POLITECNICO DI TORINO
Repository ISTITUZIONALE

FedScope—Federated Host Embeddings From Telescope Traffic: Design and Implementation

Original
FedScope—Federated Host Embeddings From Telescope Traffic: Design and Implementation / Huang, K., Sordello, A., Valentim, R.V., Vassio, L., Drago, I., Mellia, M.. - In: IEEE TRANSACTIONS ON NETWORK AND SERVICE MANAGEMENT. - ISSN 1932-4537. - 23:(2026), pp. 4213-4227. [10.1109/tnsm.2026.3685756]

Availability:
This version is available at: 11583/3010556 since: 2026-05-05T10:49:51Z

Publisher:
Institute of Electrical and Electronics Engineers Inc.

Published
DOI:10.1109/tnsm.2026.3685756

Terms of use:

This article is made available under terms and conditions as specified in the corresponding bibliographic description in the repository

Publisher copyright

(Article begins on next page)

06 October 2026

FedScope—Federated Host Embeddings From Telescope Traffic: Design and Implementation

Kai Huang, Andrea Sordello, Rodolfo Vieira Valentim, Luca Vassio, Idilio Drago, and Marco Mellia, *Fellow, IEEE*

***Abstract*—A network telescope is a range of IP addresses that host no services. Millions of bots and scanners contact it to look for vulnerable systems, and the traffic it exposes is fundamental to understanding malicious activities. The visibility a telescope offers depends on its size and geolocation, and merging the information from multiple telescopes could help increase visibility and uncover more malicious activities. However, sharing raw telescope data is complicated, calling for solutions that allow one to directly share the knowledge rather than the data obtained from multiple deployments. In this paper, we explore the application of Federated Learning (FL) to create and share such global knowledge from the malicious activities seen in distributed telescopes. For that, we introduce FedScope, an FL-based solution for generating *host embeddings* in a distributed way. We compare FedScope to local and distributed alternatives in downstream tasks, such as sender classification or coordinated activities detection. We show that FedScope 1) produces embeddings of equal or higher quality than those of a single telescope; 2) increases coverage, allowing the global model to monitor more malicious actors; 3) avoids the sharing of the raw data, limiting exchanged data.**

***Index Terms*—Federated learning, network telescope, traffic analysis, host embeddings.**

I. Introduction

NETWORK telescopes are IP address ranges that do not host any services [1]. They are primary tools for collecting information about the activity of malicious hosts in early attack stages. They receive only unsolicited packets that include routine scans from benign actors, botnet activities, packets coming from misconfigured services, and backscattering, i.e., victims' responses to attacks performed using spoofed addresses.

While providing a rich and effective source for cybersecurity, a single telescope offers only a partial view of malicious scan activities [2]. Their visibility is limited by factors such as their size (number of IP addresses) and geolocation. Intuitively, merging information from multiple telescopes could increase visibility and uncover unseen malicious activities. However, sharing raw telescope data is usually complex due to volume constraints and organisational policies.

To extract actionable knowledge from telescopes, several works started employing Machine Learning (ML) pipelines. Recently, the ML two-stage pipeline [3], [4], [5], [6] has demonstrated particularly appealing. In the first stage, a self-supervised upstream task generates embeddings–compressed representations of input data in latent space that act as generic features, learned without the need to have labels. In the second stage, one can use these vectors in *downstream tasks* for, e.g., sender classification, clustering or anomaly detection [5], [7], [8], [9]. Here, from raw telescope packets, we create *host embeddings* to summarise the activity of each host contacting the telescope (a *sender*), i.e., each sender gets represented as a vector in the embedding space. The automatic learning of embeddings avoids the need to define the features the model has to use for the final task.

Joining host embeddings that summarise the telescope traffic sounds appealing since they could allow the sharing of generic features instead of raw data, in a *Federated Learning* (FL) approach [10], [11]. However, telescopes observe traffic coming from different sets of IP addresses, and the amount of traffic that each telescope observes depends on its characteristics. Because of that, even the activity of the same senders may significantly differ when observed from different telescopes. In sum, the consolidation of embeddings built by several independent telescope operators requires ingenuity. In this work, we answer the following research questions to address these challenges:

- **RQ1**: What are the benefits of sharing telescope information to learn host activity patterns?
- **RQ2**: What are the tradeoffs in learning such patterns? Who benefits the most?
- **RQ3**: What are the challenges to learn host representation in an FL approach?

We here introduce FedScope, a system to perform federated learning of host embeddings from multiple telescopes.[1] We first assess to what extent combining multiple telescope data can improve the learning of host embeddings, assuming a

Received 1 April 2025; revised 30 December 2025 and 13 March 2026; accepted 14 April 2026. Date of publication 20 April 2026; date of current version 24 April 2026. This work was partially supported by project SERICS (PE00000014) under the MUR National Recovery and Resilience Plan funded by the European Union–NextGenerationEU and the PRIN Projects xInternet (eXplainable Internet–20225CETN9–PRIN 2022) and ACRE (AI-Based Causality and Reasoning for Deceptive Assets–2022EP2L7H). The associate editor coordinating the review of this article and approving it for publication was L. Qu. *(Corresponding author: Kai Huang.)*

Kai Huang, Andrea Sordello, Luca Vassio, and Marco Mellia are with Politecnico di Torino, 10129 Torino, Italy (e-mail: kai.huang@polito.it; andrea.sordello@polito.it; luca.vassio@polito.it; marco.mellia@polito.it).

Rodolfo Vieira Valentim and Idilio Drago are with Università di Torino, 10124 Torino, Italy (e-mail: rodolfo.valentim@unito.it; idilio.drago@unito.it).

Digital Object Identifier 10.1109/TNSM.2026.3685756

[1] A preliminary version of this work has appeared in [12], where we perform an initial assessment of distributed learning of host embeddings.

© 2026 The Authors. This work is licensed under a Creative Commons Attribution 4.0 License.
For more information, see https://creativecommons.org/licenses/by/4.0/

global view of the raw traffic. We evaluate the quality of the globally produced embeddings via host classification and clustering problems across multiple scenarios by varying both the number of operators and the size of the telescopes. This allows us to assess the benefits of data sharing.

After confirming the advantages of combining multiple telescopes, we design and engineer FedScope to perform the learning in a fully distributed way, showing that FL is a practical solution that allows the sharing of knowledge across different parties without requiring the sharing of raw data [11]. We address the challenges posed by varying resources and constraints across domains, as well as the heterogeneity of data each telescope observes.

Our results demonstrate that collaboration among multiple operators improves the quality of embeddings compared to local approaches in both classification and clustering tasks. Sharing knowledge using an FL approach, as in FedScope achieves performance comparable to a centralised learning approach (which shares raw data), while reducing the shared data. To increase the impact of FedScope we contribute it as open source to the community.[2]

We summarise our key contributions as follows:

- We introduce FedScope, the first self-supervised federated framework for learning host embeddings from network telescope traffic without sharing raw traces.
- We design a novel vocabulary synchronisation and eviction mechanism that enables federated Word2Vec-style models with dynamically changing input spaces.
- We provide an extensive evaluation on real-world datasets, showing that FedScope achieves performance comparable to centralised learning, enabling efficient knowledge sharing.

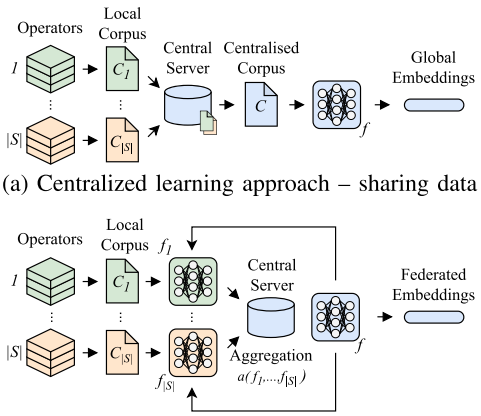
Section II presents our scenario and provides some background on host embedding learning. In Section III we describe the federated learning approach we adopt. We introduce the design and implementation of FedScope in Section IV. We then validate it by answering our research questions in Section V, Section VI and Section VII, considering supervised and unsupervised learning downstream tasks as well as runtime performance. Section VIII discusses previous work, while Section IX concludes the paper.

II. Scenario and Host Embeddings

Figure 1 summarizes the scenario we consider. Each operator deploys a telescope $s \in S$ in its network to monitor unsolicited traffic. Each telescope builds what we formally define as a *corpus* C_s, i.e., the ordered list of senders (identified by their IP address) that reach the telescope services (identified by the ports). We use such a corpus to build a self-supervised model f, using an ML pipeline borrowed from NLP. We consider three alternative approaches:

- Local learning: This is our baseline, where each telescope learns its model f_s independently as in previous works [5], [6], [13]. This baseline allows us to quantify the benefits of sharing telescope data among operators.

[2]https://github.com/SmartData-Polito/fedscope



(a) Centralized learning approach – sharing data

(b) Federated learning approach – sharing models

Fig. 1. Scenario: Centralised vs federated approaches for host representation learning.

- Centralised learning (Figure 1a): We assume that operators share data without restrictions. Here, a central entity collects all telescope data and trains the global model f considering all observed traffic. This represents a theoretical best-case scenario with complete visibility.
- Federated learning (Figure 1b): We design FedScope to build a global model without exchanging raw data. Each operator s trains a local model f_s, and shares it with the aggregator, which consolidates them in a global model f before redistributing it back.

We assume that the goal of telescope operators is to build *host embeddings* that are employed in downstream machine learning tasks. In the first stage, the operator adopts a self-supervised representation learning step, uses raw traffic traces collected by the telescope as input, to build its embeddings. Such compact numerical vectors can later serve as input to different ML tasks based on the requirements of different operators. In the following, we introduce some background on self-supervised representation learning for readers not familiar with the approach, followed by a description of the algorithm we select for testing the federated learning approach.

