On Traffic Matrix Estimation via Super-Resolution and Federated Learning

Lorenzo Pappone *Student Member, IEEE,*, Alessio Sacco, *Member, IEEE,* Flavio Esposito, *Member, IEEE,*

***Abstract*—Network traffic telemetry plays a crucial role in the management of modern networks. Estimation of the network traffic matrix is a widely recognized problem whose solutions can span a diverse set of applications. Current approaches to traffic matrix inference through statistical methods often rely on assumptions about the matrix structure, which may be invalid in certain scenarios. Data-driven methods, instead, often use detailed information about the network topology that may be unavailable or impractical to collect. To overcome these challenges, we propose a super-resolution technique for traffic matrix inference that leverages coarser measurements to predict fine-grained network traffic. Furthermore, we devise a distributed learning procedure and adapt our model to scenarios of partial network visibility. Our experiments on real network traces demonstrate that the proposed approach can infer fine-grained network traffic with high precision. Moreover, we prove that our distributed approach improves the inference accuracy with respect to its centralized counterpart, significantly lowering the training time, even in scenarios with partial network knowledge.**

***Index Terms*—traffic estimation, super resolution, deep learning, federated learning**

I. Introduction

The improvement of computer network efficiency involves complex analytics and time-sensitive troubleshooting. However, collect the correct volume of network traffic at the optimal time remains a challenging task, especially considering the vast scale of modern networks and the presence of hidden network locations. The analysis of network traffic data to obtain insights about the behavior of a distributed system is a fundamental area of study. There are numerous methods for developing network models [2], [3], [4], but there are many challenges and costs in the measurement of network traffic [5], [6]. Much of this difficulty comes from a large number of nodes at which the traffic must be measured, as well as the amount of data that must be collected. For example, a single high-speed network interface within a given region could generate hundreds of gigabytes of (unsampled) flow statistics per day if fully utilized, while the whole network might generate several gigabytes of simple network management protocol (SNMP) statistics per day [7], [8]. Many aspects of the Internet are characterized by decreasing visibility of important network properties, which is in tension with the Internet's role as critical infrastructure [9]. Researchers have sought out ways to infer network patterns from the least amount of data stored and collected [10], [11], sometimes because such data is not available, since, before the (performance) problem arose, the frequency of measurements was kept low.

Super-resolution to reconstruct a more accurate, representation of network traffic. A goal of this work is to explore how a computer vision technique called Super-Resolution (SR) can help infer visibility during network monitoring and telemetry operations of a distributed system. In particular, we focus on the inference of fine-grain network traffic details only using aggregate measurements.

In our approach, we start with a Low Resolution (LR) traffic matrix, represented as a heat-map. This LR matrix is a coarse-grained measurement derived from the original fine-grained traffic matrix, which we refer to as the High Resolution (HR) traffic matrix. To construct the LR matrix, we consider different sub-networks of the original network, grouping hosts in various ways depending on the dataset under test. Using an SR technique, we then reconstruct a more accurate, HR representation of the traffic that we would have observed on a given subnetwork, aiming to minimize the inference error. Our method employs a supervised learning approach, training a deep neural network with numerous HR traffic matrices paired with their corresponding LR versions. Once trained, this neural network is capable of inferring the high-resolution traffic matrix when only its LR counterpart is available.

We also apply the concept of Federated Learning (FL) [12] to harness the capabilities of multiple network agents while simultaneously minimizing communication overhead among these agents. This methodology is particularly suited for addressing issues of partial observability.

Federated learning for partial network visibility. In many real-world network scenarios, full network observability is often unattainable due to various constraints. For instance, in multi-domain networks spanning different administrative entities, privacy concerns or regulatory restrictions may limit data sharing. Similarly, in large-scale distributed systems or edge computing environments, centralized data collection can be impractical due to bandwidth limitations or latency issues. Other examples include IoT networks with resource-constrained devices or cloud-based services with geographically dispersed data centers [13].

Our FL-based approach can infer fine-grained network measurements when complete network visibility is not directly available. Our method leverages partial knowledge from distributed clients, each possessing visibility into a sub-network. The federated clients collaboratively train a global super-resolution model without sharing raw data, preserving privacy

This paper is an extended version of [1].

Lorenzo Pappone and Flavio Esposito are from the Department of Computer Science, Saint Louis University, USA (e-mail: lorenzo.pappone@slu.edu, flavio.esposito@slu.edu)

Alessio Sacco is with DAUIN, Politecnico di Torino, 10129 Turin, Italy (e-mail: alessio_sacco@polito.it).

and reducing data transfer overhead. The resulting global model, updated with aggregated insights from various sub-networks, is then used to predict the entire traffic matrix.

Contributions and paper organization. Inspired by the field of image processing and generative methods, in general, and by the super resolution [14], in particular, we propose an algorithm for traffic matrix inference that we call *Enhanced Deep Traffic Matrix Super-resolution*(EDTMSR).

We expand on our initial work [1] by thoroughly analyzing the performance of our distributed traffic matrix estimation technique in different scenarios: We propose a new FL methodology and formalize a new traffic matrix prediction problem to deal with partial network knowledge scenarios. We further evaluate our model in terms of the estimated traffic matrix quality by using two new metrics commonly used in computer vision. We study how the performance of our EDTMSR is sensitive to variations of visibility overlaps, that is, different traffic matrix portions are distributed among clients during the training phase. We also extensively evaluate how the distributed learning performance is affected by different percentages of client participation in the training process.

We validate our model against a datacenter traffic (i.e., Meta dataset [15]) and a WAN traffic (i.e., GEANT dataset [16]), comparing the centralized and distributed variants with ML and DL baselines. We then demonstrate how the use of the federated learning technique makes distributed training of the EDTMSR model more robust to possible clients' failures and show how the federated learning process achieves competitive results w.r.t. its centralized counterpart while performing fewer model training rounds. Finally, we evaluate our distributed learning algorithm in terms of training loss with different scale factors.

The rest of the paper is organized as follows: in Section II we discuss the related work and in Section III we formulate the traffic matrix estimation problem. Section IV details our proposed model and its federated learning counterpart. In Section V we describe our experimental setting and, then, in Section VI we discuss our evaluation results, showing the benefits of EDTMSR. Finally, in Section VII we conclude the paper.

II. Related Work

Traffic Inference and Telemetry. Being able to infer which are the routes that pass through the network of an operator is a complex but profitable task [17], with applications to traffic engineering, performance analysis, capacity planning, traffic loads change and its causes, network security, and business intelligence [11], [18]. Related work in this area can be divided into two main categories: those that consider a TM a purely-spatial concept [19], [20] and those for which a TM is a time series of TMs [21], [22], [23]. The former problem is similar to matrix completion problems [24], [11], the latter approach implies a strong correlation between TMs over time. These data are low effective rank, as shown in [21], i.e., there are strong correlations between columns (or rows), such that a measured TM can be approximated by a matrix having a relatively small rank. The results of the methods using these low-rank data show a strong dependence on this temporal correlation.

However, spatio-temporal compressive sensing cannot work with the highly non-linear relationships between high- and low-resolution network traffic samples, as they expect that linear relationships subsist between sparse traffic and inference matrices. In addition, many of the proposed traffic inference methods require the collection and analysis of fine-grained traffic matrices, as well as detailed information on the topology and structure of the network under consideration. Our method is able to overcome these limitations. We learn the mapping between coarse-grained metrics (low-resolution matrices) and their fine-grained (high-resolution) counterpart by using our super-resolution approach.

