NumPy's |random.choice()|, contributed to a more balanced and representative distribution of samples across clients. This approach likely facilitated a more effective aggregation of local models into the global model, thereby enhancing the overall efficiency of the learning process. Indeed, our re-evaluation of the IID scenario—where the dataset was first shuffled using the Pandas |shuffle()| function prior to chunking—has confirmed that such mechanisms can considerably influence the outcomes. The results, as demonstrated in [28], show improved performance across all metrics. Over the course of 10 federated learning rounds, the model achieved a minimum validation accuracy of 91.39% in round 1, with a peak of 97.09% in round 10.

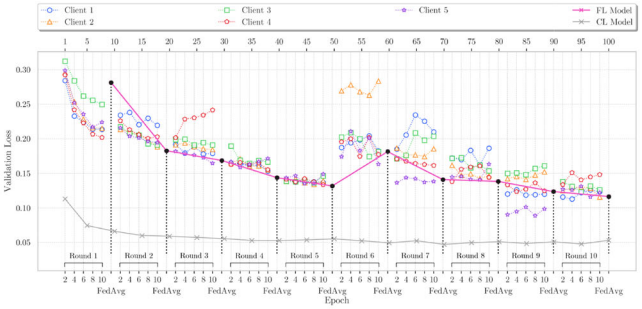
The improvement in performance under the non-IID-A scenario could be attributed to the way FL algorithms handle data heterogeneity. In an IID setting, the assumption of uniform data distribution might lead to overfitting specific patterns that are not universally representative, thus causing fluctuations in model performance. Conversely, the non-IID-A scenario's variable sample sizes across chunks introduce a form of regularization, forcing the model to learn from a broader spectrum of data characteristics. This can lead to a more generalized model that performs better on unseen data, as reflected in the validation metrics.

Moreover, the absence of significant dips in model performance suggests that the FL model's learning process is less susceptible to the overfitting of non-generalizable patterns, a common pitfall in the IID scenario. This suggests that non-IID data, when carefully managed, can potentially enhance the model's ability to generalize.

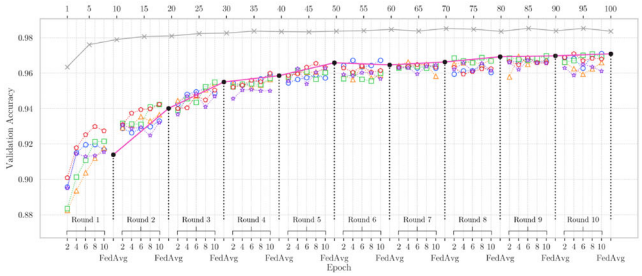
Overall, the global FL model demonstrates a consistent performance improvement over the federated rounds, also in the non-IID-A scenario, echoing the trends observed in the IID scenario. This consistency confirms the model's



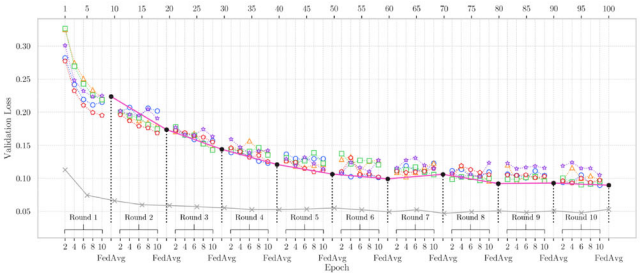
(a) IID scenario validation accuracy.



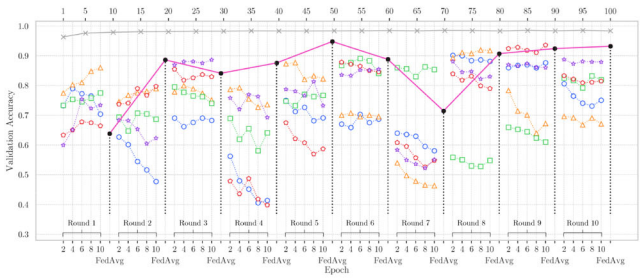
(b) IID scenario validation loss.



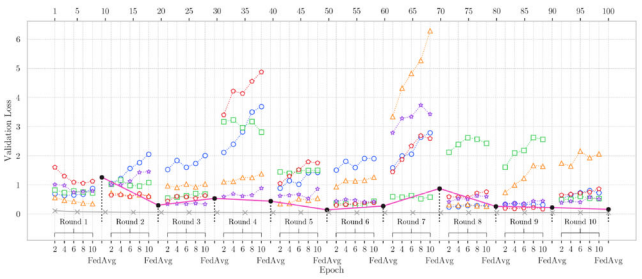
(c) non-IID-A scenario validation accuracy.



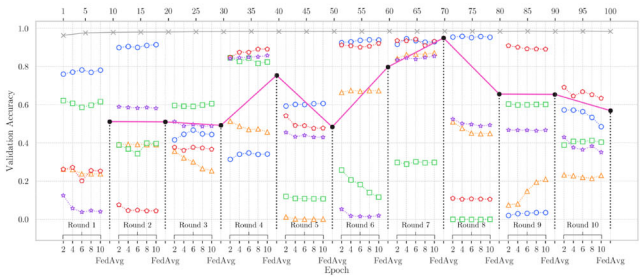
(d) non-IID-A scenario validation loss.



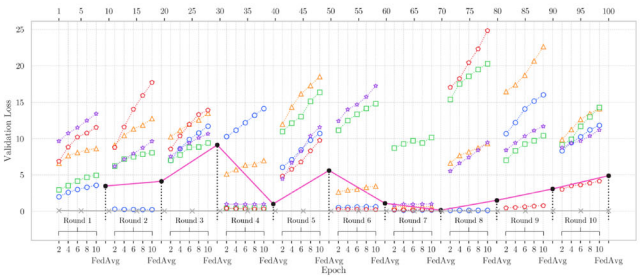
(e) non-IID-B scenario validation accuracy.