A. Self-Supervised Representation Learning

Supervised machine learning is inadequate for telescope data analysis since ground truth labels are scarce in this scenario. Self-supervised representation learning offers a valid alternative because it formulates the learning problem without external supervised signals, i.e., using the same input data as labels. For example, the prediction of the next word in a sentence or, as in our case, the prediction of the next sender IP address reaching the telescope. This self-supervised formulation allows the model to learn compact numerical features that capture underlying behaviour from the data without explicit labels. This approach is at the root of the success of NLP solutions, such as the Generative Pre-trained Transformer (GPT) models. These models pass a pre-training stage where they learn to map words to the latent space to maximise the accuracy of predicting missing parts in texts.

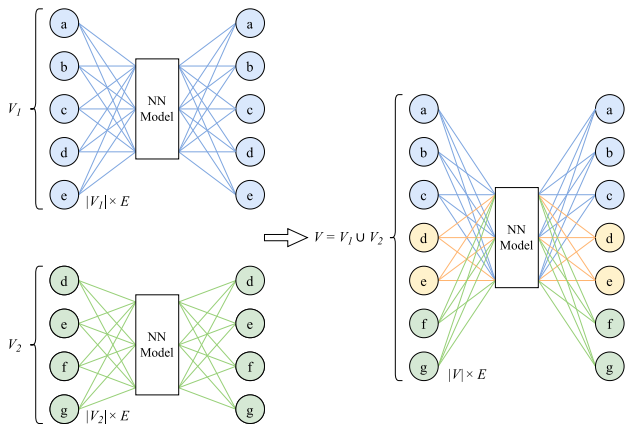


Fig. 2. Model aggregation among 2 operators; yellow neurons represent the intersection of two vocabularies.

For the analysis of telescope traffic, the host embeddings map host activity in a latent space, where hosts with similar behaviours should be positioned closer together. Researchers proposed multiple algorithms for learning the embeddings, each capturing different factors from the data, such as temporal traffic patterns [5], [6], similarities in traffic features [13] and structural traffic properties, such as typical sender/receiver patterns [6]. Here, we rely on our previous approach called i-DarkVec [5], which we summarise next.

B. Learning Host Embeddings–i-DarkVec

i-DarkVec relies on Word2Vec [14], a well-known natural language processing technique that learns a vector representation for each word in a vocabulary.[3] The goal of i-DarkVec is to represent senders of packets identified by their IP addresses, considering their traffic patterns. The algorithm maps each IP address in a meaningful latent space that can be used to uncover coordinated host activities, such as attacks from botnets, or scanning activities performed by a pool of senders. For this, i-DarkVec extracts the sequences of packets destined to the same TCP/UDP ports. Analogously to NLP, IP addresses of senders represent "words" and the sequence in which they reach the telescope represents "sentences". i-DarkVec feeds the generated sequences as input to Word2Vec, a simple neural network autoencoder in which each input/output neuron is associated with a given word present in the given vocabulary (see Fig. 2). The model learns to predict which words would likely appear in a sentence given a word in input. Notice that the autoencoder architecture is fixed, given the vocabulary. The resulting embeddings capture the semantic and syntactic relationships between words so that words that often co-occur in the same sentence are mapped closer in the latent space [14]. As Word2Vec produces word embedding, i-DarkVec produces host embeddings: senders that co-occur in time and target similar services are mapped closer in the latent space.

In practice, the representation learning stage operates on sequences derived from source IP addresses and destination ports aggregated over fixed time windows. The model does not rely on payloads, destination IP addresses, or protocol-level fields, and learns host embeddings solely from sender–port co-occurrence patterns and coarse traffic volume, thus minimising the information shared among participants.

Formally, given a *vocabulary* of senders $V = \{v_1, v_2, v_3, \ldots\}$ and the sequences of packets they send, i.e., the *corpus* C, we map each entity $v \in V \rightarrow u \in \mathbb{R}^E$ where u is the embedding of v in the E dimensional latent space. The function $e : V \rightarrow \mathbb{R}^E$ is the embedding function (e.g., Word2Vec) that we train in a self-supervised manner using the masked language technique. Finally, given the function e, let $\mathbf{X} := [e(v)]_{v \in V} \in \mathbb{R}^{|V| \times E}$ be the matrix of embeddings for all senders in V.[4] The model f will include the embedding function e and all the other weights in the neural network autoencoder.

These embeddings can then be used to solve supervised and unsupervised downstream tasks, e.g., assigning hosts a known label or finding clusters of similar hosts.

III. Learning Federated Representation

We propose a collaborative approach to learn embeddings of telescope traffic as shown in Figure 1b. In the following, we detail each required step.

A. Federated Learning at Large

Federated learning is a decentralized machine learning approach where multiple parties train a shared model collaboratively without exchanging raw data. The architecture typically consists of a central *server* coordinating model updates from distributed *clients*, which perform local training on their private datasets. At each *round*, the server collects the local models f_s from clients, aggregates them into a global model f according to a *weighting function*, and returns the global model f to each client. The primary goal of FL is to train the model while preserving data privacy and reducing communication costs.

FL assumes all clients share the same model architecture, so they have to exchange the model weights only. In our setup, this is not true and calls for additional ingenuity. In fact, not all telescopes will observe the same set of senders, and each has a different vocabulary. We thus need to define the common superset, i.e., a common vocabulary. For instance, in Figure 2 we have two telescopes, one observing traffic from hosts $V_1 = \{v_1, v_2, v_3, v_4, v_5\}$, the other from $V_2 = \{v_4, v_5, v_6, v_7\}$. When aggregating their data, the global vocabulary becomes $V = V_1 \cup V_2$. This implies a change in the model architecture, because the number of input and output neurons is now different–with new connections to/from the hidden layer neurons. Notice that E, the size of the hidden layer ($E = 3$ in the example) can be the same for all models.

B. Federated Learning Host Embedding

To build a common model from multiple networks using FL, we rely on the *FedAvg* [11] algorithm. It allows distributed

[3]We assume that readers are familiar with Word2Vec and only provide a brief overview of i-DarkVec. Details are presented in [5].

[4]In Word2Vec, the produced embeddings are the learnable weight matrix between the input and the hidden layer.

training among a large number of clients with a simple weighting function. We provide an overview of the FL approach in Figure 1b.

We adopt FedAvg because it introduces no additional assumptions and allows us to isolate the core challenges addressed by FedScope, namely federated self-supervised representation learning with non-aligned and evolving vocabularies.

Given a set of telescope operators S, each operator $s \in S$ (i) generates the corpus C_s according to the traffic it observes; (ii) trains or updates its local model f_s (including its embedding function e_s); (iii) generates the local embedding matrix $\mathbf{X}_s \in \mathbb{R}^{|V_s| \times E}$, where V_s is the local vocabulary. The central server receives the local models f_s and vocabularies V_s. It aggregates the local models into a federated model that can be applied to senders observed in at least one of the vocabularies. As said, each client may have a different vocabulary V_s and the server must compute the global vocabulary $V = \bigcup_s V_s$. Notice that, in practical cases, the model size might be constrained in size ($|V| < M$), hence also the final vocabulary size might not exceed M elements. A host eviction policy is needed in these cases.

The server aggregates the embedders $e_1, \ldots, e_{|S|}$ into a new function $e : \bigcup_s V_s \to \mathbb{R}^E$, where $e = a\left(e_1, \ldots, e_{|S|}\right)$. Similarly, the server aggregates the rest of the models $\{f_s\}_{s \in S}$. Finally, the server sends back f and V to each client. Notice that now, for each client s, $f_s = f$, $e_s = e$ and $V_s = V$. We can repeat this update procedure for multiple rounds.

We use as aggregation function $a()$ the weighted average between the local embeddings produced by each client. We show an example of aggregation of 2 operators in Figure 2. Notice that some local embeddings might not be defined for some operators, e.g., only $\{v_4, v_5\}$ appear in both vocabularies. The weighted average is only performed among the parameters related to common words, highlighted in yellow in the example.

Formally, we define the final federated embedding for a host $v \in \bigcup_s V_s$ as:

$$e(v) = \frac{\sum_{s: v \in V_s} |V_s| \cdot e_s(v)}{\sum_{s: v \in V_s} |V_s|}$$

where $|V_s| \in \mathbb{R}$ is the weight referred to the local embeddings of sender v observed in client s.

We use vocabulary size as a simple and robust weighting choice. Intuitively, observing more senders means acquiring more information. As shown in Appendix, alternative aggregation schemes yield comparable downstream performance. Notice that in this case, the weight is the same for each host $v \in V_s$.

In addition, note that only model parameters and sender identifiers are exchanged during aggregation; no raw traffic records, payloads, or protocol-level features are shared.

At last, the aggregation procedure tolerates heterogeneous client update frequencies and temporary client dropout, as only the models received in a given round are aggregated.

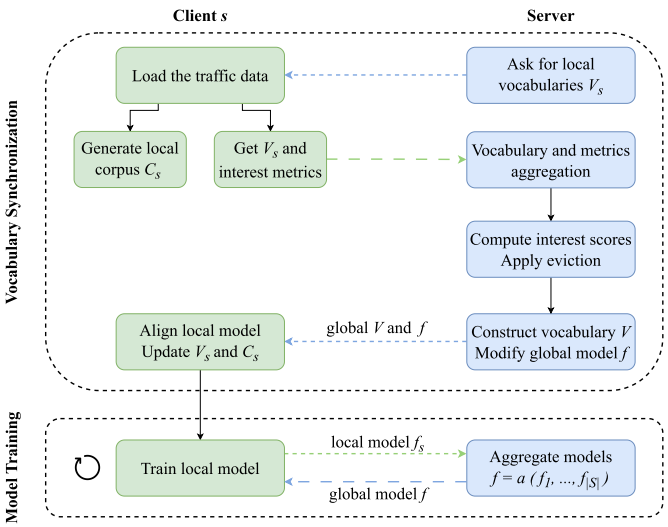


Fig. 3. Federated design of FedScope.

IV. FedScope

Deploying i-DarkVec with the distributed FL training setup in realistic environments introduces challenges. Most existing FL frameworks operate with a pre-defined model architecture and overlook our scenario where the model architecture changes over time. In our setup, it is essential to change and, at the same time, limit the model size to prevent it from overwhelming client resources.

A. FedScope Design

As in i-DarkVec, we consider slotted time and process all traffic in a time window, denoted by the index t. For each packet, we extract the source IP address, the destination port and the number of packets that source address has produced in that timeslot. At the beginning of each period, the server instructs the clients to start a new training period with all packets collected in that interval.