Super-resolution Techniques. Super-resolution is a generative process that represents a fundamental tool for a wide range of real-world applications, such as medical imaging [25], [26], security [27], [28], surveillance, and other computer vision tasks [29], [30]. The literature on Single Image Super-Resolution (SISR) is vast [31]. Early approaches to the super-resolution problem use interpolation techniques based on sampling theory [32], [33], [34], while others used image statistics to improve the results in image reconstruction [35], [36], [37].

Super-resolution approaches are typicalle applied to images and rely on different techniques, e.g., neighbor embedding [38], [39], [40], [41], sparse representation methods [42], [43], [44], [45], [46], clustered patch-spaces [47], image self-similarities [48], [49], and geometric transformations of patches in order to increase the number of examples and perform data augmentation [50]. The first super-resolution algorithm was proposed by Dong et al. [51], who demonstrated how CNN can be exploited to overcome the state-of-art image reconstruction accuracy. Other SR approaches have exploited a Generative Adversarial Network (GAN) [52] to achieve promising results [53].

In [54], the authors used mobile network traffic maps on a city scale and applied, for the first time, a super-resolution based approach to learn the telecommunication power patterns from fewer measurement probes. They utilized super-resolution in networking for the creation of physical mobile network matrices, characterized by the presence of a spatial relationship between the cells of the matrices themselves.

Our work, on the other hand, focuses on the traffic matrices generated within any network. Given the nature of Internet traffic flows, these traffic matrices *lack physical location association*. We created a *location-independent* network traffic inference method. Furthermore, we trained the proposed EDTMSR model both in a centralized and distributed way, with federated learning.

Federated Learning. Machine learning models are often built from the collection of extensive datasets to enable the detection, classification, and prediction of future events. Due to bandwidth, storage, and other business operations and non-technical concerns, it is often impractical to send all the data to a centralized location.

Analyzing and learning from data distributed among many clients without exposing data, is a goal pursued by many research communities. In addition to methods based on encryption [55], several publications have had the goal of preserving privacy by keeping data locally on clients and using a centralized server for training [56]. While some form of privacy may be guaranteed with our algorithm, we do not claim that privacy is a design principle, as our approach alone (similar to other federated learning architectures) is not resilient to privacy attacks [57], [58]. However, we conduct experiments to show how our model trained via federated learning is robust to the varying participation of federated clients in the training process, and that the model trained with federated learning is able to outperform the results obtainable with the traditional centralized training methods.

III. Problem Definition

In our work, we consider the traffic matrix (TM) as the main data unit for our experiments. In network management, a TM is often represented as a snapshot of the amount of traffic exchanged between all network hosts at a specific time. Formally, the element $TM(i,j)$ of a TM describes the volume of traffic, expressed in bytes, measured between source i and destination j. We express a TM as a 2-dimensional array $TM \in \mathbb{R}^N \times \mathbb{R}^N$, where N is the number of nodes in the network. With $TM(i,j;t)$, we denote the traffic from node i to j, averaged over the time interval $[t, t+\Delta t]$.

Traffic Matrix Super-Resolution. The objective of our Traffic Matrix Super-Resolution problem is to infer fine-grained (high-resolution) network traffic data, using as a starting point measurements collected at a lower spatial resolution in the network. In our approach, the low-resolution (LR) traffic matrix TM^{LR} represents spatially coarse-grained data, which are aggregated traffic measurements over larger network areas or clusters. For example, this could be pod-level traffic in a datacenter or regional traffic in a WAN, where multiple individual nodes are grouped together. This LR matrix can be defined as the output of an aggregation process:

$$TM^{LR} = A(TM^{HR}; \alpha), \tag{1}$$

where A is an aggregation mapping function, TM^{HR} is the High-Resolution (HR) traffic matrix representing fine-grained measurements, and α denotes the parameters of the aggregation process, such as the grouping strategy or aggregation level. Instead of modeling the degradation function as a downsampling operation as in [1], we directly work with the aggregated measurements.

Remark. *Although other convolutional models exploit temporal relation between subsequent matrices during our training phase, e.g., in [54], considering both space and time would lead to significant growth of the problem size and the costs of a federated training. Therefore, we only consider the spatial dimension.*

The goal is to recover an approximation $\widehat{TM}^{HR}$ of the ground truth HR traffic matrix, denoted as TM^{HR}, starting from its LR version TM^{LR}:

$$\widehat{TM}^{HR} = F(TM^{LR}; \theta), \tag{2}$$

where F is the super-resolution model and θ represents its parameter vector. The recovered HR traffic matrix $\widehat{TM}^{HR}$ is also denoted as *super-resolved traffic matrix* TM^{SR}.

Our traffic matrix inference task is hence modeled by the following optimization problem:

$$\widehat{\theta} = \underset{\theta}{\operatorname{argmin}}\, \mathcal{L}(\widehat{TM}^{HR}, TM^{HR}) + \lambda\Phi(\theta), \tag{3}$$

where $\mathcal{L}(\widehat{TM}^{HR}, TM^{HR})$ is the loss function between the predicted HR traffic matrix $\widehat{TM}^{HR}$ and the ground truth traffic matrix TM^{HR}, $\Phi(\theta)$ is a regularization term and λ is the trade-off parameter.

Distributed Traffic Matrix Inference. The traditional approach to distributed deep learning often requires data from distributed devices to be sent to a centralized server for model training, using a Distributed Stochastic Gradient Descent (D-SGD) algorithm. This procedure can present significant risks in terms of data privacy. Moreover, a complete view of the data is not always guaranteed. In our work, we integrate our model with a federated deep learning approach that addresses these issues by enabling distributed devices to collaboratively train a global model while retaining their data locally. With this strategy, we distribute either overlapping or non-overlapping portions of the traffic matrix across different clients, each representing a segment of the network.

We then aim to reconstruct the original matrix by utilizing the model parameters learned by these distributed agents. In our case, each client will receive equally-sized portions of the original matrix $\frac{p}{\sqrt{N}} \times \frac{q}{\sqrt{N}}$ (assuming N is a perfect square for simplicity). Let $\sqrt{N} = k$, then each client n with $1 \le n \le N$ would receive:

$$\begin{aligned} TM_{(i,j)} = TM\bigg(&\left[\frac{(i-1)p}{k} + 1, \frac{ip}{k}\right], \\ &\left[\frac{(j-1)q}{k} + 1, \frac{jq}{k}\right]\bigg) \end{aligned} \tag{4}$$

where $n = (i-1) \times k + j$.

We then formulate our distributed learning problem as a three-phase procedure: Initialization, Aggregation, and Update. During Initialization, each client receives a pretrained global model ω_t from the aggregator. The clients then train this model using their local data D_k. During the Aggregation phase, edge devices send their local gradients to the cloud aggregator. The local loss function for each edge device k is given by:

$$\min_{x \in \mathbb{R}^d} F_k(x) = \frac{1}{D_k} \sum_{i \in D_k} E_{z_i \sim D_k} f(x; z_i) + \lambda h(x) \tag{5}$$

Here, $f(\cdot;\cdot)$ is the local loss function, λ is a regularization term, and $h(x)$ is a regularizer function.

In the Update phase, the aggregator uses the Federated Averaging (FedAVG) algorithm [12] to update the global model W_{t+1} for the next iteration as follows:

$$W_{t+1} = W_t + \alpha \sum_{c=1}^{C} p_c (W_c^{t+1} - W_t) \tag{6}$$

where C is the total number of clients, p_c represents the relative importance of client c (which could be based on factors such as data volume or quality), and α is a learning rate parameter.