(f) non-IID-B scenario validation loss.



(g) non-IID-C scenario validation accuracy.



(h) non-IID-C scenario validation loss.

Fig. 5 Comparison of the FL model's performance against the CL model across four distinct scenarios: IID, non-IID-A, non-IID-B, and non-IID-C. **a**, **c**, **e**, **g** Depict the validation accuracy, and **b**, **d**, **f**, and **h** showcase the validation loss for each scenario, highlighting the variability, stability, and generalization capabilities of the FL model in diverse data conditions. Each scenario illustrates the impact of data distribution on the learning dynamics, with the FL model's performance measured against the consistent benchmark of the CL model

capability to adapt and learn effectively from decentralized and diverse datasets.

7.1.3 Non-IID-B scenario

In the non-IID-B scenario, our analysis underscores the pronounced variability in client performance, a direct consequence of the challenging non-IID conditions where each chunk brings variable sample sizes and distributions

across application types. Validation accuracy (Fig. 5e) and loss (Fig. 5f) vividly illustrate this variability, highlighting the diverse learning trajectories of individual clients. Despite these challenges, the global FL model demonstrates remarkable adaptability and resilience. Notable fluctuations in performance, particularly observed in rounds 3 and 7, demonstrate the FL model's robustness in navigating through the complexities of heterogeneous data distributions.

The journey of the FL model in this scenario is characterized by its ability to gradually bridge the performance variability among clients, showcasing a steady convergence toward improved accuracy and reduced loss. This progression is pivotal, as it illustrates the model's capacity to synthesize disparate local learnings into a cohesive and effective global model. Impressively, despite the initial disparities and intermittent fluctuations, the FL model's performance eventually aligns closely with that of the CL model, achieving a minimum validation accuracy of 63.81% in round 1 and peaking at 94.79% in round 5. This performance sheds light on the potential of federated learning to leverage decentralized, diverse datasets effectively and achieving outcomes comparable to centralized approaches, even under the stringent conditions posed by the non-IID-B scenario.

This scenario reaffirms the viability of FL as a robust framework for collaborative learning, capable of overcoming the intrinsic challenges posed by non-IID data distributions and achieving parity with centralized learning benchmarks.

7.1.4 Non-IID-C scenario

For the non-IID-C scenario, as evidenced by Fig. 5 and h, we observe the FL model clearly underperforming compared to CL. This scenario introduces an additional layer of complexity, with some application types being absent from certain chunks, challenging the FL model's ability to learn effectively across all application types.

A particularly noteworthy observation is the significant performance spike observed in the FL model in round 7, where it reached a maximum validation accuracy of 94.96%. This sudden increase suggests that the specific data distribution in this round may have serendipitously complemented the model's strengths or compensated for its weaknesses, resulting in an unusually high performance. However, this level of performance was not maintained in the subsequent rounds, exposing the sensitivity of the FL model to the distribution and diversity of client data. Over the course of 10 federated rounds, the minimum validation accuracy of 48.41% was observed in round 5. This instance highlights the crucial role of data diversity in achieving stable and consistent model performance across different federated learning rounds.

The lower overall performance in the non-IID-C scenario can be attributed to the limitations inherent in the FL setup, where each client employs an identical neural network architecture. This uniformity becomes a bottleneck when the data distribution lacks representation for all application types, leading to certain neurons (corresponding to the absent application types) remaining inactive during training. Conversely, the validation chunks include all application types, introducing a discrepancy between the training and validation phases. This mismatch poses significant challenges for the FL model, as it struggles to generalize across the full spectrum of application types due to the skewed training data.