To address the challenges of our scenario, we propose an FL pipeline for FedScope and introduce a novel interaction between the FL clients and the FL server before the traditional FL model aggregation phase.

This new phase enables building and updating the global model architecture by exchanging local vocabularies $\{V_s\}_{s \in S}$. With these, we compute the global vocabulary V and use it to update the model f. This interaction occurs before the standard FL client/server interaction for the local model training and the global model aggregation. Figure 3 presents the details of FedScope pipeline. It consists of two phases.

1) Vocabulary Synchronisation: In this novel phase, the server aggregates client vocabularies V_s.

- **Client**: Each client processes the raw traffic observed in the period t, builds the local corpus C_s, i.e. extracts the features required by i-DarkVec, builds the local vocabulary V_s and computes senders' interest metrics–see Section IV-B for details.
- **Server**: After receiving all clients' vocabularies and interest metrics, the server updates the global vocabulary V

with new senders appearing in the local vocabularies. In this phase, the clients only share with the server the presence of sender v_i and the sender's interest metrics. We do not share any other information, i.e., no destination IP address, no port contacted by v, no payload, etc.
If $|V| > M$, the server retains only the top-M most interesting senders. The model embeddings are then initialised or pruned according to the sender present in V.

At the end of this phase, all clients agree on a common vocabulary and global model with aligned architecture. Notice that clients may need to update the corpus C_s if the server filtered out any sender which is present in C_s.

2) Model Training: In this phase, we conduct the standard FL training round following the *FedAvg* framework: Each client trains the model f_s with the local corpus C_s; The server receives all local models f_s and aggregates them into the global model f and sends it back to all clients. Since the model and vocabulary are already synchronised, we can repeat the FL training phase for multiple rounds without repeating the first phase.

B. IP Eviction Policy

Because each telescope keeps observing new senders [2], the dimensionality of the vocabulary V grows. Sharing information between multiple operators will make these numbers grow even faster, increasing the model size and calling for additional computational resources. Conversely, some senders will disappear over time, thus making the information captured by the model stale and likely noisy. To address these issues and make the solution scalable, we introduce an eviction policy to limit the global vocabulary V size to M and model embedding matrix $\mathbf{X}$.

The objective is to prioritise senders. To achieve this, we define the *interest score* $I_t(v)$ for each sender $v \in V$ at time t based on the following intuitions:

- We consider recent senders that generate a large volume of traffic more significant than those sending a few packets.
- The behavioural patterns of senders targeting a broad range of destination ports or protocols are inherently more interesting and important.

Formally, given the global vocabulary $V(t)$ at time window t, for each sending host v, we define the interest score $I_t(v)$ as:

$$I_t(v) = \begin{cases} \beta I_{t-1}(v) + (1-\beta) i_t(v) & \text{if } v \in V(t-1) \\ i_t(v) & \text{if } v \notin V(t-1) \end{cases} \quad (1)$$

where:

$$i_t(v) = \log(\#pkt_v(t)) + \log(\#dst_port_v(t)).$$

$\beta \in (0, 1)$ is the parameter for the exponentially moving average between the current host activity $i_t(v)$ and the host's past interest. $\#pkt_v(t)$ and $\#dst_port_v(t)$ are the numbers of packets and destination ports of sender v observed globally in time window t. These are obtained by aggregating for each sender the interest metrics from different clients, that are the local $\#pkt_{v,s}(t)$ and $\#dst_port_{v,s}(t)$. Given the typical heavy-tailed distribution of the number of packets per sender [2], we rescale both $\#pkt_v(t)$ and $\#dst_port_v(t)$ using the logarithm function to prevent one from dominating the other.

We enforce the upper bound M on vocabulary size $|V|$ and, at each time slot t, we keep senders with a higher interest score to limit the model's size and fit resource constraints for real-world deployment. We perform this eviction policy on the central server when computing $V(t)$ such that $|V(t)| \leq M$.

C. FedScope Implementation

Several frameworks and libraries provide off-the-shelf implementations of FL. We identify the most suitable one to use as a starting point and adapt it to accommodate our new vocabulary synchronisation phase. Given the support and popularity among the scientific community, we select the Flower framework [15], a lightweight open-source Python framework that provides a well-suited FL implementation for real-world deployment.

The library enables the definition of clients, which locally train the models, and a server, which orchestrates the entire FL training process. Clients and the server communicate via the Google high-performance Remote Procedure Call (gRPC) framework for seamless integration on different machines and operating systems. We adapt the design presented in Figure 3 to integrate it within Flower. We add the *Vocabulary Synchronisation* phase, during which the server sends the newly computed vocabulary V and the global model architecture to the clients. Each client then aligns the local model e_s and vocabulary V_s, updates the corpus C_s, and starts the traditional *Model Training* phase, which can repeat for several rounds.

Armed with the updated Flower framework, we implement FedScope as a Python application capable of fully supporting the FL implementation of i-DarkVec[5] in real-world scenarios.[6] We containerise both the server and client components to simplify the distribution and deployment in remote telescopes. We make FedScope freely available to the community.

V. Supervised Learning Experiments

To investigate our research questions, we design specific experiments to compare the performance of the distributed i-DarkVec solutions in a controlled and ideal environment in which we do not need to consider constraints for system implementation. Our goal is to investigate which benefits brought by the collaboration among different telescopes. For this, we compare the performance of embeddings trained i) locally, ii) centralised, and iii) via federated learning in supervised downstream tasks.

We compare the performance of the alternative solutions on the same dataset that we process in an offline manner. This in

[5]Original i-Darkvec implementation is based on the Gensim [16] implementation of Word2Vec. Here, we migrate it to a pure PyTorch implementation for compatibility with the existing Flower frameworks.

[6]While the current Flower-based prototype naturally tolerates temporary client dropout, dynamic client joining is constrained by the orchestration model of the framework and is left for future engineering extensions.

vitro scenario allows us to control and repeat the experiments using the same dataset over and over.[7]

We set all the hyper-parameters consistent with the original i-DarkVec algorithm [5]: We partition our telescope traffic dataset into daily batches. For each batch, we train and update the models for 1 epoch with embedding size $E = 200$. At last, we disable the eviction policy setting $M = \infty$.

A. Dataset

We rely on real-world network traffic collected from several telescopes at different times, and we use, for the classification task, a ground truth retrieved from publicly available lists.

1) Telescopes: The first dataset, called dataset ⓐ, is composed of a /24 telescope s_1 located within our campus network, and a /19 telescope s_2 operated in Brazil. We collected 2 months of traffic in 2021, from May 1 to June 30, during which the /24 telescope observed more than 64 million packets sent by 532 thousand senders, while the /19 telescope recorded over 1.5 billion packets from more than 3 million senders.[8] We filter out senders transmitting fewer than 5 packets per day [5], retaining 15% of the senders in the /24 telescope and 30% in the /19 telescope. To investigate the potential of different telescope sizes, we partition the /19 Brazilian telescope s_2 into smaller telescopes with sizes ranging from /20 to /28. When not specified, the size of the telescope s_2 used in our experiment is a /24 network (same size as s_1), which is sampled from the /19 address range. For all analyses related to embedding performance, we used the first month of data (i.e., from May 1 to May 31), as this period is sufficient for evaluation. For the resource benchmark of FedScope (Section VI-C), we instead used the full two months of data to have a longer analysis period.

We introduce a second dataset, called dataset ⓑ, to evaluate the impact of collaboration among more telescopes. This dataset includes the traffic from 7 network telescopes that we collected using distributed network telescopes [17] during October 2025. This dataset is composed of traffic coming from the same two telescopes that generated ⓐ, and five new telescopes belonging to several Italian and Brazilian universities.

2) Ground Truth—Senders Identification: We consider a supervised downstream task to compare the representativeness of the embeddings. Specifically, we perform *host classification* by assigning senders to known classes characterised by well-known coordinated behaviours. We construct the ground truth based on classes of senders whose coordination is known a priori, leveraging two data sources: (i) the presence of fingerprints of Mirai-like malware observed in received packets [18], and (ii) publicly available information retrieved from online repositories of acknowledged internet scanners,[9] i.e., non-hostile senders performing scanning activities. The final ground truth comprises 13 classes of senders that we observe in the s_1 and s_2 telescopes. These senders are well-known institutions that perform internet-wide scans. They publicly state the set of IP addresses they use for their goal, and their activity is constant over time.

TABLE I
NUMBER OF SENDERS PRESENT IN THE GROUND TRUTH

	Class	# Senders ⓐ	# Senders ⓑ
GT_1	**Mirai-like**	20 934	135 527
GT_2	**Censys**	1 848	4 080
GT_3	**Rapid7**	376	–
GT_4	**ShadowServer**	306	4 389
GT_5	**SecurityTrails**	265	265
GT_6	**Driftnet**	522	796
GT_7	**InternetCensus**	273	1 079
GT_8	**BinaryEdge**	204	7 441
GT_9	**Onyphe**	217	1 728
GT_{10}	**NetSystems**	1 280	–
GT_{11}	**MichiganUni.**	764	1 527
GT_{12}	**Shodan**	47	33 145
GT_{13}	**Internap**	33	–
GT_{14}	**Recyber**	–	885
GT_{15}	**CriminalIP**	–	584
GT_{16}	**Netscout**	–	510
GT_{17}	**LeakIX**	–	106
GT_{18}	**Netsecscan**	–	16
	TOTAL	26 075	192 068

The classification task is highly unbalanced, with thousands of senders in the Mirai-like class, while only hundreds or dozens belong to classes of other acknowledged scanners. The remaining senders are *Unknown*. We build two different ground truths with the same classes, just with different senders, with senders that were active in 2021 or 2025, respectively. Table I details the classes present in the ground truth.

B. Classification Task Methodology

Motivated by the assumption that high-quality embeddings can project senders of the same class (i.e., belonging to one of the coordinated groups of the ground truth) into adjacent regions of the latent space, we perform our downstream classification task relying on a simple k-Nearest-Neighbours (k-NN) classifier. It assigns each sender to the most frequent label through majority voting among the classes within the k nearest neighbours in the embedding space. Thus, the more homogeneous the senders engaged in similar activities in the embedding space, the better the classification performance. We use *cosine distance* to measure distance among embeddings and report the performance of the k-NN classifier considering the F1-score.