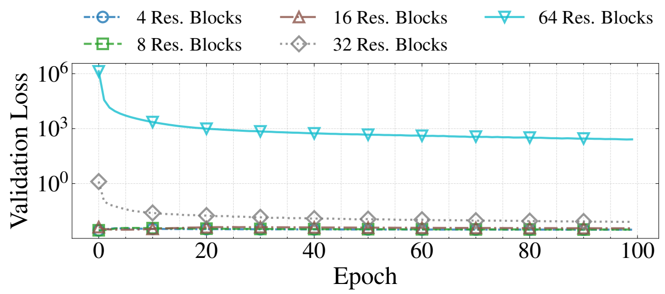


Fig. 1: Comparison of validation loss varying the number of residual blocks in the super-resolution model.

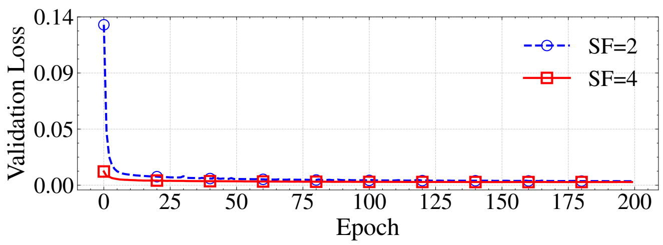


Fig. 2: Comparison of validation loss for scale factors (SF) x2 and x4 on the GÉANT dataset. Model with scale factor x4 is pre-trained on traffic matrix with scale factor x2.

IV. Our Solution: Super-Resolution for Traffic Matrix Inference

In this section, we describe how we solve the TMSR problem presenting the details of our *Enhanced Deep Traffic Matrix Super-Resolution Network (EDTMSR)*. We also explain the training process of EDTMSR through federated learning.

A. Model Architecture

Given the efficiency of residual blocks proposed in [14], we design our architecture with a number of stacked residual blocks, each comprising two convolutional layers separated by a Parametric ReLU [59] activation function. Each convolutional layer is equipped with small 3x3 kernels and 64 feature maps, followed by a batch normalization layer. We experimentally evaluated the optimal number of residual blocks for our task. As shown in Fig. 1, the analysis led us to design our core model with $N_b = 4$ residual blocks, which enable a more stable training process and a smoother convergence compared to deeper architectures. The merged feature map then undergoes up-sampling to match the desired scale in the final layer. The up-sampling is realized through a sub-pixel convolution layer [60].

For a faster convergence and to achieve a better generalization [14], we first train our model in a scale factor $x2$ scenario and then use these pre-trained weights and parameters to initialize our model when running over traffic matrices with higher scale factors. We experienced how this not only speeds up the learning process but also improves model accuracy.

In Fig. 2, we report the validation loss obtained using scale factor 2 trained from scratch and scale factor x4 using the pre-trained model on scale factor x2. The figure shows a faster convergence on scale factor 4, demonstrating the effectiveness of our initialization strategy. This improved convergence can be attributed to the correlation between scale factors, which we consider as interrelated tasks.

B. Federated Learning for Performance and Partial Traffic Visibility Inference

This section describes the federated learning configuration for our `EDTMSR` model. Our goal is to evaluate the ability of such model to predict the fine-grained measurements from real coarse-grained ones. Training a neural network with a federated approach allows each client (i.e., a single node of the federated network) to perform traffic measurements according to different criteria and with arbitrary sampling rates.

We explore two possible federated training configurations: the first has the main objective of speeding up the training procedure of `EDTMSR` by leveraging multiple agents and distributing tasks. The second configuration aims to deal with partial visibility of each agent in the global network while still maintaining acceptable model accuracy. Both solutions leverage *FedAvg* [12] as a federated learning algorithm to train a more robust and performing model. The rest of this section describes the details of both training settings.

Training Boost Configuration. With such a configuration, we assume the network data is available to all the participants, thus all clients can observe and measure the traffic of all computer network nodes. In the federated learning process, we randomly distribute an equal amount of traffic matrices to the participating clients.

A consequence of such random splitting of traffic matrices is that the distribution of traffic patterns on each client training set may be unbalanced. However, in the evaluation section, we show that such an unbalanced distribution does not hinder the performance of our traffic inference. Although the assumption of full visibility of this configuration of `EDTMSR` may be strong, this setting is particularly effective to reduce the convergence time of the model.

Partial Traffic Visibility. In this second configuration, we remove the assumption that all clients participating in the federated training process can observe and measure the traffic of all nodes of the network. This is because clients belonging to different network partitions or regions may be unable to obtain such information but still be interested in estimating traffic volumes. With this configuration, we aim to assess to what extent the distributed training approach enables visibility gains into the network traffic of other clients, despite the lack of global knowledge of the network. By limiting visibility to only a sub-portion of the entire network, each federated client has to monitor, aggregate, and process a smaller traffic matrix representing only a portion of the entire network. This configuration is particularly relevant in contexts where agents are diverse network operators that own different regions and/or do not wish to share network information with peers. Furthermore, we study how this partial traffic knowledge might overlap by tweaking the percentage of network knowledge shared among the clients. In a full non-overlapping scenario, each client has a unique subset of nodes, representing completely disjoint views of the network. As we increase the overlap percentage, we gradually introduce shared nodes among clients and analyze scenarios where multiple entities have visibility into common parts of the network.

V. Experimental Settings

In this section, we describe the datasets used for our evaluation and detail the approach we employed for their preprocessing. Then, we discuss the experimental environment used to produce and test our proposed EDTMSR architecture in both centralized and federated settings, and the metrics used for its evaluation.

A. Datasets

Deep Learning (DL) techniques require extensive and representative data to build an effective neural network model, characterized by both high performance, in terms of training time, and accuracy in the reconstruction of TMs. In this work, we leverage a dataset collected from GÉANT [16]. This anonymized dataset consists of traffic matrices built using Interior gateway protocol (IGP) routing information, NetFlow data collected overall edge links, and Border Gateway Protocol (BGP) the GÉANT network routing information sampled every 15 minutes for 4 months.
The network in which the traffic matrices were collected is formed by 23 nodes. We also use the production traffic from a Meta datacenter [15] to validate our model. The dataset includes hundreds of thousands of 10-Gbps nodes, where each sample contains packet header information and additional metadata such as locality and packet length. The value (i, j) of a traffic matrix corresponds to the traffic going from node i to node j, expressed in *Kbit*.

B. Low-Resolution Traffic Matrix Generation

To generate the coarse-grained measurements necessary for training our model, we employ the following approach. For the GEANT network, we utilize a K-means clustering algorithm to group hosts based on their network distances. The traffic exchanged between these clusters is then aggregated to populate the elements of the LR matrix. This process effectively creates a coarser representation of the network traffic patterns. For each HR matrix in our dataset, we generate a corresponding LR matrix using this method, forming the paired samples (LR, HR) used in our training process. This approach allows us to simulate scenarios where only aggregated, coarse-grained measurements are available, while still maintaining a relationship with the underlying fine-grained traffic patterns we aim to reconstruct. After applying K-means clustering to group hosts into K clusters, we can define the aggregation function A in Eq. 1 as:

$$TM_{i,j}^{LR} = \sum p \in C_i \sum_{q \in C_j} TM_{p,q}^{HR} \tag{7}$$

Here, C_i and C_j represent the sets of hosts belonging to clusters i and j, respectively.