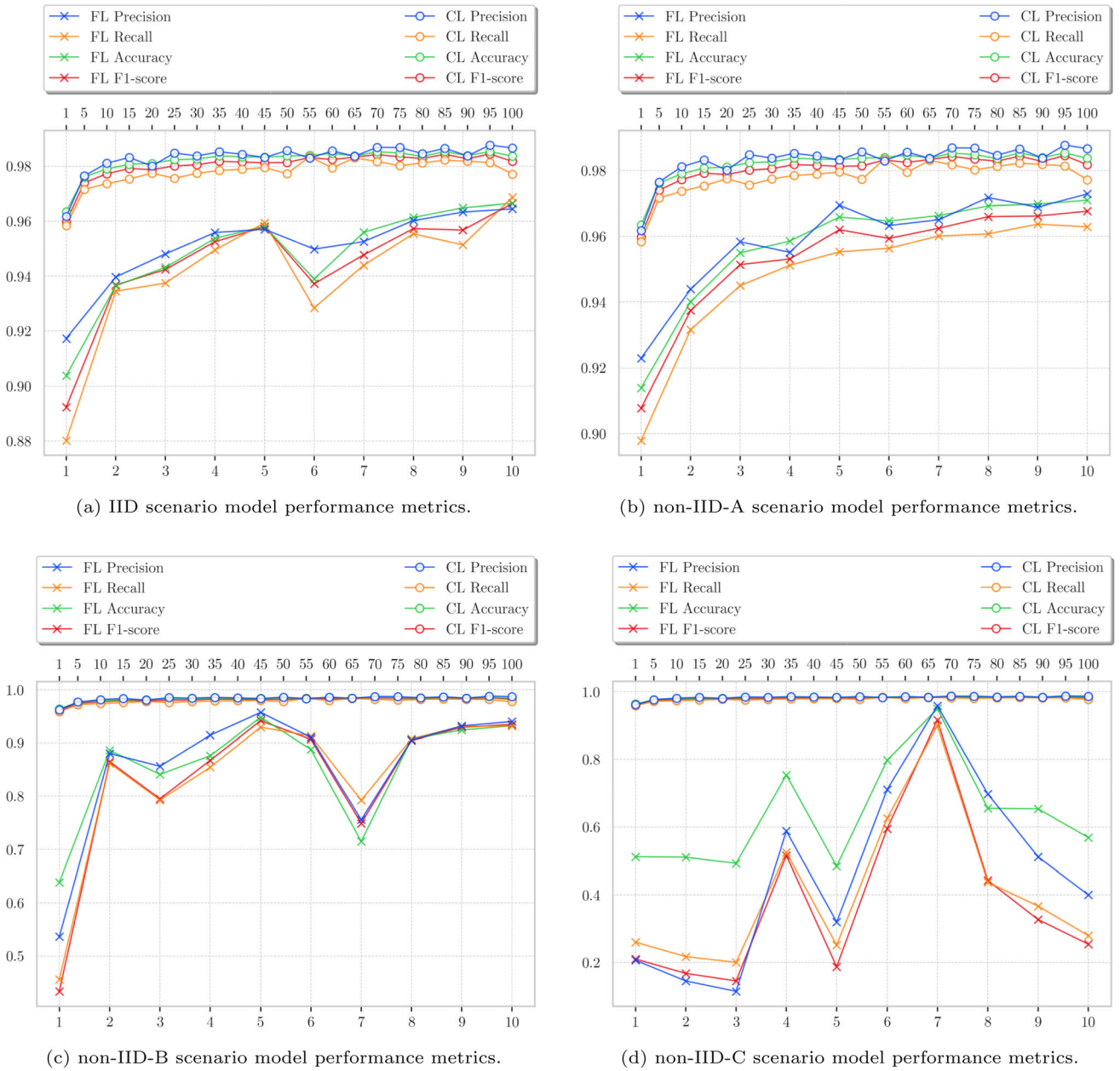
In essence, the non-IID-C scenario accentuates the criticality of diverse and representative data in training robust FL models. The absence of comprehensive data representation hinders the model's ability to generalize effectively, resulting in performance discrepancies when compared to more balanced or centralized learning environments. This scenario serves as a poignant reminder of the complexities and limitations of applying FL in environments characterized by significant data distribution variances.

7.2 Model performance

Figure 6 presents the precision, recall, accuracy, and F_1-score metrics of the FL model, compared against their CL counterparts. For reference, we also include the accuracy presented in Fig. 5, to facilitate better interpretability and comprehensibility of the achieved performance. For clearer visual representation, FL model results are shown for every FedAvg update (bottom x axis), while CL model results are shown for every fifth epoch (top x axis).

The performance metrics reinforce the observations made in Sect. 7.1, providing additional evidence of the FL models' effective learning and generalization across different scenarios. Despite not matching the CL model's performance, the FL models yield competitive results, particularly noteworthy given the inherent challenges in FL setups.

In the IID scenario, the FL model achieved peak performance with a precision of 96.45%, recall of 96.86%, and an F_1-score of 96.64%. For the non-IID-A scenario, the model demonstrated even higher precision at 97.28%, albeit with a slightly lower recall of 96.28%, culminating in an F_1-score of 96.76%. In the more challenging non-IID-B scenario, the FL model still maintained commendable performance levels, with a precision of 95.70%, recall of 92.93%, and an F_1-score of 94.15%. Comparatively, the FL models' performance approaches that of the CL model,



(a) IID scenario model performance metrics.

(b) non-IID-A scenario model performance metrics.

(c) non-IID-B scenario model performance metrics.

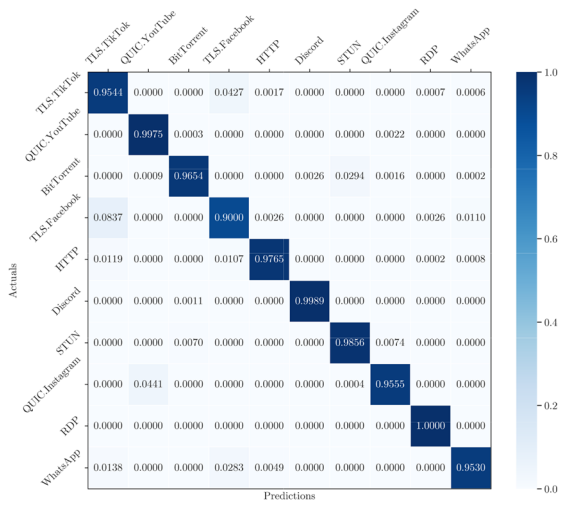
(d) non-IID-C scenario model performance metrics.

Fig. 6 Comparison of model performance metrics—precision, recall, accuracy, and F_1-score—between global FL models and CL across four scenarios: IID, non-IID-A, non-IID-B, and non-IID-C

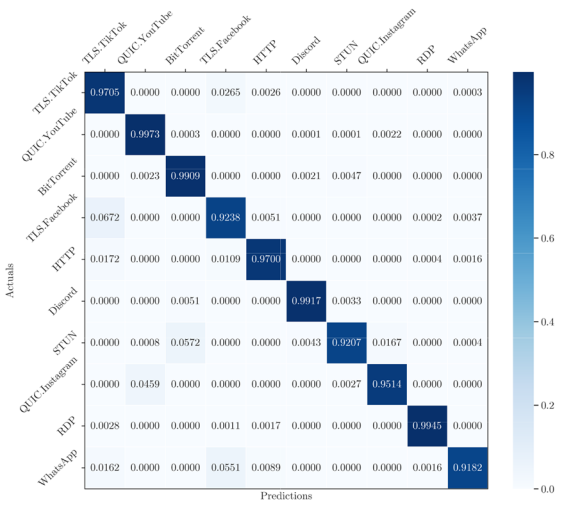
which achieved precision, recall, and F1-scores of 98.66%, 97.71%, and 98.17%, respectively. This closeness in performance metrics, particularly in challenging non-IID environments, reinforces the potential of federated learning as a feasible and privacy-respecting approach in decentralized scenarios.

