Multi-Slice Privacy-Aware Traffic Forecasting at RAN Level: A Scalable Federated-Learning Approach

Hnin Pann Phyu, Razvan Stanica, Diala Naboulsi

Multi-Slice Privacy-Aware Traffic Forecasting at RAN Level: A Scalable Federated-Learning Approach

Hnin Pann Phyu, *Student Member, IEEE,* Razvan Stanica, Diala Naboulsi, *Member, IEEE,*

***Abstract*—Next-generation mobile networks are expected to meet the requirements of a wide range of new vertical services. Hence, the network slicing concept has been introduced, in which Mobile Virtual Network Operators (MVNOs) are allowed to provide various types of services over the same physical infrastructure, owned by an Infrastructure Provider (InP). To cope with an ever-changing traffic demand, MVNOs seek to pre-allocate/reconfigure the resources at the base stations in an anticipatory manner, based on traffic demand predictions. Ideally, conducting per-slice traffic forecasting requires information that is likely to disclose MVNO confidential information (i.e., business strategy or private user data). To secure data ownership while conducting traffic forecasting, we propose the Federated Proximal Long Short-Term Memory (FPLSTM) framework, which allows MVNOs to train their local models with their private dataset at each base station; subsequently, an associated InP global model can be updated through the aggregation of the local models. The results obtained by training the models on a real-world dataset indicate that the forecasting performance of our proposed approach is as accurate as state-of-the-art centralized solutions, while improving data privacy. To enable scalability, we further propose the Information-based Clustering FPLSTM (IC-FPLSTM) and Random Clustering FPLSTM (RC-FPLSTM) frameworks, dealing with large-scale cellular networks. These solutions demonstrate computation and communication cost efficiency significantly above the state-of-the-art.**

***Index Terms*—Network Slicing, 5G Networks, Machine Learning, Traffic Forecasting, Federated Learning**

I. Introduction

The mobile communications sector has been witnessing a tremendous increase in emerging vertical services that require different types of quality of service (QoS). These vertical services, in general, fall into three categories: enhanced Mobile Broadband (eMBB), Ultra-Reliable and Low-Latency communication (URLLC), and massive Machine-Type Communication (mMTC) [1]. To cope with the immense development of vertical services, the concept of network slicing is regarded as one of the most prominent research directions in future mobile communication networks [2], which can bring a new revenue stream to mobile operators [3]. In general, a sliced architecture encompasses two main business entities: mobile virtual network operators (MVNOs) and infrastructure providers (InPs) [4]. Specifically, the physical network infrastructure is controlled and owned by InPs, who provide the physical and virtual resources to MVNOs, offering various services with heterogeneous requirements to end-users, through network slicing technology [5]. While all the MVNOs are sharing the same physical resources, slice isolation in terms of processing and data storage is provided by the InP [6], to guarantee performance and ensure data privacy among slices.

Faced with dynamic service demands in a sliced architecture, the main challenge of MVNOs is the ability to offer efficient resource allocation while respecting the diverse QoS requirements [7]. This is especially the case in Radio Access Networks (RANs) [8], where static and dynamic approaches can be adopted. While a static strategy is straightforward to implement, resources assigned at base stations for a specific slice instance may be unused or in surplus during the slice lifetime. Conversely, in a dynamic strategy, the assigned resources are reconfigured based on the traffic demand on each slice [9], enabling a more efficient utilization of resources [10]. However, for this, accurate traffic forecasting information is required at the beginning, and for the duration, of the reconfiguration interval of each slice instance [11]. The gain brought by forecasting mobile traffic, when integrated in so-called anticipatory network mechanisms, has been quantified in [12], where a perfect predictor improves the RAN throughput by 35-40% compared to a non-anticipatory approach.