For the Meta datacenter network dataset, we construct our HR matrices by considering the traffic exchanged between racks, aggregated over a 10-second time window. To generate the corresponding LR matrices, we sum the traffic from racks belonging to the same pod. To maintain a consistent scale factor between HR and LR matrices, we consider only N racks per pod when building the HR matrix. This approach allows us to easily obtain the LR version by aggregating the traffic between the N racks for each pod. With this approach, we create a hierarchical representation of the network traffic, with racks as the fine-grained level and pods as the coarse-grained level. Formally, let $TM^{HR} \in \sum R^{(P \cdot N) \times (P \cdot N)}$ be the HR traffic matrix, where P is the number of pods, and N is the number of racks per pod considered. The element $TM_{i,j}^{HR}$ represents the traffic from rack i to rack j over a 10-second window. The LR matrix $TM^{LR} \in \mathbb{R}^{P \times P}$ is obtained through the aggregation function A defined in Eq. 1, where each element of TM^{LR} is computed as:

$$TM_{k,l}^{LR} = \sum i \in P_k \sum_{j \in P_l} TM_{i,j}^{HR} \tag{8}$$

Here, P_k and P_l represent the sets of rack indices belonging to pods k and l, respectively. With this representation, we capture the aggregation of rack-level traffic to pod-level traffic and maintain a consistent relationship between the HR and LR matrices.

C. Training and Implementation Details

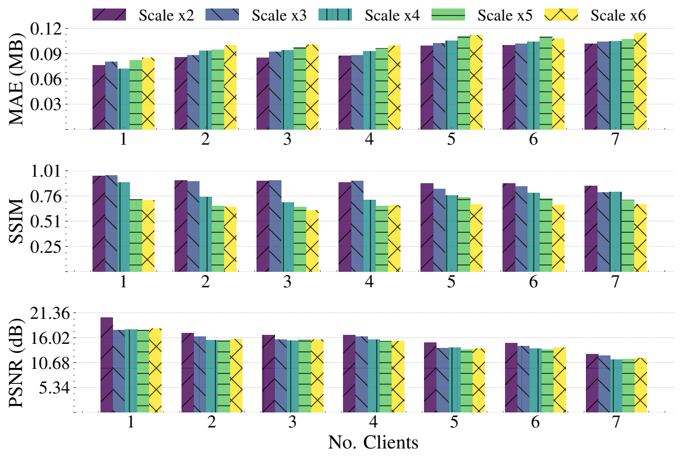
To train our model in a centralized way, we use the Adam optimizer [61] with a starting learning rate of 10^{-3}, $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 10^{-7}$.

One critical aspect of the FedAvg [12] algorithm is that it requires two optimizers, one for the clients and one for the server. The client optimizer is only used to compute local model updates on each client, and for this reason, we used the same Adam optimizer we already described for the centralized training setting. The server optimizer applies the averaged update to the global model at the server. We used Stochastic Gradient Descent (SGD) [62] with a learning rate of 1.0 as the server optimizer. For both centralized and federated training settings, we set the minibatch size at 16, that is 16 traffic matrices are fed to the algorithm for each training round. We trained our models through the L1 loss and perform a maximum of 30 training epochs and 10^3 steps (batches of samples) for each epoch.

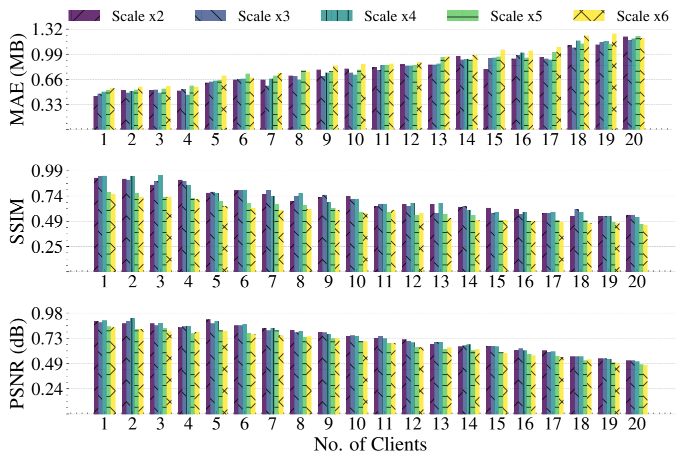
For the federated learning process, we used the *TensorFlow Federated* [63] library. The hardware platform used for both centralized and federated training and testing is equipped with 4 Nvidia Tesla V100 GPUs and Intel Xeon Gold CPU.

D. Evaluation Metrics

To comprehensively assess the quality and fidelity of our generated high-resolution malware traffic matrices, we employ three complementary metrics: Mean Absolute Error (MAE),



(a) GEANT dataset



(b) Meta datacenter traffic dataset

Fig. 3: Performance comparison of the proposed federated learning model on GEANT and Meta datacenter traffic datasets. (a) GEANT dataset results for 2 to 6 clients. (b) Meta datacenter traffic dataset results for 2 to 10 clients. Each subfigure displays Mean Absolute Error (MAE) (top), Structural Similarity Index Measure (SSIM) (middle), and Peak Signal-to-Noise Ratio (PSNR) (bottom) as a function of the number of clients and upscaling factors.

Structural Similarity Index (SSIM), and Peak Signal-to-Noise Ratio (PSNR). MAE provides a straightforward measure of pixel-level differences between generated and original images, offering insight into overall accuracy. SSIM [64] evaluates the perceived quality of generated images by considering the structural similarities. The closer to 1, the more similar are the compared images:

$$SSIM(x, y) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)} \tag{9}$$

Where μ_x, μ_y are the means of input images x and y, σ_x, σ_y are the standard deviations, σ_{xy} is the covariance, and C_1, C_2 are constants to avoid instability.

PSNR quantifies the ratio of maximum signal power to noise, helping us assess how well important details are preserved in the synthetic images.

$$PSNR = 10 \cdot \log_{10}\left(\frac{\text{MAX}^2}{MSE}\right) \tag{10}$$

Where n is the number of observations, MAX is the maximum possible pixel value, and MSE is the mean square error.

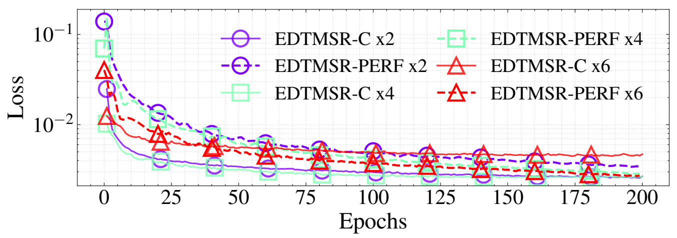


Fig. 4: Comparison of training losses between `EDTMSR-C` (centralized) and `EDTMSR-PERF` (federated) for different upscaling factors.

VI. Evaluation Results

In this section, we report the evaluation of our super-resolution model for the TM inference problem leveraging a federated learning approach. We first detail our model configurations with different federated settings and evaluate their performance (Sec. VI-A). Then, we compare our model in a partial traffic visibility scenario against its centralized version (Sec. VI-A2). We discuss `EDTMSR` performance in a federated scenario using full network knowledge (Sec. VI-A1) and in a partial network visibility scenario with different percentages of shared knowledge between federated clients (Sec. VI-A3). Further, we evaluate the resiliency of our distributed learning approach to different percentages of federated clients drop in Sec. VI-A4. Finally, we compare our model against multiple baselines (Sec. VI-B).

A. Evaluating TM Inference with Federated Learning

In this work, we define two configurations called Federated Learning (FL) for Performance (`EDTMSR-PERF`) and Federated Learning (FL) for Partial Traffic Visibility (`EDTMSR-PTV`). Each configuration is characterized by different settings of the federated training process, determining the characteristics of the distributed environment in which our EDTMSR model is trained, with different data distribution strategies. For `EDTMSR-PERF`, we set and analyze the impact of the number of clients participating in the distributed training process. However, in this configuration, we assume that each client has full knowledge of the network, and the availability of full-size fine-grained matrices for ground truth. Contrarily, for `EDTMSR-PTV` configuration, we leverage the preprocessing in Sec. V-B to determine the distribution of the traffic matrices across the clients. Indeed, for this configuration, we assume that clients only have partial knowledge of the network and, consequently, only a portion of the original traffic matrix. We analyze the performance of EDTMSR-PERF and EDTMSR-PTV configurations across all the considered evaluation metrics defined in Sec. V-D.