However, in the non-IID-C scenario (despite reaching a peak performance with a precision of 95.82%, a recall of 89.96%, and an F1-score of 91.63% in round 7), the inability to sustain this performance underlines the importance of data diversity and representation. This underlines the crucial role diverse and representative training data play in developing robust FL models, especially in complicated data landscapes.

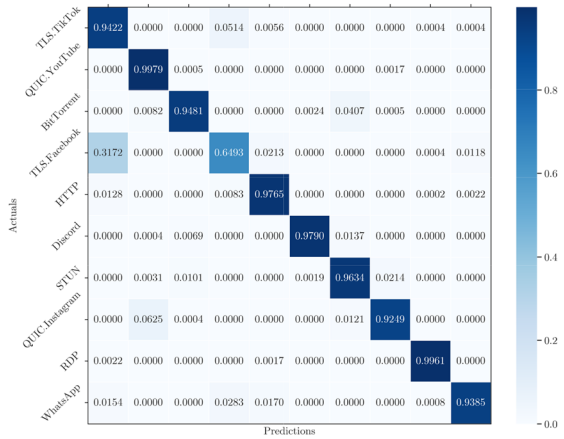
From Fig. 6a–d, the slight discrepancies observed between precision, recall, accuracy, and F1-score metrics are attributable to the inherent nature of FL's decentralized learning process. In FL, the aggregation of diverse local updates to form a global model can lead to slight deviations in performance metrics due to the varying characteristics of local datasets. However, these variations are marginal,



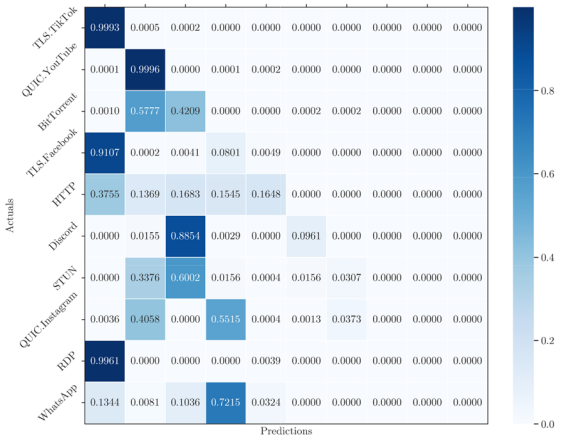
(a) IID scenario confusion matrix.



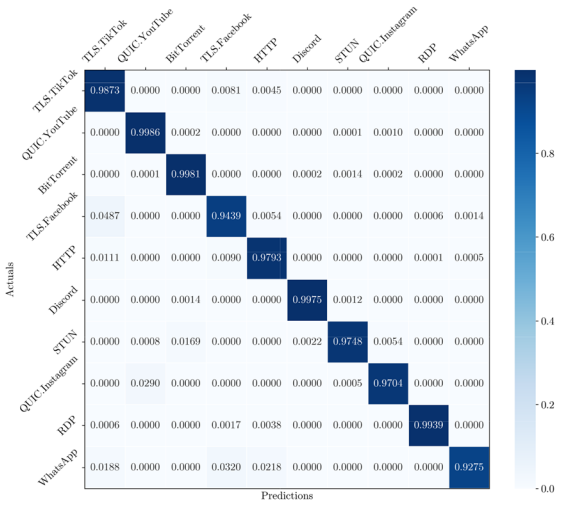
(b) non-IID-A scenario confusion matrix.



(c) non-IID-B scenario confusion matrix.



(d) non-IID-C scenario confusion matrix.



(e) CL model performance metrics.

Fig. 7 Normalized confusion matrices for the FL model across four scenarios: IID, non-IID-A, non-IID-B, and non-IID-C, all evaluated at the last state of the model, compared against the CL model

affirming the FL models' ability to effectively learn and generalize from decentralized and heterogeneous datasets.

Overall, these insights underscore the nuanced yet promising role of FL in leveraging decentralized data for collaborative and incremental learning. They not only highlight FL's capabilities and areas ripe for enhancement but also affirm its feasibility as an alternative to conventional CL methodologies, especially pertinent in scenarios requiring data privacy and, thus, separate data silos.

7.3 Classification performance

The normalized confusion matrices in Fig. 7 offer a comprehensive view of the classification capabilities of the FL model across four scenarios: IID, non-IID-A, non-IID-B, and non-IID-C, with each evaluated at their final training stage, compared against the CL model. Each matrix illustrates the model's performance in accurately classifying traffic patterns into ten distinct application types—'TLS.TikTok,' 'QUIC.YouTube,' 'BitTorrent,' 'TLS.Facebook,' 'HTTP,' 'Discord,' 'STUN,' 'QUIC.Instagram,' 'RDP,' and 'WhatsApp.' The x axis represents the predicted labels, while the y axis represents the actual labels across the ten network applications. The color intensity is proportional to the prediction accuracy, with darker shades indicating higher accuracy. The matrix provides insight into the performance of the model in each class, as well as the misclassifications between classes. From Fig. 7, we observe a high degree of accuracy in classifying most of the applications, as indicated by the high proportions along the diagonal of the matrices.

In the IID (Fig. 7a) and non-IID-A (Fig. 7b) scenarios, the confusion matrices demonstrate a high degree of accuracy across all application types, with minimal misclassifications. This outcome shows the FL model's capability to learn effectively from evenly distributed and representative data. Additionally, the FL model also adeptly manages the complexities in model training introduced through varied sample sizes across chunks, maintaining a competitive edge in classification accuracy. In both cases, the models closely mirror the performance of the centralized model (Fig. 7e).