We adopt a *Leave-One-Out* validation approach considering all the labelled senders in our collection. For each sender, we compare the label produced by the k-NN classifier with the one in the ground truth. Since we cannot verify the characteristics of the *Unknown* class, we consider them only when computing the k-NN, without reporting classification metrics for the class. We set $k = 7$ for the k-NN classifier as in [5].

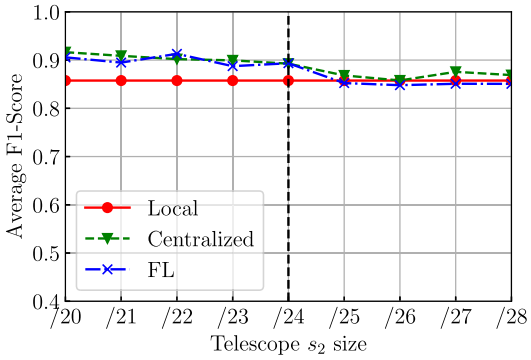
C. Results: Two Telescopes of the Same Size

In Table II we report the per-class F1-Score for the senders active in our dataset ⓐ. We consider a scenario where there are

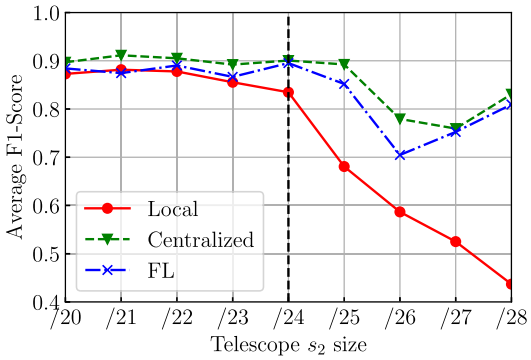
[7]It uses the original i-DarkVec code based on the Gensim library that we run considering the local, centralised, and FL learning approaches.

[8]We make our data available upon request.

[9]https://gitlab.com/mcollins_at_isi/acknowledged_scanners

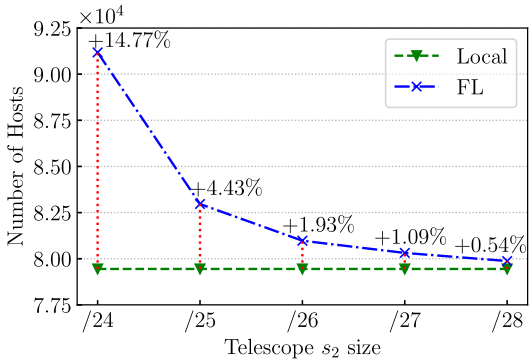


(a) Telescope s_1

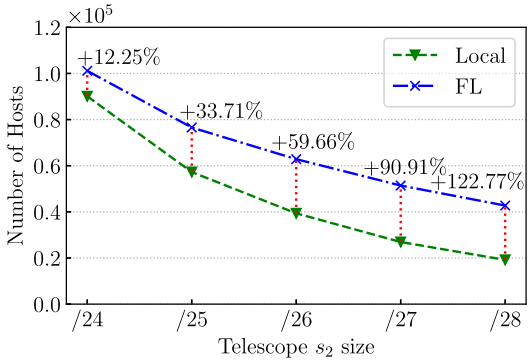


(b) Telescope s_2

Fig. 4. Classification performance of traffic observed by the /24 telescope s_1 (a) and by telescope s_2 (b) using dataset (a). Telescope s_2 varies in size.



(a) Telescope s_1



(b) Telescope s_2

Fig. 5. Coverage in the FL approach with a /24 telescope s_1 and a varying size telescope s_2 (dataset (a)). Note the different y-axis scale. The percentage shows the increase for FL.

2 operators, each with a /24 telescope. We report the metrics for

- Only the senders active in each considered telescope (s_1 and s_2) when the embeddings are generated locally.
- Only the senders active in each considered telescope when the embeddings are generated with the centralised approach.
- Only the senders active in each considered telescope when the embeddings are generated with the FL approach.
- The whole set of senders observed in *both* the networks ($s_1 \cup s_2$) when collaborating with centralised or FL approaches.

Classification performance benefit. Firstly, focus on the senders of the two telescopes separately. Both the Centralised and FL approaches achieve good performance ($\geq$ 0.89 of average F1-Score in both operators), improving the local embeddings generation (F1-Score of 0.86 in s_1 and 0.83 in s_2).

For classes with large support (>250 senders), the local approach suffices to obtain an excellent F1-Score except for the Driftnet class, for which the Centralised (FL) approach results in +0.29 (+0.24) average F1-Score. Classes with small support like Shodan or Internap, benefit from the collaborative approaches gained in both telescopes, up to +0.33 F1-Score for Internap with the Centralised approach in s_2, confirming the usefulness of sharing the knowledge between operators.

When focusing on the full set of observed senders ($s_1 \cup s_2$), the centralised results are comparable with the FL ones, resulting in an average F1-Score of 0.89.

Coverage benefit. As i-DarkVec requires a sufficient amount of traffic to generate an embedding for a sender, all senders that send less than 5 packets are not updated for that batch. However, since scanners and botnets target multiple telescopes in the same observation period with different intensity [2], some of those may result as inactive in a telescope and as active in another one. Thus, building a single global model will extend the embeddings to model more senders independently of which telescope they are active. To gauge this, compare the *Support* figures in Table II. Clearly, the collaborative $s_1 \cup s_2$ setup allows both telescopes to build an embedding for more than 2k additional Mirai senders compared to the ones locally observed. In total, the collaborative approaches extend the set of covered senders by 14.77% in s_1 and 12.25% in s_2, i.e., 11.7k and 11.0k respectively (see Figure 5).

Answer to RQ1: The collaborative setup allows the participating telescopes to both improve the quality of the sender representation (F1-score improves by 3-6%) and to create embeddings for more senders (14% more senders can be classified), improving the visibility on downstream tasks.

D. Results: Two Telescopes of Different Sizes

In Figure 4a we consider the scenario where the two telescopes have different sizes. We keep s_1 fixed and vary

TABLE II

F1-SCORE AND COVERED SENDERS FOR THE 7-NEAREST-NEIGHBOURS CLASSIFIER APPLIED ON THE HOST EMBEDDINGS GENERATED THROUGH DIFFERENT APPROACHES USING DATASET ⓐ. THE BEST RESULTS FOR EACH TASK ARE IN BOLD. FOR THE SAKE OF COMPLETENESS, WE REPORT THE SUPPORT OF THE UNKNOWN CLASS. MINOR DIFFERENCES BETWEEN CENTRALISED AND FEDERATED RESULTS ARE EXPECTED DUE TO REGULARISATION EFFECTS INDUCED BY MODEL AVERAGING UNDER HETEROGENEOUS DATA DISTRIBUTIONS

Class		s_1				s_2				$s_1 \cup s_2$		
		Local	**Centralised**	**FL**	**Support**	**Local**	**Centralised**	**FL**	**Support**	**Centralised**	**FL**	**Support**
GT_1	**Mirai-like**	0.91	**0.94**	**0.94**	10 912	0.90	**0.93**	**0.93**	11 879	**0.91**	**0.91**	14 343
GT_2	**Censys**	0.99	0.99	**1.00**	355	**1.00**	**1.00**	**1.00**	328	**0.99**	**0.99**	355
GT_3	**Rapid7**	**1.00**	**1.00**	**1.00**	344	**1.00**	**1.00**	**1.00**	344	**1.00**	**1.00**	344
GT_4	**ShadowServer**	**1.00**	**1.00**	**1.00**	289	**1.00**	**1.00**	**1.00**	289	**1.00**	**1.00**	289
GT_5	**SecurityTrails**	**1.00**	**1.00**	**1.00**	258	**1.00**	**1.00**	**1.00**	258	**1.00**	**1.00**	258
GT_6	**Driftnet**	0.69	**0.98**	0.95	254	0.74	**0.98**	0.94	253	**0.98**	0.94	254
GT_7	**InternetCensus**	0.99	0.99	**1.00**	252	**1.00**	**1.00**	**1.00**	251	0.99	**1.00**	252
GT_8	**BinaryEdge**	0.44	0.47	**0.50**	164	0.39	0.48	**0.51**	158	0.47	**0.50**	167
GT_9	**Onyphe**	0.96	**0.98**	**0.98**	130	0.97	**0.98**	**0.98**	129	**0.98**	**0.98**	130
GT_{10}	**NetSystems**	0.91	0.89	**0.95**	50	–	–	–	–	0.89	**0.95**	50
GT_{11}	**MichiganUni.**	**0.66**	0.65	0.64	50	0.60	**0.67**	0.64	50	**0.65**	0.64	50
GT_{12}	**Shodan**	0.80	0.82	**0.88**	36	0.80	0.82	**0.89**	36	0.82	**0.88**	36
GT_{13}	**Internap**	0.79	**0.88**	0.79	32	0.63	**0.96**	0.86	24	**0.88**	0.79	32
AVERAGE		0.86	**0.89**	**0.89**	13 126	0.83	**0.90**	**0.90**	13 825	**0.89**	**0.89**	16 560
Unknown		–	–	–	66 319	–	–	–	76 272	–	–	110 553

s_2 size from /20 to /28. The vertical dashed line represents the case of Table II where the two telescopes have equal size.

Classification performance benefit. Figure 4a shows the result for the fixed-size telescope s_1, while Figure 4b for the variable size telescope s_2. When s_2 is larger than s_1 (leftmost part of Figure 4a), the F1-score improves by 3-5% w.r.t the Local embeddings. The same occurs for s_2 when it is smaller than s_1 (rightmost part of Figure 4b). In this second case, the performance gain is more evident given the visibility of s_2 becomes severely limited when it is smaller than a /25 or /26 subnet. Notice that when the s_2 telescope is extremely small (e.g.,/27 or /28), the additional knowledge brought by the s_1 allows s_2 to obtain high-quality embeddings (F1-Score gain >30% for s_2). Conversely, the contribution of s_2 to the knowledge of s_1 offers marginal benefits.