1) Evaluation of EDTMSR-PERF configuration: Figure 3 show the performance of our proposed federated learning model on two datasets: GEANT and Meta datacenter traffic. We report the MAE, SSIM and PSNR - as functions of the number of clients and upscaling factors. For the GEANT dataset, we run experiments for 2 to 6 clients, while we extend the evaluation for Meta traffic to 20 clients. This reason of a different client range stems from the datasets' characteristics. The GEANT network's limited host count restricted the maximum scale factor to x6, resulting in a minimum $3x3$ coarse-grained matrix. We omitted scales producing matrices smaller than $3x3$ due to performance limitations. Conversely, the Meta dataset's larger volume allowed evaluation up to 20 clients. For GEANT, as we noted that performance tend to stabilize around 5 or 6 clients, we only report results up to 7 clients. The Meta dataset presents a larger amount of data and number of hosts, facilitating the extension of the analysis up to 20 clients, and providing insights into performance trends over a wider range of federated participants.

As expected, we observe that increasing the number of clients generally leads to slightly worse performance, particularly for lower upscaling factors. This trend is more pronounced in the Meta dataset, where performance degradation is noticeable as the number of clients increases. Moreover, increasing the upscaling factor also results in worse performance, as each client has fewer samples to work with. Interestingly, for larger scale factors ($\geq \times 4$), the impact of the number of clients is almost negligible. In these cases, the federated approach's performance tends to converge closer to that of the centralized approach. These findings suggest that distributing data samples among more clients can be beneficial for predicting traffic matrices, particularly for coarser-grained measurements, due to the diversity of data in the distributed learning process and the reduced overfitting.

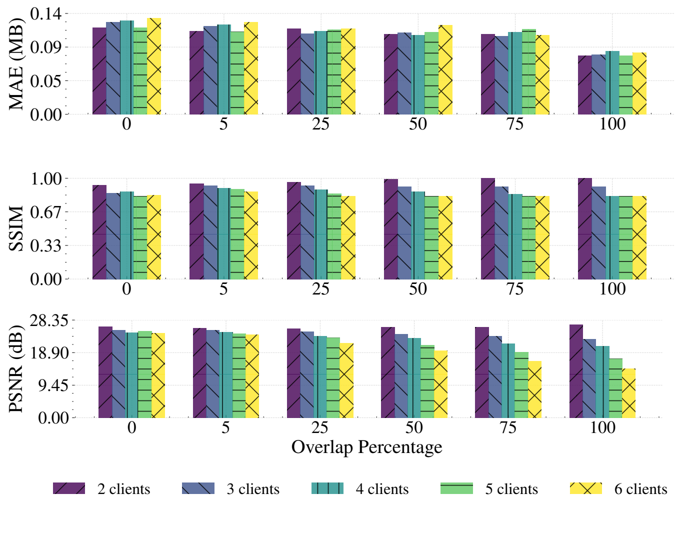
2) EDTMSR-PTV vs. Centralized Configuration: In Figure 4, we first compare the training loss on a validation set between EDTMSR-C, namely the centralized version of our model (i.e. a single client with full network knowledge), and EDTMSR-PTV configuration over 200 training rounds. We consider EDTMSR-PTV with 4 clients, each possessing a unique portion of the full matrix (i.e. no shared knowledge of the network). The comparison reveals that EDTMSR-C version slightly outperforms EDTMSR-PTV configuration. However, it is noteworthy that the EDTMSR-PTV configuration achieves excellent performance in terms of Mean Absolute Error (MAE) and convergence, despite the lack of complete network visibility. Interestingly, we observe that for smaller scale factors (e.g., x2), the performance gap between the centralized and federated versions is more pronounced. However, as we move to higher scale factors (x6), the federated configuration outperforms the centralized approach with faster convergence. This crossover demonstrates the benefits of predicting fine-grained traffic from distributed, highly coarse measurements, particularly when dealing with larger-scale factors.

3) Evaluation of EDTMSR-PTV: In this experiment, we show the performance of our proposed model trained across multiple clients in partial network visibility scenarios. In Figure 5, we analyze the impact of shared network knowledge overlap across clients, varying the number of clients from 2 to 6 for GEANT and 2 to 10 for Meta. We report MAE, SSIM and PSNR metrics as functions of the percentage of shared knowledge in the network. We explore scenarios ranging from 0% overlap, i.e. each client has a unique subset of the network and is trained on a sub-portion of the traffic matrix that only includes the corresponding subset, to 100% overlap, i.e. an equivalent configuration to EDTMSR-PERF, where clients are given the full matrix as ground truth for training.

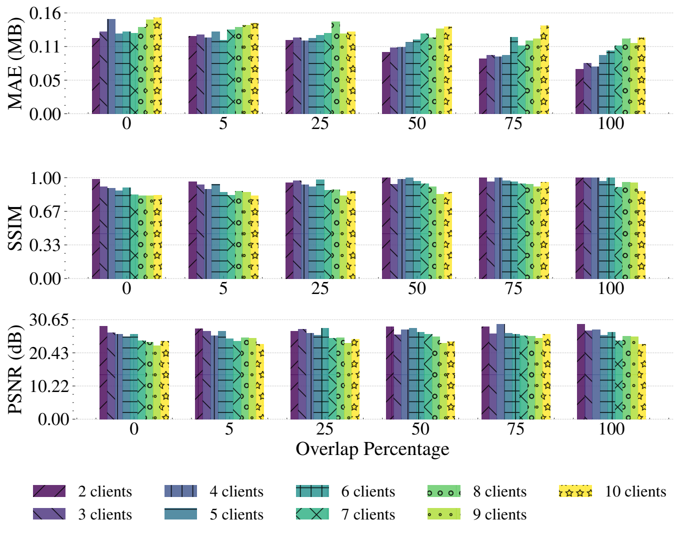
To generate the overlapping behavior, we set the percentage of total nodes that are shared across clients. Every client receives the same subset of selected shared nodes, along with a unique subset of non-shared ones. The coarse view of the network is generated following the methods described in Sec. V-B. For the GEANT dataset, we employ k-means clustering based on node locations to create the required number of nodes for the coarse-grained network view. For the Meta datacenter, we leverage the pod-rack interrelationship, aggregating all rack traffic belonging to the same pod to generate the coarse-grained (low-resolution) traffic matrix. To obtain even coarser matrices, we further aggregate pod traffic based on the nodes belonging to specific portions of the matrix at the previous aggregation level. Nevertheless, increasing the number of clients in the federated settings leads to a smaller matrix size, as unique subsets of network nodes have to be assigned to a number of clients.

For the GEANT network (Figure 5a), we observe that varying the number of clients does not significantly impact performance. However, as expected, increasing the overlap percentage generally improves overall performance across all metrics. Notably, even with 0% overlap, the model achieves performance close to the fully overlapped case. In contrast, the Meta dataset (Figure 5b) shows a more evident performance degradation as the number of clients increases up to 10, especially at lower overlap percentages. This result can be attributed to the excessive sparsity of the large-size Meta traffic matrix. With more clients, the likelihood that some clients will receive inactive parts of the network (i.e., areas with low traffic activity) is higher, leading to less informative training data for those clients. However, it is worth noting that the model achieves competitive performance even when distributing non-overlapping sub-networks to a number of clients. This finding demonstrates the feasibility of applying our super-resolution model to a federated learning scenario without or only partially sharing knowledge of network information between clients while preserving privacy and data isolation.