The non-IID-B scenario, characterized by greater variability in performance improvement due to the more challenging non-IID conditions, including variable sample sizes and distributions, reveals the FL model's resilience (Fig. 7c). The confusion matrix for this scenario uncovers specific areas of misclassification, notably between TLS.TikTok and TLS.Facebook. Given that both applications utilize TLS for secure communication, distinguishing between their traffic patterns is inherently challenging. Despite these misclassification instances, the FL model's overall performance in the non-IID-B scenario demonstrates its capacity to adjust and learn from highly heterogeneous data, affirming the potential of federated learning in complex real-world applications.

The non-IID-C scenario poses the most significant challenge, marked by the absence of certain application types in training chunks. This limitation is evident in the FL model's inferiority to the centralized model (Fig. 7d).

Overall, the confusion matrices for the IID, non-IID-A, and non-IID-B scenarios affirm the nuanced yet promising capabilities of federated learning. While challenges persist, particularly in the more complex non-IID-C scenario, FL models demonstrate a remarkable capacity for adaptation and learning. These findings emphasize the potential of federated learning as a viable and effective approach for collaborative and privacy-preserving machine learning across decentralized data landscapes.

8 Discussion

Our evaluation validates that FL for network traffic flow classification offers a compelling and privacy-aware alternative to traditional methods. While the CL model showed consistently superior performance, the FL approach demonstrated commendable outcomes within a few training iterations, especially under the IID, non-IID-A, and non-IID-B scenarios; a consistent performance, closely approximating the scores of the CL model.

High variability in client accuracies and losses stems from the learning paradigm and data distribution type. In FL, clients retain their data while jointly training and learning a shared model, leading to differing local model performances owing to the diversity in local datasets and learning parameters. Despite this variability, server-side aggregation of these diverse local updates led to an enhanced global model, demonstrated by its stable and low loss. This variability further emphasizes the resilience and adaptability of FL in managing diverse, decentralized data distributions, attesting to its applicability in real-world scenarios where data privacy is critical.

Our experiment was conducted with five clients, a comparatively smaller number than typical FL studies involving dozens or hundreds of clients. This decision was grounded in the nature of our data, and the practical implications of our research, especially considering that our clients represent complex computer networks with hundreds of thousands of communication endpoints. In FL literature, this scenario is known as cross-silo learning, where keeping actual data stowed away and conducting joint learning in a privacy-preserving manner is a strong requirement. We aimed to maintain realism in our experiment by keeping the client count comparatively low. Consequently, our results offer insights into the robustness

of FL, even under challenging data distribution scenarios, justifying its potential for network traffic flow classification.

In FL setups, typically, only a subset of all clients is involved in local model training and global model evaluation owing to communication overhead, privacy concerns, and computational capacity constraints. However, relying solely on a subset of clients for training and validation may introduce model bias and diminish robustness, depending on data quality and availability at the participating clients. This concern is particularly critical in scenarios with non-IID data, where diverse training and validation inputs are vital for model generalization and performance assessment. Such an approach might not fully capture the FL models' potential under varied data distributions.

In our study, we, therefore, chose to train the local model involving all clients and evaluate the aggregated FL model exclusively on the server side to maintain a focused approach to performance assessment, although this method diverges from real-world practices. Future research should explore the effects of partial client participation on the FL models' performance, including the bias-variance trade-off. Investigating these dynamics can uncover methods to enhance FL systems, particularly in contexts where full client engagement is impractical. Understanding these aspects can provide valuable insights for designing more robust FL strategies.

Our research affirms the value of further exploring the application of FL to network traffic flow classification. While our study's breadth was expanded by the moderate client count and exploration of both IID and non-IID scenarios, there are yet unexplored areas. Future studies might investigate the impacts of increasing client numbers or more intricate non-IID scenarios without compromising the realism of the context.

It is true that FL provides some privacy protection by design, yet there exists a growing related literature revealing that some characteristics can be inferred about individual training datasets from the respective gradients. Potential attacks include model inversion [29], membership inference [58], reconstruction attack [69], (hyper)parameter inference [62], and property inference [48]. In turn, techniques such as differential privacy (DP, [19]) and secure aggregation (SA, [46]) can be applied to counter these attacks. DP comes with a hard privacy guarantee but at the expense of utility loss; even SA has been shown to be vulnerable to a quality inference attack, where the quality of the individual training datasets could be derived [55]. Add the array of ever-improving cryptographical solutions such as secure multiparty computation [27], and it is clear that the level of privacy protection provided by collaborative learning mechanisms is still being researched actively.

In summary, our findings endorse FL as a privacy-preserving alternative for network traffic flow classification, providing significant implications for enhancing data privacy in network analytics, especially where sharing raw data might present security or privacy concerns.

9 Conclusion

This paper has delved into the novel utilization of FL for multi-class traffic flow classification, seeking to overcome the challenges posed by data privacy regulations. By enabling the incremental training of ML models on local datasets without the necessity of sharing raw data, FL offers a potent solution for traffic classification that respects privacy concerns.

In a comprehensive comparison of FL to CL, we investigated both IID and non-IID data distribution scenarios. Our findings underscore the robust performance of the FL model, which, even under non-IID conditions, remains resilient, and effective.

The achieved results support the potential of federated learning for contributing significantly to the future of traffic flow classification, offering a high-performing, privacy-preserving method even in challenging contexts. Furthermore, this work paves the way for additional research into other potential applications of FL in various network management and security tasks; this is underpinned by a large novel traffic flow dataset shared with the research community.