At last, comparing the results obtained by the Centralised and FL approaches, we notice that the FL approach obtains slightly lower performance than the Centralised solution. This is due to the sharing of the filtered corpora $C = C_1 \cup C_2$ rather than building the centralised corpus C directly from the raw data.

Coverage benefit. In Figure 5 we report the extended coverage due to the collaborative approaches. Overall, the sharing of information creates embeddings for more senders than the local approach. Figure 5a highlights that integrating the knowledge of the small telescope provides a limited increase of the coverage –i.e., +15% when collaborating with a telescope of the same size, down to +0.5% when collaborating with a /28 telescope.

Conversely, in Figure 5b we see that the small telescopes strongly benefit from the collaborative setting. Thanks to the broader point of view of the /24 telescope, the coverage increases by 34% up to 123% when s_2 is a /25 or /28 telescope.

E. Multiple Telescopes

Next, we evaluate the performance of the FL approach with multiple telescopes. For this evaluation, we use the dataset ⓑ and report the resulting F1-Score in Figure 6. Look at the first column, which corresponds to the Local scenario. When each operator observes a sufficient amount of traffic to generate meaningful embeddings locally (e.g.,/24 telescopes), the local approach results in good-quality embeddings (average F1-Score of 0.77). Conversely, with telescopes smaller than /24 the telescope lacks sufficient data to produce robust embeddings.

Fig. 6. Average F1-Score when embeddings are generated through the FL collaborative between an increasing number of operators using dataset ⓑ.

Consider now the collaboration scenario. When /24 or /25 telescopes federate themselves into a large telescope, the F1-score improve significantly (bottom two rows). Even /26 telescope can improve their models by collaborating, going from 0.64 to 0.77 F1-score.

Instead, very small telescopes (e.g.,/28) cannot contribute to building a performing model, regardless of the number involved (F1-Score do not improve).

Answer to RQ2: The collaboration is always beneficial. Benefits are asymmetric: smaller telescopes gain substantially,

TABLE III
MACRO F1-SCORE AND COVERED SENDERS FOR THE 7-NEAREST-NEIGHBOURS CLASSIFIER APPLIED ON THE SENDERS EMBEDDINGS GENERATED IN VITRO AND BY FEDSCOPE

	In vitro	FedScope	Support
GT$_1$	0.91	0.94	14 343
GT$_2$	0.99	0.99	355
GT$_3$	1.00	1.00	344
GT$_4$	1.00	1.00	289
GT$_5$	1.00	1.00	258
GT$_6$	0.94	0.96	254
GT$_7$	1.00	1.00	252
GT$_8$	0.50	0.43	167
GT$_9$	0.98	0.99	130
GT$_{10}$	0.95	0.89	50
GT$_{11}$	0.64	0.97	50
GT$_{12}$	0.88	0.94	36
GT$_{13}$	0.79	0.84	32
AVERAGE	0.89	0.92	16 560
Unknown			110 960

while larger ones mainly improve coverage because the traffic they observe suffices to build the embeddings.

VI. FEDSCOPE BENCHMARK

In this section, we run FedScope and compare its performance with that obtained in the simulated environment based on the Gensim implementation of iDarkVec. Next, we present the impact of our eviction model on performance. Finally, we benchmark the computational resources required by FedScope.

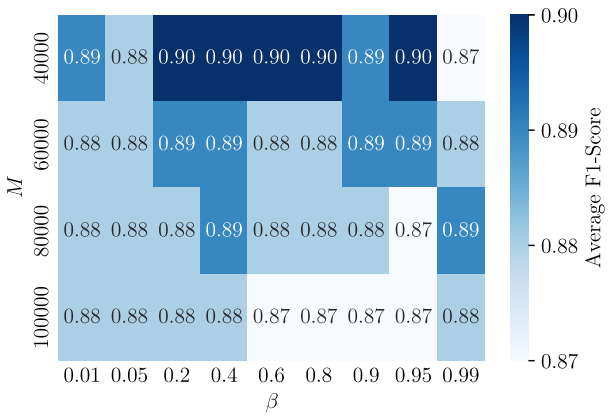
A. FedScope Classification Performance

Table III compares the results of the 7-NN classifier applied to the host embeddings generated by FedScope with those computed in vitro in the previous section using dataset ⓐ. The results indicate that the embeddings produced by FedScope are similar to those from the in vitro environment. The difference in the F1-score can be attributed to the different libraries used to implement i-DarkVec in the two scenarios (the Gensim and PyTorch implementations of Word2Vec).

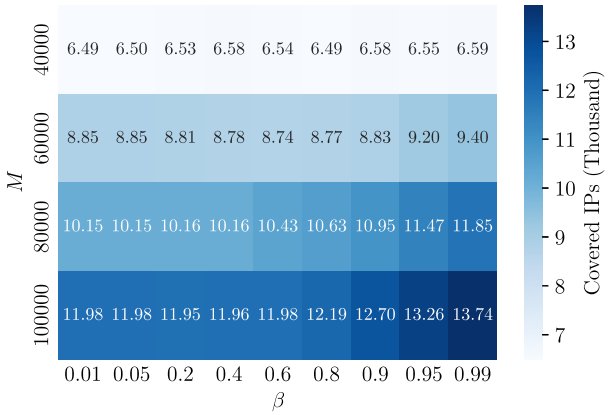
B. Impact of IP Eviction

Here we compare the quality of the embedding when we apply our eviction policy using dataset ⓐ. We set different values of M (from 40 000 to 100 000 and different values of β in the interest score computation (from 0.01 to 0.99). For this analysis, we leverage controlled experiments, ignoring the overhead due to the setup and maintenance of the distributed platform required by FedScope. Figure 7a shows the impact of the eviction: using different values of M and β. We observe little changes in the macro average F1-Score as it is always in the range 0.87 to 0.90.

Figure 7b shows how many senders we can classify using the k-NN. As expected, M strongly influences this number: the larger M is, the more senders we can consider. Conversely, the impact of β is rather limited, especially on large values of M. Reducing the impact of recent senders (larger β) lets the model focus on stable senders. Given our ground truth is mostly made by scanners that are always active and stable, this allows us to consider a larger fraction of ground truth senders.



(a) Macro F1-Score



(b) Covered senders

Fig. 7. Performance of the 7-Nearest-Neighbours classifier applied to sender embeddings of the dataset ⓐ applying the eviction policy.

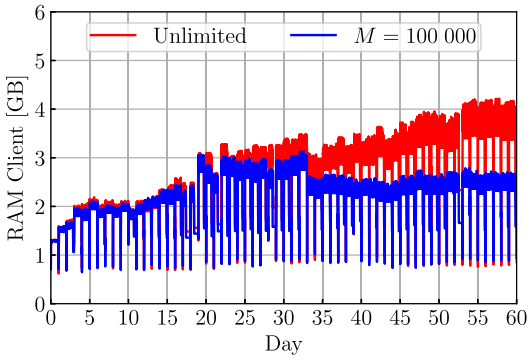
C. Resource Benchmark

Our primary concern is the memory required to execute FedScope as, in actual deployment, some clients may have this resource limited e.g., they are edge devices. Since the model and the vocabulary continuously grow over time as new senders are added, this leads to an increasing memory usage, which may exhaust the client's availability. For this reason, we assess that the eviction mechanism introduced by FedScope is sufficient to cap the memory usage.

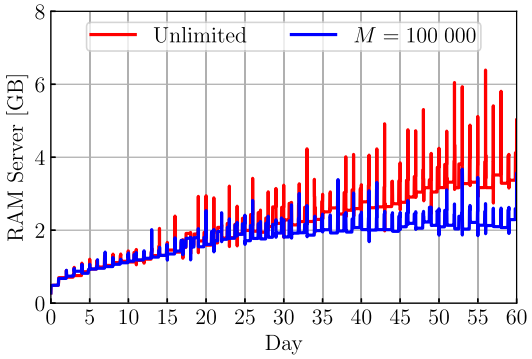
A second important resource to monitor is the cost of communication exchange. A key advantage of FL in general is its reduced data transmission compared to centralized ML, i.e., exchanging model weights is lighter than exchanging the raw dataset. It is important to verify if this still holds for FedScope since its modified FL pipeline includes the additional vocabulary synchronization, which implies more data exchange.

To conduct our RAM and network benchmark, we collected FedScope's statistics using its containerized version executed using Docker Compose with two clients and a central server.

This choice allows us to simulate a real distributed system without the challenges of deploying a fully distributed system. Each client gets the same trace as those used in the previous experiments by preloading traces on the client disks. We then let the system run, during which the clients and the server



(a) Memory usage of one client instance.



(b) Memory usage of the server instance.

Fig. 8. Memory usage of FedScope execution.

execute the FedScope synchronization and model training phases. All communications occur in the virtual network offered by the virtualisation environment.

We run the scenario for two months to clearly show the impact on the resources over a longer period of time. We run FedScope in two scenarios, which differ only by setting $M = \infty$ (unlimited case) or $M = 100\,000$ (constrained cases). We choose $M = 100\,000$ as it is roughly half of the total number of senders observed. With dataset (a), the model grows between 4 and 9 MB per day, depending on the number of unseen IP addresses that now must be added at the input and output layer. In the first scenario, with $M = 100{,}000$, the model reaches a maximum size of 160 MB. In contrast, in the $M = \infty$ scenario, the model grows up to 340 MB.

1) Client's Memory Usage: Figure 8a illustrates the memory usage of one of the two FedScope client instances.

We observe periodic fluctuations in memory usage that correspond to the client's *Model Training Phase* and *Idle Periods*: during training, the memory usage grows due to the processing of the daily dataset. During the idle state, the clients just wait for server update messages.[10] Additionally, memory consumption during model training is strongly influenced by the dataset size for the given day.

In the first $[0, 22]$ days, the eviction policy is inactive as the total number of servers is smaller than M. Both scenarios exhibit similar trends. Starting from day 22 we observe the impact of the eviction policy: in the scenario without limitation, the client's memory continues to grow steadily, reaching up to 4 GB on the last day. In contrast, with $M = 100\,000$, the memory usage stabilises, with memory usage that remains below 3GB.