4) Analysis of Client Participation Variability: In practical scenarios, network nodes or clients may become unavailable due to various reasons such as network failures, maintenance, or resource constraints. By evaluating our model's performance under different client dropout conditions, we investigate its resilience and applicability in dynamic, distributed environments. Moreover, this analysis suggests how our federated learning approach performs when entire portions



(a) GEANT dataset



(b) Meta datacenter traffic dataset

Fig. 5: Performance of our proposed model in federated settings for (a) GEANT dataset and (b) Meta datacenter traffic dataset. For each dataset, the plots show Mean Absolute Error (MAE) (top), Structural Similarity Index Measure (SSIM) (middle), and Peak Signal-to-Noise Ratio (PSNR) (bottom) as a function of the percentage of shared knowledge in the network. Results are presented for varying numbers of clients - 2 to 6 in (a), 2 to 10 in (b) - in the federated learning setup.

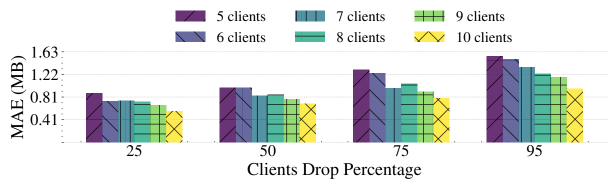


Fig. 6: Mean Absolute Error (MAE) vs Client Drop Percentage for Meta dataset varying the number of clients in the federated learning. Lower MAE indicates better performance. The trend shows that higher drop percentages and fewer total clients both lead to increased MAE, indicating degraded performance in these scenarios.

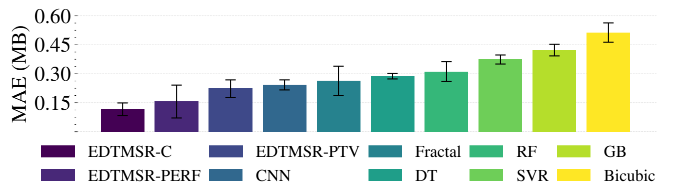


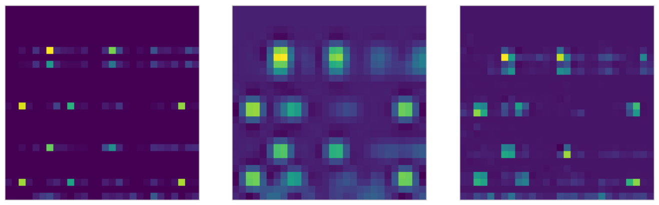
Fig. 7: Comparison over Meta datacenter traffic between variants of our proposed model - i.e., centralized (`EDTMSR-C`), federated with partial traffic visibility and no overlap (`EDTMSR-PTV`), and federated with 100% overlap (`EDTMSR-PERF`) - and Bicubic [34], Decision Trees (DT) [65], Random Forest (RF) [66], Support Vector Regressor (SVR) [67], Gradient Boosting (GB) [68], Convolutional Neural Networks (CNN) [69] and Fractal model [70]

.

of the network are removed. In Figure 6, we report the MAE for Meta traffic dataset when varying the number of clients and the clients drop percentage (i.e., the percentage of the total number of clients that are removed from the federated learning process). For this experiment, we consider the `EDTMSR-PTV` configuration with 0% overlap. Our results show that our model that the error significantly increases when dropping almost 95% of the participating clients, as expected. However, while this is more evident for a small number of clients, contrarily, the model achieves low inference error when more clients are considered. For example, with 10 clients and inference error shows less variability when varying the drop percentage. This reveals that: *(i)* as we distribute non-overlapping portions of the network to different clients, many clients receive low-traffic activity portions that do not contribute significantly to the global model update. Thus, their absence has a minimal impact on overall performance; *(ii)* smaller unique sub-matrices are distributed among clients, making the overall process more robust to high drop percentages. This granular distribution ensures that the loss of any single client does not result in a substantial loss of critical information, maintaining model stability even at high dropout rates.

B. *Comparison with Baselines over Datacenter Traffic*

To further validate the performance of EDTMSR, we compare our solution against different baselines in terms of mean absolute error (MAE), using the Meta datacenter production traffic [15]. Among our baselines, we include bicubic interpolation as the simplest matrix reconstruction technique, some of the state-of-art ML-based predictors such as Decision Trees [65], Random Forest [66], Support Vector Regressor [67], and Gradient Boosting [68] and, as a DL counterpart, we adapted ImageNet [69] Convolutional Neural Network



Original (a) Bicubic (b) EDTMSR-PTV (c)

Fig. 8: Visual comparison of Meta (Facebook) datacenter rack traffic matrix. (a) Traffic matrix ground truth; (b) Predicted traffic using Bicubic interpolation; (c) Predicted traffic using EDTMSR-PTV (**our model**).

to the traffic matrix prediction problem. Furthermore, we include a comparison with a fine-grained traffic estimation approach [70]. This method infers end-to-end traffic matrices, relying on temporal dependencies of sampled traffic traces and employing a combination of techniques such as fractal interpolation, cubic spline interpolation, and the weighted geometric average algorithm. For simplicity, we refer to this method as Fractal.

We consider `EDTMSR-PTV` in its fully distributed version, where each client has a unique subset of the network nodes. We use a four-hour-long pod traffic collection as our training set, aggregated by 10-second intervals. We subsequently test the models on 10-minute rack traffic collected within a different time from the training traffic data. We repeat the experiments five times on the test set and average the results. Figure 7 shows that our model, in its centralized and distributed variants, outperforms all the baselines, including the CNN, which performs the best among the selected models. `EDTMSR-C`, the centralized version of our model, improves the performance of the best baseline by 45% on the Meta datacenter production traffic. Our distributed model achieves comparable performance to the centralized version for both variant `EDTMSR-PTV` and `EDTMSR-PERF`, while the Bicubic interpolation shows a higher prediction error due to the significant data loss obtained during the downsampling operation. Figure 8 shows a visual comparison between the traffic matrices inferred by the bicubic interpolation and our distributed model (`EDTMSR-PTV`). The bicubic interpolation is performed on the LR version of the rack traffic (i.e., by downsampling the test traffic matrix), while EDTMSR is pre-trained on pod traffic aggregated within a different time interval. Despite the sparsity nature of the traffic matrix, our model can identify the peaks in the traffic volumes with high accuracy.

VII. Conclusion

In this work, we proposed Enhanced Deep Traffic Matrix Super-Resolution (`EDTMSR`), a super-resolution model for fine-grained traffic matrix estimation. Through the use of a super-resolution model to infer HR traffic matrices from their corresponding LR version, this work provides a novel inference technique that reduces the cost and complexity of collecting, transferring, and analyzing network measurements.

We integrate our model with a distributed learning procedure to deal with partial network visibility. Our experiments with real-world traffic datasets demonstrate that the proposed models precisely infer fine-grained traffic volumes. We find that our proposed method achieves a low prediction error outperforming existing network traffic matrix interpolation techniques and state-of-art ML/DL predictors.

Acknowledgement

We are grateful to Roberto Amoroso for his initial work on this project conducted at Saint Louis University. This work has been partially supported by NSF award # 2201536, and by Comcast.