Several challenges still exist in implementing FL for TFC, including issues related to communication overhead, algorithm convergence, and model personalization. We anticipate that future research will continue to explore and refine these aspects.

Acknowledgements This study was supported by the GÉANT Innovation Programme 2022, János Bolyai Research Scholarship of the Hungarian Academy of Sciences, the ÚNKP-23-5-BME-461 New National Excellence Program of the Ministry for Culture and Innovation from the source of the National Research, and Development and Innovation Fund. Project no. 138903 has been implemented with the support provided by the Ministry of Innovation and Technology from the NRDI Fund, financed under the FK_21 funding scheme. The work presented in this paper was supported by project no. TKP2021-NVA-02. Project no. TKP2021-NVA-02 has been implemented with the support provided by the Ministry of Culture and Innovation of Hungary from the National Research, Development and Innovation Fund, financed under the TKP2021-NVA funding scheme.

Funding Open access funding provided by Budapest University of Technology and Economics.

Data availability The datasets generated and analyzed during the current study, including the codes, are available in the FlowFrontiers repository, https://github.com/FlowFrontiers/IFLforTFC.

Declarations

Conflict of interest The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

1. Abbasi M, Taherkordi A, Shahraki A (2022) FLITC: A novel federated learning-based method for IoT traffic classification. In: 2022 IEEE International conference on smart computing (SMARTCOMP). IEEE, https://doi.org/10.1109/smartcomp55677.2022.00055
2. Abiteboul S, Stoyanovich J (2019) Transparency, fairness, data protection, neutrality: data management challenges in the face of new regulation. J Data Inf Qual. https://doi.org/10.1145/3310231
3. Aceto G, Ciuonzo D, Montieri A et al (2021) DISTILLER: encrypted traffic classification via multimodal multitask deep learning. J Netw Comput Appl 183–184:102985. https://doi.org/10.1016/j.jnca.2021.102985
4. Alsaedi A, Moustafa N, Tari Z et al (2020) Ton_iot telemetry dataset: a new generation dataset of iot and iiot for data-driven intrusion detection systems. IEEE Access 8:165130–165150. https://doi.org/10.1109/ACCESS.2020.3022862
5. Alshammari R, Zincir-Heywood AN (2009a) Machine learning based encrypted traffic classification: Identifying ssh and skype. In: 2009 IEEE symposium on computational intelligence for security and defense applications, IEEE, pp 1–8
6. Alshammari R, Zincir-Heywood AN (2009b) A preliminary performance comparison of two feature sets for encrypted traffic classification. In: Proceedings of the international workshop on computational intelligence in security for information systems CISIS'08, Springer, pp 203–210
7. Aouini Z, Pekar A (2022) Nfstream: a flexible network data analysis framework. Comput Netw 204:108719. https://doi.org/10.1016/j.comnet.2021.108719
8. Aouini Z, Kortebi A, Ghamri-Doudane Y, et al (2018) Early classification of residential networks traffic using c5. 0 machine learning algorithm. In: 2018 Wireless Days (WD), IEEE, pp 46–53
9. Auld T, Moore AW, Gull SF (2007) Bayesian neural networks for internet traffic classification. IEEE Trans Neural Networks 18(1):223–239
10. Bacquet C, Zincir-Heywood AN, Heywood MI (2009) An investigation of multi-objective genetic algorithms for encrypted traffic identification. In: Computational intelligence in security for information systems. Springer, p 93–100
11. Bar-Yanai R, Langberg M, Peleg D, et al (2010) Realtime classification for encrypted traffic. In: International symposium on experimental algorithms, Springer, pp 373–385
12. Bernaille L, Teixeira R (2007) Early recognition of encrypted applications. In: International conference on passive and active network measurement, Springer, pp 165–175
13. Beutel DJ, Topal T, Mathur A, et al (2022) Flower: a friendly federated learning research framework. https://doi.org/10.48550/ARXIV.2007.14390
14. Bosshart P, Daly D, Gibb G et al (2014) P4: Programming protocol-independent packet processors. SIGCOMM Comput Commun Rev 44(3):87–95. https://doi.org/10.1145/2656877.2656890
15. Boutaba R, Salahuddin MA, Limam N et al (2018) A comprehensive survey on machine learning for networking: evolution, applications and research opportunities. J Internet Serv Appl 9(1):16. https://doi.org/10.1186/s13174-018-0087-2
16. Bujlow T, Carela-Español V, Barlet-Ros P (2015) Independent comparison of popular dpi tools for traffic classification. Comput Netw 76:75–89. https://doi.org/10.1016/j.comnet.2014.11.001
17. Condoluci M, Mahmoodi T (2018) Softwarization and virtualization in 5g mobile networks: benefits, trends and challenges. Comput Netw 146:65–84. https://doi.org/10.1016/j.comnet.2018.09.005
18. Dainotti A, Pescapé A, Sansone C (2011) Early classification of network traffic through multi-classification. In: International workshop on traffic monitoring and analysis, Springer, pp 122–135
19. Desfontaines D, Pejó B (2020) Sok: Differential privacies. Proceedings on privacy enhancing technologies
20. Dijkhuizen NV, Ham JVD (2018) A survey of network traffic anonymisation techniques and implementations. ACM Comput Surv. https://doi.org/10.1145/3182660
21. Dixon L, Ristenpart T, Shrimpton T (2016) Network traffic obfuscation and automated internet censorship. IEEE Secur Privacy 14(6):43–53. https://doi.org/10.1109/MSP.2016.121
22. Draper-Gil G, Lashkari AH, Mamun MSI, et al (2016) Characterization of encrypted and VPN traffic using time-related features. In: Proceedings of the 2nd international conference on information systems security and privacy. SCITEPRESS - Science and and Technology Publications, https://doi.org/10.5220/0005740704070414, https://www.unb.ca/cic/datasets/vpn.html
23. Engelen G, Rimmer V, Joosen W (2021) Troubleshooting an intrusion detection dataset: the cicids2017 case study. In: 2021 IEEE security and privacy workshops (SPW), IEEE, pp 7–12
24. Erman J, Arlitt M, Mahanti A (2006a) Traffic classification using clustering algorithms. In: Proceedings of the 2006 SIGCOMM workshop on mining network data, pp 281–286
25. Erman J, Mahanti A, Arlitt M (2006b) Qrp05-4: Internet traffic identification using machine learning. In: IEEE Globecom 2006, IEEE, pp 1–6
26. Este A, Gringoli F, Salgarelli L (2009) Support vector machines for tcp traffic classification. Comput Netw 53(14):2476–2490
27. Fereidooni H, Marchal S, Miettinen M, et al (2021) Safelearn: Secure aggregation for private federated learning. In: 2021 IEEE security and privacy workshops (SPW), pp 56–62, https://doi.org/10.1109/SPW53761.2021.00017
28. FlowFrontiers (2024) https://github.com/FlowFrontiers/IFLforTFC/blob/main/4-evaluate-fl-iid-shuffled.ipynb, Accessed on 21 Feb 2024
29. Fredrikson M, Jha S, Ristenpart T (2015) Model inversion attacks that exploit confidence information and basic countermeasures. In: Proceedings of the 22nd ACM SIGSAC conference on computer and communications security