[10]In these experiments, time is emulated. Each training phase lasts several minutes, while the idle phase ends as soon as the aggregation phase is completed and clients can start processing the data, which is readily available on disk.

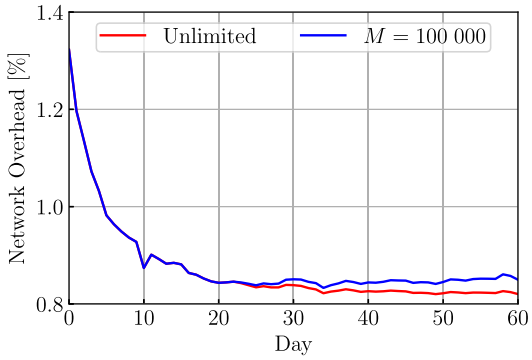


Fig. 9. Overhead of the data exchanged by FedScope compared to standard FL data exchange.

2) Server's Memory Usage: Figure 8 illustrates the effect on FedScope server's memory usage. In this case, the memory peaks correspond to moments when the server receives the local models e_s at the end of each *Model Training Phase*. As before, in the first $[0, 22]$ days we observe the same pattern in both scenarios. When the eviction policy becomes effective, the eviction policy limits the server memory that never exceeds 4 GB. In contrast, with no eviction policy, the memory continues to grow as the dictionary and model size keep growing.

3) Network Impact: The communication cost during FedScope training scales linearly with the model size, as the *Model Training* phase exchanges model embeddings that are in the order of megabytes. In contrast, the vocabulary V and the interest scores $i_t(v)$ account for only a negligible portion of the exchanged data, remaining on the order of kilobytes.

Figure 9 illustrates the overhead introduced by *Vocabulary Synchronization*, normalized to the cost of traditional FedAvg *Model Training*.

Overall, the results indicate that the additional *Vocabulary Synchronization* impact is minimal (smaller than 1.4%). This confirms that sharing the local models e_s causes the largest data exchange. Being the model smaller during the first days, the relative network overhead due to the *Vocabulary Synchronization* is larger. It gets amortised when the model size grows. Here, limiting the model size makes the additional overhead constant (while the unlimited case would artificially keep it reducing).

4) Timing: In our virtualized setup, each training round lasts from a few minutes up to one hour, depending on the size of the daily dataset. The vast majority of this time, i.e., 98%, is spent in the *Model Training* phase. In contrast, the additional *Vocabulary Synchronization* phase requires less than one minute per round, resulting in a negligible impact on the overall execution time.

Answer to RQ3: The FL approach is feasible. The eviction policy lets FedScope effectively control the amount of resources the clients and the server consume and prevents

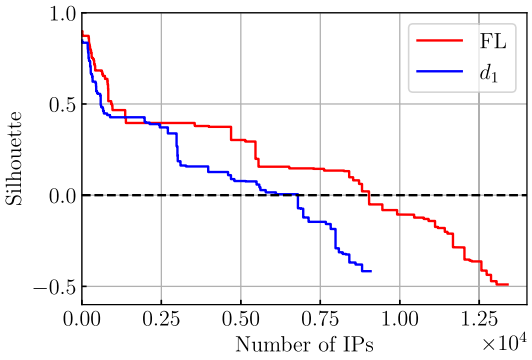


Fig. 10. Silhouette values of senders grouped by cluster ID. Wide horizontal steps indicate large cluster size. Clusters detected from local (s_1) and FL embeddings.

the model from growing very large. Focusing on the most recently active senders allows FedScope to control resource usage without degrading embedding quality, while maintaining comparable coverage.

VII. Unsupervised Clustering Task

We now explore if the benefits brought by the FL setup observed in the supervised task are also present when facing unsupervised problems, using dataset ⓐ. Previous works have shown that clustering hosts in the embedding space allows identifying groups of coordinated senders [5], [9] and novel activities [19]. Here, we apply the same clustering algorithms on top of the embeddings, and we evaluate if the collaborative approach improves the quality of the extracted clusters.

A. Methodology

We adopt the same methodology of [9]: we build a k-nearest-neighbours graph [20], where the vertices represent the senders and an edge exists between two vertices if the senders are k-nearest-neighbours in the embedding space. Then, we cluster senders applying on such a graph the widely known Louvain community detection algorithm [21]. It iteratively detects clusters of nodes by maximising the graph tendency of forming communities of well-isolated and compact clusters of nodes. Finally, we assess the clustering quality by means of the *Silhouette* (Sh) [22], a measure of the clusters' separation in the embedding space. Values close to 1 indicate a perfect separation of clusters; values lower than 0 indicate overlap.

We cluster senders active on the last day of our collection and rely on the same experimental setting of Section V-B where two /24 telescopes s_1 and s_2 collaborate. We compare clusters obtained from embeddings generated locally on s_1 with those obtained from the FL approach $s_1 \cup s_2$.

B. Experimental Results

Figure 10 reports the silhouette and size of the detected clusters. Results show that the FL approach often refines the embedding space, leading to more compact or better separated clusters in many cases.

TABLE IV
Notable Examples of FL Clusters Quality Improvement. Best Silhouette Results are Highlighted in Bold

Label	Cluster	s_1		FL		Δ	
		Size	*Sh*	Size	*Sh*	Size	*Sh*
ShadowServer	C_1	15	0.49	15	**0.65**	0	0.16
	C_2	15	0.55	15	**0.67**	0	0.12
	C_3	12	0.75	12	**0.82**	0	0.07
	C_4	15	0.73	15	**0.79**	0	0.06
	C_5	15	0.78	15	**0.83**	0	0.05
	C_6	16	0.39	16	**0.42**	0	0.03
Censys	C_7	17	0.40	16	**0.71**	-1	0.31
	C_8	48	0.48	48	**0.65**	0	0.17
	C_9	16	0.61	16	**0.78**	0	0.17
	C_{10}	16	0.67	16	**0.76**	0	0.09
	C_{11}	16	0.66	16	**0.75**	0	0.09
	C_{12}	16	0.74	16	**0.80**	0	0.06
	C_{13}	16	0.71	16	**0.74**	0	0.03
Unknown	C_{14}	342	0.08	399	**0.47**	57	0.39
	C_{15}	274	0.34	208	**0.68**	-66	0.34
	C_{16}	692	0.01	449	**0.30**	-243	0.29
	C_{17}	168	-0.12	219	**0.02**	51	0.14
	C_{18}	78	-0.07	93	**0.06**	15	0.12
	C_{19}	165	0.84	195	**0.87**	30	0.03

In Table IV we provide some notable examples of cluster improvement via FL. Despite being an unsupervised task, we rely on the same ground truth of Section V to partially explain some clusters. We match two clusters across the approaches by maximising the Jaccard index on their members.

Firstly, some clusters reflect the prior knowledge on the ground truth, i.e., cluster with ShadowServer ($C_{[1-6]}$) and Censys ($C_{[7-13]}$) senders. These clusters exhibit regular activity patterns containing the same senders in both the local and FL scenario with high quality ($Sh \geq 0.39$). With FL embeddings, these clusters generally demonstrate higher silhouette values compared to the local scenario (+8% of average improvement for ShadowServer clusters and 13% for Censys clusters). Notably, with local embeddings, cluster C_7 contains one "unknown" sender in addition to the 16 senders of Censys. Cooperating through FL allows us to refine such a cluster (Sh of 0.40 locally compared to 0.71 of FL) and to exclude the "unknown" sample, which was wrongly assigned to it.

Clusters containing mostly "unknown" senders ($C_{[14-19]}$) exhibit remarkable changes across scenarios. For example, C_{14} results in an enhanced compactness with FL, i.e., it includes 57 more samples resulting in +39% of silhouette. Conversely, for C_{15} and C_{16}, FL refines the clusters removing >50 senders included with local embeddings. This causes ≈ 30% of silhouette improvement.

The clustering task results help both to identify sub-groups of senders in well-known scanners and to highlight unknown coordinated scanning activities. By manually analysing some of these clusters, we confirm the same groups of senders are involved in the same malicious activities that we found in our previous work [19].

All in all, building a general knowledge of observing hosts' behaviours in multiple vantage points allows for enhancing the quality of the embeddings. FL embeddings of hosts engaged in similar activities are better isolated in the latent space,

allowing for refining the detected clusters and consequently increasing the silhouette. This property is important to support the identification of novel or evolving coordinated activities in future analyses.

VIII. Related Work

A. Telescopes for Security Monitoring

Individual telescopes offer a limited view of attacks [2]. Previous work has shown that distributed and heterogeneous telescopes significantly increase visibility and understanding of attacks [23], [24], [25]. Some studies have used such sources to analyse the increase in scanning activities [26], [27], while others explored various telescope designs, including in-cloud deployments [28] and augmented telescopes with honeypot capabilities [24], [29], [30].

While collaboration among operators and the deployment of diverse telescopes bring benefits, previous works have primarily adopted ad-hoc methods to achieve data sharing among operators. Data anonymisation approaches are typically applied to hide privacy-sensitive information, such as the IP addresses of hosts reaching the telescopes and payload information in packets [31]. While effective in protecting information, these steps reduce the value of shared data.

Unlike these approaches, we first quantify the gains of sharing information among operators. Then, we present FedScope, which allows the learning of shareable host embeddings in a federated manner, enabling knowledge transfer without raw data exchange.

B. Representation Learning for Cybersecurity

Representation learning has become a valuable approach in cybersecurity, particularly for analysing network traffic patterns, including telescope data.

Different works have developed techniques to systematically classify and characterise telescope traffic. The literature [32], [33] has progressed from classic algorithms such as k-means (e.g., for clustering traffic based on features) to pipelines composed of a first stage of representation learning followed by downstream ML tasks. Examples include our methods DarkVec [9] and i-DarkVec [5], which associate IP addresses with specific activity types and identify both known and novel attack patterns. These approaches draw inspiration from NLP techniques, leveraging models like Word2Vec [14] to learn numerical representations of network-related events.

Several other methods exist, including graph-based approaches [6] and autoencoders [13], [34], [35]. Alternative proposals focusing on telescope data, such as IP2Vec [7] and DANTE [8], create embeddings for IP addresses by analysing packet sequences and temporal co-occurrence patterns. These representation learning techniques have shown useful results in extracting patterns from unlabelled datasets.