References

[1] R. Amoroso, L. Pappone, and F. Esposito, "A federated learning approach to traffic matrix estimation using super-resolution techniques," in *IEEE 20th Consumer Communications & Networking Conference (CCNC)*. IEEE, 2023, pp. 473–476.

[2] H. Feng and Y. Shu, "Study on network traffic prediction techniques," in *Proceedings - 2005 International Conference on Wireless Communications, Networking and Mobile Computing, WCNM 2005*, vol. 2, 2005, pp. 995–998.

[3] Z. Chen, J. Wen, and Y. Geng, "Predicting future traffic using Hidden Markov Models," in *Proceedings - International Conference on Network Protocols, ICNP*, vol. 2016-December. IEEE Computer Society, 12 2016.

[4] A. Sacco, F. Esposito, and G. Marchetto, "Restoring application traffic of latency-sensitive networked systems using adversarial autoencoders," *IEEE Transactions on Network and Service Management*, vol. 19, no. 3, pp. 2521–2535, 2022.

[5] P. Tune, M. Roughan, H. Haddadi, and O. Bonaventure, "Internet traffic matrices: A primer," *Recent Advances in Networking*, vol. 1, pp. 1–56, 2013.

[6] A. Sacco, M. Flocco, F. Esposito, and G. Marchetto, "Partially oblivious congestion control for the internet via reinforcement learning," *IEEE Transactions on Network and Service Management*, vol. 20, no. 2, pp. 1644–1659, 2022.

[7] N. Duffield *et al.*, "Sampling for passive internet measurement: A review," *Statistical Science*, vol. 19, no. 3, pp. 472–498, 2004.

[8] E. Nygren, R. K. Sitaraman, and J. Sun, "The akamai network: a platform for high-performance internet applications," *ACM SIGOPS Operating Systems Review*, vol. 44, no. 3, pp. 2–19, 2010.

[9] K. Claffy, D. Clark, J. Heidemann, F. Bustamante, M. Jonker, A. Schulman, and E. Zegura, "Workshop on overcoming measurement barriers to internet research (wombir 2021) final report," *ACM SIGCOMM Computer Communication Review*, vol. 51, no. 3, p. 33–40, jul 2021.

[10] M. Yu, L. Jose, and R. Miao, "Software defined traffic measurement with opensketch," in *10th USENIX Symposium on Networked Systems Design and Implementation (NSDI 13)*. Lombard, IL: USENIX Association, Apr. 2013, pp. 29–42.

[11] A. Sacco, F. Esposito, and G. Marchetto, "Completing and predicting internet traffic matrices using adversarial autoencoders and hidden markov models," *IEEE Transactions on Network and Service Management*, vol. 20, no. 3, pp. 2244–2258, 2023.

[12] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Artificial Intelligence and Statistics*. PMLR, 2017, pp. 1273–1282.

[13] O. Aouedi, A. Sacco, L. U. Khan, D. C. Nguyen, and M. Guizani, "Federated learning for human activity recognition: Overview, advances, and challenges," *IEEE Open Journal of the Communications Society*, vol. 5, pp. 7341–7367, 2024.

[14] B. Lim, S. Son, H. Kim, S. Nah, and K. M. Lee, "Enhanced deep residual networks for single image super-resolution," in *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. IEEE, 2017, pp. 1132–1140.

[15] A. Roy, H. Zeng, J. Bagga, G. Porter, and A. C. Snoeren, "Inside the social network's (datacenter) network," in *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*, 2015, pp. 123–137.

[16] S. Uhlig, B. Quoitin, J. Lepropre, and S. Balon, “Providing public intradomain traffic matrices to the research community,” *Computer Communication Review*, vol. 36, no. 1, pp. 83–86, 2006.

[17] R. Castro, M. Coates, G. Liang, R. Nowak, and B. Yu, “Network tomography: Recent developments,” *Statistical science*, vol. 19, no. 3, pp. 499–517, 2004.

[18] G. Gürsun and M. Crovella, “On traffic matrix completion in the internet,” in *Proceedings of the Internet Measurement Conference (IMC ’12)*, 2012, pp. 399–412.

[19] A. Medina, N. Taft, K. Salamatian, S. Bhattacharyya, and C. Diot, “Traffic matrix estimation: Existing techniques and new directions,” *ACM SIGCOMM Computer Communication Review*, vol. 32, no. 4, pp. 161–174, 2002.

[20] Y. Zhang, M. Roughan, C. Lund, and D. L. Donoho, “Estimating point-to-point and point-to-multipoint traffic matrices: An information-theoretic approach,” *IEEE/ACM Transactions on networking*, vol. 13, no. 5, pp. 947–960, 2005.

[21] A. Lakhina, K. Papagiannaki, M. Crovella, C. Diot, E. D. Kolaczyk, and N. Taft, “Structural analysis of network traffic flows,” in *Proceedings of the joint international conference on Measurement and modeling of computer systems*, 2004, pp. 61–72.

[22] Y. Zhang, M. Roughan, W. Willinger, and L. Qiu, “Spatio-temporal compressive sensing and internet traffic matrices,” in *Proceedings of the ACM SIGCOMM 2009 conference on Data communication*, 2009, pp. 267–278.

[23] D. Jiang, L. Huo, and Y. Li, “Fine-Granularity Inference and Estimations to Network Traffic for SDN,” *PloS one*, vol. 13, no. 5, p. e0194302, 2018.

[24] R. H. Keshavan, A. Montanari, and S. Oh, “Matrix completion from a few entries,” *IEEE transactions on information theory*, vol. 56, no. 6, pp. 2980–2998, 2010.

[25] J. S. Isaac and R. Kulkarni, “Super resolution techniques for medical image processing,” in *2015 International Conference on Technologies for Sustainable Development (ICTSD)*. IEEE, 2015, pp. 1–6.

[26] Y. Huang, L. Shao, and A. F. Frangi, “Simultaneous super-resolution and cross-modality synthesis of 3d medical images using weakly-supervised joint convolutional sparse coding,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR ’17)*, 2017, pp. 6070–6079.

[27] L. Zhang, H. Zhang, H. Shen, and P. Li, “A super-resolution reconstruction algorithm for surveillance images,” *Signal Processing*, vol. 90, no. 3, pp. 848–859, 2010.

[28] P. Rasti, T. Uiboupin, S. Escalera, and G. Anbarjafari, “Convolutional neural network super resolution for face recognition in surveillance monitoring,” in *International conference on articulated motion and deformable objects*. Springer, 2016, pp. 175–184.

[29] M. Haris, G. Shakhnarovich, and N. Ukita, “Task-driven super resolution: Object detection in low-resolution images,” *arXiv preprint arXiv:1803.11316*, 2018.

[30] Y. Bai, Y. Zhang, M. Ding, and B. Ghanem, “Sod-mtgan: Small object detection via multi-task generative adversarial network,” in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 206–221.

[31] Z. Wang, J. Chen, and S. C. Hoi, “Deep learning for image super-resolution: A survey,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.

[32] J. Allebach and P. W. Wong, “Edge-directed interpolation,” in *Proceedings of 3rd IEEE International Conference on Image Processing*, vol. 3. IEEE, 1996, pp. 707–710.

[33] L. Zhang and X. Wu, “An edge-guided image interpolation algorithm via directional filtering and data fusion,” *IEEE transactions on Image Processing*, vol. 15, no. 8, pp. 2226–2238, 2006.

[34] R. Keys, “Cubic convolution interpolation for digital image processing,” *IEEE transactions on acoustics, speech, and signal processing*, vol. 29, no. 6, pp. 1153–1160, 1981.