30. Geurts P, Ernst D, Wehenkel L (2006) Extremely randomized trees. Mach Learn 63(1):3–42. https://doi.org/10.1007/s10994-006-6226-1
31. Goldsteen A, Ezov G, Shmelkin R et al (2022) Anonymizing machine learning models. In: Garcia-Alfaro J, Muñoz-Tapia JL, Navarro-Arribas G et al (eds) Data privacy management, cryptocurrencies and blockchain technology. Springer, Cham, pp 121–136. https://doi.org/10.1007/978-3-030-93944-1_8
32. Guo Y, Wang D (2023) FEAT: a federated approach for privacy-preserving network traffic classification in heterogeneous environments. IEEE Internet Things J 10(2):1274–1285. https://doi.org/10.1109/jiot.2022.3204975
33. Hofstede R, Čeleda P, Trammell B et al (2014) Flow monitoring explained: from packet capture to data analysis with netflow and ipfix. IEEE Commun Surv Tutor 16(4):2037–2064. https://doi.org/10.1109/COMST.2014.2321898
34. Jin Z, Liang Z, He M et al (2023) A federated semi-supervised learning approach for network traffic classification. Int J Netw Manag. https://doi.org/10.1002/nem.2222
35. Kairouz P, McMahan HB, Avent B, et al (2021) Advances and open problems in federated learning. Foundations and Trends® in Machine Learning 14(1–2):1–210. https://doi.org/10.1561/2200000083
36. Lanvin M, Gimenez PF, Han Y, et al (2023) Errors in the CICIDS2017 dataset and the significant differences in detection performances it makes. In: Lecture Notes in Computer Science. Springer Nature Switzerland, p 18–33, https://doi.org/10.1007/978-3-031-31108-6_2
37. Lashkari AH, Gil GD, Mamun MSI, et al (2017) Characterization of tor traffic using time based features. In: Proceedings of the 3rd international conference on information systems security and privacy. SCITEPRESS - Science and Technology Publications,https://doi.org/10.5220/0006105602530262, https://www.unb.ca/cic/datasets/tor.html
38. Lashkari AH, Kaur G, Rahali A (2020) DIDarknet: A contemporary approach to detect and characterize the darknet traffic using deep image learning. In: 2020 the 10th International conference on communication and network security. ACM, https://doi.org/10.1145/3442520.3442521, https://www.unb.ca/cic/datasets/darknet2020.html
39. Lee S, Levanti K, Kim HS (2014) Network monitoring: present and future. Comput Netw 65:84–98. https://doi.org/10.1016/j.comnet.2014.03.007
40. Li B, Springer J, Bebis G et al (2013) A survey of network flow applications. J Netw Comput Appl 36(2):567–581. https://doi.org/10.1016/j.jnca.2012.12.020
41. Li T, Li N (2009) On the tradeoff between privacy and utility in data publishing. In: Proceedings of the 15th ACM SIGKDD international conference on knowledge discovery and data mining. Association for Computing Machinery, New York, NY, USA, KDD '09, p 517-526, https://doi.org/10.1145/1557019.1557079
42. Liu L, Engelen G, Lynar T, et al (2022) Error prevalence in nids datasets: A case study on cic-ids-2017 and cse-cic-ids-2018. In: 2022 IEEE conference on communications and network security (CNS), pp 254–262, https://doi.org/10.1109/CNS56114.2022.9947235
43. Liu Y, Li W, Li YC (2007) Network traffic classification using k-means clustering. In: Second international multi-symposiums on computer and computational sciences (IMSCCS 2007), IEEE, pp 360–365
44. Majeed U, Khan LU, Hong CS (2020) Cross-silo horizontal federated learning for flow-based time-related-features oriented traffic classification. In: 2020 21st Asia-Pacific network operations and management symposium (APNOMS). IEEE, https://doi.org/10.23919/apnoms50412.2020.9236971
45. McMahan B, Moore E, Ramage D, et al (2017a) Communication-Efficient Learning of Deep Networks from Decentralized Data. In: Singh A, Zhu J (eds) Proceedings of the 20th international conference on artificial intelligence and statistics, proceedings of machine learning research, vol 54. PMLR, pp 1273–1282, https://proceedings.mlr.press/v54/mcmahan17a.html
46. McMahan B, Moore E, Ramage D, et al (2017b) Communication-Efficient Learning of Deep Networks from Decentralized Data. In: Singh A, Zhu J (eds) Proceedings of the 20th International conference on artificial intelligence and statistics, proceedings of machine learning research, vol 54. PMLR, pp 1273–1282, https://proceedings.mlr.press/v54/mcmahan17a.html
47. van der Mei R, van den Berg H, Ganchev I, et al (2018) State of the art and research challenges in the area of autonomous control for a reliable internet of services. In: Lecture Notes in Computer Science. Springer International Publishing, p 1–22, https://doi.org/10.1007/978-3-319-90415-3_1
48. Melis L, Song C, De Cristofaro E, et al (2019) Exploiting unintended feature leakage in collaborative learning. In: 2019 IEEE Symposium on Security and Privacy (SP), IEEE
49. Mijumbi R, Serrat J, Gorricho JL et al (2016) Network function virtualization: state-of-the-art and research challenges. IEEE Commun Surv Tutor 18(1):236–262. https://doi.org/10.1109/COMST.2015.2477041
50. Moore A, Zuev D, Crogan M (2005) Discriminators for use in flow-based classification. Tech. Rep. RR-05-13, University of Cambridge, https://www.cl.cam.ac.uk/research/srg/netos/projects/archive/nprobe/data/papers/sigmetrics/index.html
51. Moore AW, Zuev D (2005) Internet traffic classification using bayesian analysis techniques. In: Proceedings of the 2005 ACM SIGMETRICS international conference on Measurement and modeling of computer systems. ACM, https://doi.org/10.1145/1064212.1064220
52. Mun H, Lee Y (2021) Internet traffic classification with federated learning. Electronics 10(1):27. https://doi.org/10.3390/electronics10010027
53. Nguyen TT, Armitage G (2008) A survey of techniques for internet traffic classification using machine learning. IEEE Commun Surv Tutor 10(4):56–76. https://doi.org/10.1109/SURV.2008.080406
54. Pau G, Bakhshi T (2017) State of the art and recent research advances in software defined networking. Wirel Commun Mob Comput 2017:7191647. https://doi.org/10.1155/2017/7191647
55. Pejó B, Biczók G (2023) Quality inference in federated learning with secure aggregation. IEEE Trans Big Data. https://doi.org/10.1109/TBDATA.2023.3280406
56. Roughan M, Sen S, Spatscheck O, et al (2004) Class-of-service mapping for qos: a statistical signature-based approach to ip traffic classification. In: Proceedings of the 4th ACM SIGCOMM conference on Internet measurement, pp 135–148
57. Sharafaldin I, Lashkari AH, Ghorbani AA (2018) Toward generating a new intrusion detection dataset and intrusion traffic characterization. In: Proceedings of the 4th international conference on information systems security and privacy, CISSP 2018, pp 108–116, https://doi.org/10.5220/0006639801080116
58. Shokri R, Stronati M, Song C, et al (2017) Membership inference attacks against machine learning models. In: 2017 IEEE symposium on security and privacy (SP), IEEE
59. Strobel M, Shokri R (2022) Data privacy and trustworthy machine learning. IEEE Secur Priv 20(05):44–49. https://doi.org/10.1109/MSEC.2022.3178187
60. Sun C, Chen B, Bu Y, et al (2023) Traffic classification method based on federated semi-supervised learning. In: Proceedings of the 2022 6th international conference on electronic information