We advocate the use of federated learning for the exchange of models among telescope operators. While we have validated our proposal using i-DarkVec [5], the approach applies to other representation learning alternatives. However, applying FedScope with other methods requires adaptations to how models used to learn embeddings are consolidated.

C. Applications of Federated Learning in Cybersecurity

Federated learning enables model building without sharing raw data, making it well-suited for cybersecurity applications. Telescope data may include payload information, raising both operational security concerns (e.g., exposing telescope infrastructure) and potential privacy implications. Federated learning addresses these concerns by sharing only model parameters rather than raw traffic traces.

Many works apply federated learning solutions to cybersecurity applications. Examples include intrusion detection models [36], risk intelligence systems [37], IoT network monitoring [38], [39], and DDoS attack detection and classification [40], [41]. These approaches typically focus on single-stage pipelines or end-to-end models tailored to specific problems, as opposed to our focus on integrating a representation learning pipeline.

Only a few recent studies aim to learn federated representations in a self-supervised manner to build general knowledge that can serve multiple purposes [42], [43]. Among these, the work in [44] is closest to ours, as the authors also consider activities observable in telescopes. This work focuses on creating representations for a limited set of malware traffic. In contrast, our work aims to learn comprehensive representations applicable to multiple tasks, providing a broader solution for knowledge exchange among telescope operators.

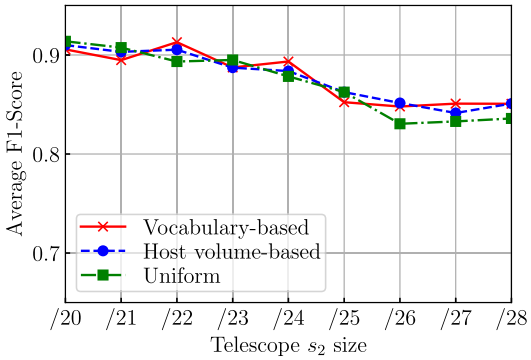
D. Different Federated Learning Paradigms for Cybersecurity

Federated learning constitutes not a single paradigm but a family of related approaches that vary in architecture and aggregation strategy, privacy objectives, and communication patterns [45]. In cybersecurity, researchers most commonly adopt Horizontal Federated Learning (HFL) [46], [47], where participating clients share similar feature spaces but differ in their sample sets, such as network packet header fields [48] and application metadata [37].

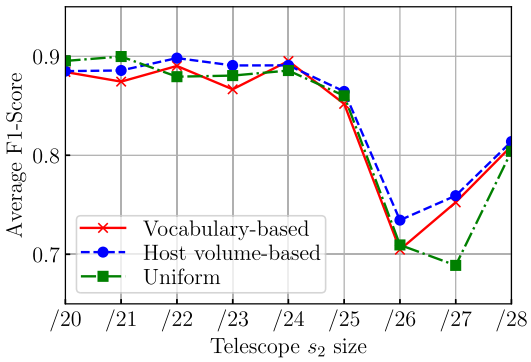
To construct a global model from locally trained models, researchers have proposed various aggregation approaches. These approaches differ in how they combine local weight updates, address data heterogeneity, and mitigate communication or computational constraints [45]. Among them, Federated Averaging (FedAvg) remains the most adopted algorithm; it iteratively aggregates local model parameters by computing a weighted average of client updates. As a result, many federated learning studies use FedAvg as the de facto baseline [37], [40], [41]. We adopt FedAvg for simplicity, while acknowledging that other aggregation strategies could improve the quality of the learned embeddings.

IX. Conclusion

This paper evaluated the aggregation of data from multiple telescopes for learning host embeddings. We showed that the collaboration among telescopes (i) increases the F1-Score in downstream classification tasks (e.g., up to 6%), and (ii) allows the construction of embeddings for a larger number of hosts, thereby increasing coverage in downstream tasks. In unsupervised downstream tasks, combining multiple telescopes helps



(a) Performance on telescope s_1



(b) Performance on telescope s_2

Fig. 11. Classification performance of traffic observed by the /24 darknet d_1 and a varying size darknet d_2, with different weighting schemes.

to uncover more clusters of potentially malicious hosts. The benefits from aggregating data from multiple telescopes are particularly relevant for smaller telescopes, which otherwise would miss less active hosts.

Considering the challenges for sharing raw telescope data, we introduced and evaluated FedScope. We showed that the FL-based approach to learn host embeddings enables similar performance in downstream ML tasks as those obtained with centralised alternatives. However, a carefully designed policy to evict inactive hosts is needed to control the resources of providers contributing to the learning of models as more telescopes aggregate data. We contribute the source code of FedScope in the hope that it will foster collaboration among different telescope providers.

Future directions include the deployment of FedScope in practice and its extension to other scenarios, e.g., to learn host embeddings from production networks and other sources. Such deployments would enable more comprehensive threat monitoring across diverse networks. However, they will require an analysis of possible data leakage via learned models, as well as establishing appropriate privacy-preserving mechanisms to protect sensitive information.

APPENDIX

A. Alternative Weighting Schemes

For completeness, we test two alternative weighting schemes beyond the vocabulary-based approach.

1) Host Volume-Based Weights: Embeddings are updated at different frequencies during training based on how often each IP address appears in the corpus. Following this intuition, we assign each host a different weight proportional to its activity level. Formally, we define the final federated embedding for a host $v \in \bigcup_s V_s$ as:

$$e(v) = \frac{\sum_{s:v \in V_s} \#pkt_{v,s} \cdot e_s(v)}{\sum_{s:v \in V_s} \#pkt_{v,s}}$$

where $\#pkt_{v,s} \in \mathbb{R}$ is the number of packets that v sends to telescope s.

2) Uniform Weight: For comparison, we test averaging the embeddings with uniform weights. We define the final federated embedding for a host $v \in \bigcup_s V_s$ as:

$$e(v) = \frac{\sum_{s:v \in V_s} e_s(v)}{|\{\, s \mid v \in V_s \,\}|}$$

where $|\{\, s \mid v \in V_s \,\}|$ denotes the number of telescopes contacted by v.

3) Impact of Weighting Schemes: Figure 11 explores the impact of different weighting schemes in the FL approach when two providers are involved. Telescope s_1 is a /24, while s_2 size varies.

The weighting scheme shows minimal effects on embedding quality in terms of classification performance. When both providers make substantial contributions (size $\geq$/25), aggregating the models without weighting yields comparable performance (average F1-Score> 0.85). While differences are observable when aggregating telescopes with highly unbalanced sizes, the overall impact on downstream performance remains limited across all tested schemes.

Overall, the weighting scheme is not a critical component–any scheme that approximately reflects the relative sizes of different telescopes proves adequate.

ACKNOWLEDGMENT

This manuscript reflects only the authors' views and opinions and the Ministry cannot be considered responsible for them.

REFERENCES

[1] C. Fachkha and M. Debbabi, "Darknet as a source of cyber intelligence: Survey, taxonomy, and characterization," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 2, pp. 1197–1227, 2nd Quart., 2016.

[2] F. Soro, I. Drago, M. Trevisan, M. Mellia, J. Ceron, and J. J. Santanna, "Are darknets all the same? On darknet visibility for security monitoring," in *Proc. IEEE Int. Symp. Local Metrop. Area Netw. (LANMAN)*, Jul. 2019, pp. 1–6.

[3] Y. Liu et al., "Graph self-supervised learning: A survey," *IEEE Trans. Knowl. Data Eng.*, vol. 35, no. 6, pp. 5879–5900, Jun. 2023.

[4] Z. B. Houidi, R. Azorin, M. Gallo, A. Finamore, and D. Rossi, "Towards a systematic multi-modal representation learning for network data," in *Proc. 21st ACM Workshop Hot Topics Netw.*, Nov. 2022, pp. 181–187.

[5] L. Gioacchini, L. Vassio, M. Mellia, I. Drago, Z. B. Houidi, and D. Rossi, "I-DarkVec: Incremental embeddings for darknet traffic analysis," *ACM Trans. Internet Technol.*, vol. 23, no. 3, pp. 1–28, Aug. 2023.

[6] L. Gioacchini, A. Cavallo, M. Mellia, and L. Vassio, "Exploring temporal GNN embeddings for darknet traffic analysis," in *Proc. 2nd Graph Neural Netw. Workshop*, Dec. 2023, pp. 31–36.

[7] M. Ring, A. Dallmann, D. Landes, and A. Hotho, "IP2 Vec: Learning similarities between IP addresses," in *Proc. IEEE Int. Conf. Data Mining Workshops (ICDMW)*, Nov. 2017, pp. 657–666.

[8] D. Cohen et al., "DANTE: A framework for mining and monitoring darknet traffic," in *Proc. Eur. Symp. Res. Comput. Secur.*, 2020, pp. 88–109.

[9] L. Gioacchini, L. Vassio, M. Mellia, I. Drago, Z. B. Houidi, and D. Rossi, "DarkVec: Automatic analysis of darknet traffic with word embeddings," in *Proc. 17th Int. Conf. Emerg. Netw. Exp. Technol.*, Dec. 2021, pp. 76–89.

[10] M. Alazab, S. R. M. Priya, M. Parimala, P. K. R. Maddikunta, T. R. Gadekallu, and Q.-V. Pham, "Federated learning for cybersecurity: Concepts, challenges, and future directions," *IEEE Trans. Ind. Informat.*, vol. 18, no. 5, pp. 3501–3509, May 2022.

[11] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. Y. Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. Artif. Intell. Statist.*, 2016, pp. 1273–1282.

[12] K. Huang, L. Gioacchini, M. Mellia, and L. Vassio, "Incremental federated host embeddings for network telescopes traffic analysis," in *Proc. IEEE 44th Int. Conf. Distrib. Comput. Syst. Workshops (ICDCSW)*, Jul. 2024, pp. 41–46.

[13] M. Kallitsis, R. Prajapati, V. Honavar, D. Wu, and J. Yen, "Detecting and interpreting changes in scanning behavior in large network telescopes," *IEEE Trans. Inf. Forensics Security*, vol. 17, pp. 3611–3625, 2022.