[35] Y.-W. Tai, S. Liu, M. S. Brown, and S. Lin, “Super resolution using edge prior and single image detail synthesis,” in *2010 IEEE computer society conference on computer vision and pattern recognition*. IEEE, 2010, pp. 2400–2407.

[36] K. I. Kim and Y. Kwon, “Single-image super-resolution using sparse regression and natural image prior,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 32, no. 6, pp. 1127–1133, 2010.

[37] Z. Xiong, X. Sun, and F. Wu, “Robust web image/video super-resolution,” *IEEE transactions on image processing*, vol. 19, no. 8, pp. 2017–2028, 2010.

[38] M. Bevilacqua, A. Roumy, C. Guillemot, and M. L. Alberi-Morel, “Low-complexity single-image super-resolution based on nonnegative neighbor embedding,” 2012.

[39] H. Chang, D.-Y. Yeung, and Y. Xiong, “Super-resolution through neighbor embedding,” in *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, vol. 1. IEEE, 2004, pp. I–I.

[40] X. Gao, K. Zhang, D. Tao, and X. Li, “Image super-resolution with sparse neighbor embedding,” *IEEE Transactions on Image Processing*, vol. 21, no. 7, pp. 3194–3205, 2012.

[41] S. T. Roweis and L. K. Saul, “Nonlinear dimensionality reduction by locally linear embedding,” *science*, vol. 290, no. 5500, pp. 2323–2326, 2000.

[42] R. Timofte, V. De Smet, and L. Van Gool, “A+: Adjusted anchored neighborhood regression for fast super-resolution,” in *Asian conference on computer vision*. Springer, 2014, pp. 111–126.

[43] J. Yang, J. Wright, T. S. Huang, and Y. Ma, “Image super-resolution via sparse representation,” *IEEE transactions on image processing*, vol. 19, no. 11, pp. 2861–2873, 2010.

[44] J. Yang, Z. Wang, Z. Lin, S. Cohen, and T. Huang, “Coupled dictionary training for image super-resolution,” *IEEE transactions on image processing*, vol. 21, no. 8, pp. 3467–3478, 2012.

[45] R. Zeyde, M. Elad, and M. Protter, “On single image scale-up using sparse-representations,” in *International conference on curves and surfaces*. Springer, 2010, pp. 711–730.

[46] J. Yang, J. Wright, T. Huang, and Y. Ma, “Image super-resolution as sparse representation of raw image patches,” in *2008 IEEE conference on computer vision and pattern recognition*. IEEE, 2008, pp. 1–8.

[47] C.-Y. Yang and M.-H. Yang, “Fast direct super-resolution by simple functions,” in *Proceedings of the IEEE international conference on computer vision*, 2013, pp. 561–568.

[48] G. Freedman and R. Fattal, “Image and video upscaling from local self-examples,” *ACM Transactions on Graphics (TOG)*, vol. 30, no. 2, pp. 1–11, 2011.

[49] D. Glasner, S. Bagon, and M. Irani, “Super-resolution from a single image,” in *2009 IEEE 12th international conference on computer vision*. IEEE, 2009, pp. 349–356.

[50] J.-B. Huang, A. Singh, and N. Ahuja, “Single image super-resolution from transformed self-exemplars,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 5197–5206.

[51] C. Dong, C. C. Loy, K. He, and X. Tang, “Learning a deep convolutional network for image super-resolution,” in *European conference on computer vision*. Springer, 2014, pp. 184–199.

[52] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, “Generative adversarial nets,” in *Advances in neural information processing systems*, 2014, pp. 2672–2680.

[53] C. Ledig, L. Theis, F. Huszár, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang *et al.*, “Photo-realistic single image super-resolution using a generative adversarial network,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4681–4690.

[54] C. Zhang, X. Ouyang, and P. Patras, “ZipNet-GAN: Inferring fine-grained mobile traffic patterns via a generative adversarial neural network,” in *CoNEXT 2017 - Proceedings of the 2017 13th International Conference on emerging Networking EXperiments and Technologies*, vol. 17, 2017, pp. 363–375.

[55] A. C. Yao, “Protocols for secure computations,” in *23rd annual symposium on foundations of computer science (sfcs 1982)*. IEEE, 1982, pp. 160–164.

[56] H. Yu, J. Vaidya, and X. Jiang, “Privacy-preserving svm classification on vertically partitioned data,” in *Pacific-asia conference on knowledge discovery and data mining*. Springer, 2006, pp. 647–656.

[57] P. Vepakomma, A. Singh, O. Gupta, and R. Raskar, “Nopeek: Information leakage reduction to share activations in distributed deep learning,” in *2020 International Conference on Data Mining Workshops (ICDMW)*, 2020, pp. 933–942.

[58] C. Park, D. Hong, and C. Seo, “An attack-based evaluation method for differentially private learning against model inversion attack,” *IEEE Access*, vol. 7, pp. 124 988–124 999, 2019.

[59] K. He, X. Zhang, S. Ren, and J. Sun, “Delving deep into rectifiers: Surpassing human-level performance on imagenet classification,” in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1026–1034.

[60] W. Shi, J. Caballero, and F. Husz&, “x00e1; r, johannes totz, andrew p. aitken, rob bishop, daniel rueckert, and zehan wang. real-time single

image and video super-resolution using an efficient sub-pixel convolutional neural network," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 1874–1883.

[61] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2015.

[62] L. Bottou, F. E. Curtis, and J. Nocedal, "Optimization methods for large-scale machine learning," *Siam Review*, vol. 60, no. 2, pp. 223–311, 2018.

[63] T. T. Authors, "TensorFlow Federated," https://www.tensorflow.org/federated, 2020.

[64] Z. Wang and A. C. Bovik, "Mean squared error: Love it or leave it? a new look at signal fidelity measures," *IEEE signal processing magazine*, vol. 26, no. 1, pp. 98–117, 2009.

[65] J. R. Quinlan, "Induction of decision trees," *Machine learning*, vol. 1, pp. 81–106, 1986.

[66] L. Breiman, "Random forests," *Machine learning*, vol. 45, pp. 5–32, 2001.

[67] H. Drucker, C. J. Burges, L. Kaufman, A. Smola, and V. Vapnik, "Support vector regression machines," *Advances in neural information processing systems*, vol. 9, 1996.

[68] J. H. Friedman, "Greedy function approximation: a gradient boosting machine," *Annals of statistics*, pp. 1189–1232, 2001.

[69] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 2009, pp. 248–255.

[70] D. Jiang, Z. Zhao, Z. Xu, C. Yao, and H. Xu, "How to reconstruct end-to-end traffic based on time-frequency analysis and artificial neural network," *AEU-International Journal of Electronics and Communications*, vol. 68, no. 10, pp. 915–925, 2014.

Lorenzo Pappone is a Ph.D. candidate in Computer Science at Saint Louis University. He received the M.Sc. degree (summa cum laude) in Computer Engineering from University of Naples Federico II in 2021. His main research interests include applied machine learning and networked systems.

Alessio Sacco is an Assistant Professor at Politecnico di Torino. He received the M.Sc. degree (summa cum laude) and the Ph.D. degree (summa cum laude) in computer engineering from the Politecnico di Torino, Torino, Italy, in 2018 and 2022, respectively. His research interests include architecture and protocols for network management; implementation and design of cloud computing applications; algorithms and protocols for service-based architecture, such as Software Defined Networks (SDN), used in conjunction with Machine Learning algorithms.

Flavio Esposito is an Associate Professor with the Department of Computer Science at Saint Louis University (SLU). He received an M.Sc. degree in Telecommunication Engineering from the University of Florence and a Ph.D. in computer science from Boston University. Flavio's main research interests include network management, network virtualization, and distributed systems.