technology and computer engineering. ACM, https://doi.org/10.1145/3573428.3573586
61. Sun R, Yang B, Peng L, et al (2010) Traffic classification using probabilistic neural networks. In: 2010 Sixth international conference on natural computation, IEEE, pp 1914–1919
62. Tramèr F, Zhang F, Juels A, et al (2016) Stealing machine learning models via prediction apis. In: 25th USENIX security symposium (USENIX Security 16)
63. Wei T, Wang Y, Li W (2022) The deep flow inspection framework based on horizontal federated learning. In: 2022 23rd Asia-Pacific network operations and management symposium (APNOMS). IEEE, https://doi.org/10.23919/apnoms56106.2022.9919969
64. Whang SE, Roh Y, Song H et al (2023) Data collection and quality challenges in deep learning: a data-centric AI perspective. VLDB J. https://doi.org/10.1007/s00778-022-00775-9
65. Williams N, Zander S, Armitage G (2006) A preliminary performance comparison of five machine learning algorithms for practical ip traffic flow classification. ACM SIGCOMM Comput Commun Rev 36(5):5–16. https://doi.org/10.1145/1163593.1163596
66. Zhang J, Chen C, Xiang Y et al (2012) Internet traffic classification by aggregating correlated naive bayes predictions. IEEE Trans Inf For Secur 8(1):5–15. https://doi.org/10.1109/TIFS.2012.2223675
67. Zhou D, Yan Z, Fu Y et al (2018) A survey on network data collection. J Netw Comput Appl 116:9–23. https://doi.org/10.1016/j.jnca.2018.05.004
68. Zhou P (2020) Federated deep payload classification for industrial internet with cloud-edge architecture. In: 2020 16th International conference on mobility, sensing and networking (MSN). IEEE, https://doi.org/10.1109/msn50589.2020.00048
69. Zhu L, Liu Z, Han S (2019) Deep leakage from gradients. In: Advances in neural information processing systems
70. Zhu M, Chen Z, fan Chen K, et al (2022) Attention-based federated incremental learning for traffic classification in the internet of things. Comput Commun 185:168–175. https://doi.org/10.1016/j.comcom.2022.01.006
71. Zhu X, Shu N, Wang H, et al (2021) A distributed traffic classification model based on federated learning. In: 2021 7th international conference on big data computing and communications (BigCom). IEEE, https://doi.org/10.1109/bigcom53800.2021.00022