[14] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," 2013, *arXiv:1301.3781*.

[15] D. J. Beutel et al., "Flower: A friendly federated learning research framework," 2020, *arXiv:2007.14390*.

[16] R. Rehurek and P. Sojka, "Gensim–Python framework for vector space modelling," NLP Centre, Fac. Inform.,Masaryk Univ., Brno, Czech Republic, Tech. Rep., 2011, vol. 3, no. 2.

[17] A. Sordello et al., "Holoscope: Open and lightweight distributed telescope & honeypot platform," 2025, *arXiv:2512.19842*.

[18] J. M. Ceron, K. Steding-Jessen, C. Hoepers, L. Z. Granville, and C. B. Margi, "Improving IoT botnet investigation using an adaptive network layer," *Sensors*, vol. 19, no. 3, p. 727, Feb. 2019.

[19] K. Huang, L. Gioacchini, M. Mellia, and L. Vassio, "Dynamic cluster analysis to detect and track novelty in network telescopes," in *Proc. IEEE Eur. Symp. Secur. Privacy Workshops*, Jul. 2024, pp. 287–296.

[20] D. Eppstein, M. S. Paterson, and F. F. Yao, "On nearest-neighbor graphs," *Discrete Comput. Geometry*, vol. 17, pp. 263–282, Apr. 1997.

[21] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, "Fast unfolding of communities in large networks," *J. Stat. Mechanics, Theory Exp.*, vol. 2008, no. 10, Oct. 2008, Art. no. P10008.

[22] P. J. Rousseeuw, "Silhouettes: A graphical aid to the interpretation and validation of cluster analysis," *J. Comput. Appl. Math.*, vol. 20, pp. 53–65, Nov. 1987.

[23] C. Munteanu, S. J. Saidi, O. Gasser, G. Smaragdakis, and A. Feldmann, "Fifteen months in the life of a honeyfarm," in *Proc. ACM Internet Meas. Conf.*, Oct. 2023, pp. 282–296.

[24] L. Izhikevich, M. Tran, M. Kallitsis, A. Fass, and Z. Durumeric, "Cloud watching: Understanding attacks against cloud-hosted services," in *Proc. ACM Internet Meas. Conf.*, Oct. 2023, pp. 313–327.

[25] P. Richter and A. Berger, "Scanning the scanners: Sensing the internet from a massively distributed network telescope," in *Proc. Internet Meas. Conf.*, Oct. 2019, pp. 144–157.

[26] Z. Durumeric, M. Bailey, and J. A. Halderman, "An internet-wide view of internet-wide scanning," in *Proc. 23rd USENIX Secur. Symp.*, 2014, pp. 65–78.

[27] H. Griffioen, G. Koursiounis, G. Smaragdakis, and C. Doerr, "Have you SYN me? Characterizing ten years of internet scanning," in *Proc. ACM Internet Meas. Conf.*, Nov. 2024, pp. 149–164.

[28] E. Pauley, P. Barford, and P. McDaniel, "DScope: A Cloud-Native internet telescope," in *Proc. 32nd USENIX Secur. Symp.*, Aug. 2023, pp. 5989–6006.

[29] R. Hiesgen, M. Nawrocki, A. King, A. Dainotti, C. T. Schmidt, and M. Wählisch, "Spoki: Unveiling a new wave of scanners through a reactive network telescope," in *Proc. 31st USENIX Secur. Symp.*, 2022, pp. 431–448.

[30] F. Soro et al., "Enlightening the darknets: Augmenting darknet visibility with active probes," *IEEE Trans. Netw. Service Manage.*, vol. 20, no. 4, pp. 5012–5025, Dec. 2023.

[31] C. Han, J. Takeuchi, T. Takahashi, and D. Inoue, "Dark-TRACER: Early detection framework for malware activity based on anomalous spatiotemporal patterns," *IEEE Access*, vol. 10, pp. 13038–13058, 2022.

[32] C. Fachkha, E. Bou-Harb, and M. Debbabi, "Inferring distributed reflection denial of service attacks from darknet," *Comput. Commun.*, vol. 62, pp. 59–71, May 2015.

[33] M. Jonker, A. King, J. Krupp, C. Rossow, A. Sperotto, and A. Dainotti, "Millions of targets under attack: A macroscopic characterization of the DoS ecosystem," in *Proc. Internet Meas. Conf.*, Nov. 2017, pp. 100–113.

[34] L. Gioacchini, I. Drago, M. Mellia, Z. Ben Houidi, and D. Rossi, "Generic multi-modal representation learning for network traffic analysis," 2024, *arXiv:2405.02649*.

[35] R. Zhao et al., "Yet another traffic classifier: A masked autoencoder based traffic transformer with multi-level flow representation," in *Proc. AAAI Conf. Artif. Intell.*, 2023, vol. 37, no. 4, pp. 5420–5427.

[36] Z. Chen, N. Lv, P. Liu, Y. Fang, K. Chen, and W. Pan, "Intrusion detection for wireless edge networks based on federated learning," *IEEE Access*, vol. 8, pp. 217463–217472, 2020.

[37] H. Fereidooni, A. Dmitrienko, P. Rieger, M. Miettinen, A.-R. Sadeghi, and F. Madlener, "FedCRI: Federated mobile cyber-risk intelligence," in *Proc. Netw. Distrib. Syst. Secur. Symp.*, 2022.

[38] D. C. Nguyen, M. Ding, P. N. Pathirana, A. Seneviratne, J. Li, and H. Vincent Poor, "Federated learning for Internet of Things: A comprehensive survey," *IEEE Commun. Surveys Tuts.*, vol. 23, no. 3, pp. 1622–1658, 3rd Quart., 2021.

[39] B. Ghimire and D. B. Rawat, "Recent advances on federated learning for cybersecurity and cybersecurity for federated learning for Internet of Things," *IEEE Internet Things J.*, vol. 9, no. 11, pp. 8229–8249, Jun. 2022.

[40] Q. Tian, C. Guang, C. Wenchao, and W. Si, "A lightweight residual networks framework for DDoS attack classification based on federated learning," in *Proc. IEEE Conf. Comput. Commun. Workshops (INFOCOM WKSHPS)*, May 2021, pp. 1–6.

[41] V. Pourahmadi, H. A. Alameddine, M. A. Salahuddin, and R. Boutaba, "Spotting anomalies at the edge: Outlier exposure-based cross-silo federated learning for DDoS detection," *IEEE Trans. Dependable Secure Comput.*, vol. 20, no. 5, pp. 4002–4015, Sep. 2023.

[42] T. Ongun, S. Boboila, A. Oprea, T. Eliassi-Rad, J. Hiser, and J. Davidson, "CELEST: Federated learning for globally coordinated threat detection," 2022, *arXiv:2205.11459*.

[43] B. van Berlo, A. Saeed, and T. Ozcelebi, "Towards federated unsupervised representation learning," in *Proc. 3rd ACM Int. Workshop Edge Syst., Analytics Netw.*, Apr. 2020, pp. 31–36.

[44] Y.-W. Chang, H.-Y. Chen, C. Han, T. Morikawa, T. Takahashi, and T.-N. Lin, "FINISH: Efficient and scalable NMF-based federated learning for detecting malware activities," *IEEE Trans. Emerg. Topics Comput.*, vol. 11, no. 4, pp. 934–949, Oct. 2023.

[45] B. S. Guendouzi, S. Ouchani, H. E. Assaad, and M. E. Zaher, "A systematic review of federated learning: Challenges, aggregation methods, and development tools," *J. Netw. Comput. Appl.*, vol. 220, Nov. 2023, Art. no. 103714. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1084804523001339

[46] H. Zhang, J. Ye, W. Huang, X. Liu, and J. Gu, "Survey of federated learning in intrusion detection," *J. Parallel Distrib. Comput.*, vol. 195, Jan. 2024, Art. no. 104976. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0743731524001400

[47] J. L. Hernandez-Ramos et al., "Intrusion detection based on federated learning: A systematic review," *ACM Comput. Surveys*, vol. 57, no. 12, pp. 1–65, Jul. 2025, doi: 10.1145/3731596.

[48] H. Mun and Y. Lee, "Internet traffic classification with federated learning," *Electronics*, vol. 10, no. 1, p. 27, Dec. 2020. [Online]. Available: https://www.mdpi.com/2079-9292/10/1/27

Kai Huang received the M.Sc. degree in ICT for smart societies. He is currently pursuing the Ph.D. degree in computer and control engineering with Politecnico di Torino. His research interests include the intersection of data science and network measurements.

Andrea Sordello received the master's degree in computer engineering from Politecnico di Torino in 2024, where he is currently pursuing the Ph.D. degree in computer and control engineering with the SmartData@PoliTO Research Center. His current research interests include network traffic analysis.

Rodolfo Vieira Valentim is currently a Research Assistant with the University of Turin (UniTo), specializing in machine learning and cybersecurity. His research interests include leveraging data-driven models to enhance security analysis and threat detection.

Luca Vassio received the M.Sc. degree (cum laude) in mathematical modeling and the Ph.D. degree (cum laude) in telecommunication engineering. He is currently an Associate Professor with Politecnico di Torino. His research interests include big data analytics to machine learning and optimization approaches, including GNNs. He applies them to internet measurements, social networks, and mobility. He collaborated, among others, with MIT, Bell Laboratories, and GE Aviation.

Idilio Drago received the Ph.D. degree from the University of Twente, The Netherlands. He is currently an Associate Professor with the University of Turin, Italy. His research interests include network security, machine learning, and internet measurements. He is particularly interested in how machine learning can help extract knowledge from network data and secure the networks. He was awarded the IETF/IRTF Applied Networking Research Prize.

Marco Mellia (Fellow, IEEE) is currently a Full Professor with PoliTO, Italy. He coordinates the SmartData@PoliTO Centre, an interdisciplinary laboratory focusing on machine learning, data science, and applications to network management and cybersecurity. He has co-authored over 250 papers published in international journals and leading conferences. He won the IRTF ANR Prize at IETF-88 and many best paper awards. He is the EiC of *Proceedings of the ACM on Networking*.

Open Access funding provided by 'Politecnico di Torino' within the CRUI CARE Agreement