Three potential multi-slice traffic forecasting options exist [13], [14]: *i)* MVNOs with their own centralized forecasting models, *ii)* MVNOs using their own decentralized forecasting models, and *iii)* MVNOs that share their data with the InP to train a centralized forecasting model. The first approach results in high communication costs for the MVNOs, due to the need to transfer large datasets to the central node. In the second approach, possible correlations among base stations are more likely to remain unexploited, as the local models are trained separately. Such a situation could lead to longer training time and hence incur additional computation cost. In the third approach, sensitive and possibly private MVNO data needs to be transmitted to the InP.

Indeed, since MVNO slices are attached to diverse types of base stations (i.e., macro, micro, pico, and femto) [15], the dataset shared with the centralised InP model may include either private domain data or personal domain data. Private domain data covers all the generated/collected data belonging to a private organization, such as the aggregated traffic data of macro base stations. Even though there is only a small chance of exposing the identity of an individual user through aggregated macro base station traffic, the business strategy and the potential revenue streams of MVNOs could be revealed.

On the other hand, personal domain data covers all individual or household data that can expose private user information (identity, habits, preferences, beliefs). For instance, aggregated

H. Pann Phyu and D.Naboulsi are with the Département de Génie Logiciel et des Technologies de l'information, École de Technologie Supérieure, Université du Québec, Montreal, QC H3C 1K3, Canada (e-mail: hnin.pann-phyu.1@ens.etsmtl.ca, diala.naboulsi@etsmtl.ca).

Razvan Stanica is with Univ Lyon, INSA Lyon, Inria, CITI, Villeurbanne, France (e-mail: razvan.stanica@insa-lyon.fr).

This work was supported by the École de Technologie Supérieure (ÉTS), the National Natural Sciences and Engineering Research Council of Canada (NSERC) through research grant RGPIN-2020-06050 and by the French Government through the France Relance program.

data from small-cell base stations (i.e. pico or femto) belonging to one household falls into the category of personal domain data. Indeed, such sensitive information shall not be accessible to any third-party organizations. Therefore, MVNOs should safeguard their data only within their isolated premises, not only to secure their business strategy, but also to respect the privacy of individual users.

Over the years, classical solutions like ARIMA [12] and Holt-Winters [16] have been widely applied for mobile traffic forecasting, although they are not suitable to extract and predict the complicated spatiotemporal features of mobile traffic in presence of user mobility [17]. Solid forecasting performance in this field has been demonstrated in the recent years by machine learning based forecasting solutions, deep learning based techniques in particular. Two deep learning approaches seem particularly suitable for the traffic forecasting problem. Recurrent neural networks are largely used for time series forecasting [18], while convolutional neural networks are able to exploit the spatial correlations between base stations [13]. However, these solutions are generally integrated in centralized forecasting frameworks in the literature.

Federated learning (FL) is a collaborative decentralized machine learning scheme which can overcome the privacy and communication cost challenges mentioned above. In essence, FL implies training local decentralized models with the help of a global model, located on a centralized controller, without the local models sharing sensitive data among each other or with the global model. With respect to network slicing, FL can enable by that a privacy-preserving forecasting framework where MVNOs can collaboratively train their local forecasting models, at the level of base stations, with low communication cost. In this case, the InP is considered as the most appropriate candidate for being a centralized coordinator, holding the global agent, as it is connected to all the different MVNOs, and it does not need to access any sensitive information. Needless to say, the InP has a strong motivation in improving the forecasting accuracy of the MVNOs, since this would optimize the overall resource demands of the network and allow the InP to minimise its capital expenditure.

The configurations of different types of base stations vary in terms of hardware (i.e., CPU, storage) and link connectivity (i.e., optical fiber, microwave). Such system-level heterogeneity may cause a negative impact on collaborative model training, as some local agents may drop out during this phase if their computation or communication resources are not sufficient. Moreover, data samples produced by the different base stations are temporally dependent upon each other. Therefore, the data used in the studied scenario is non independent and identically distributed (non-IID), not only between slices, but also between base stations, resulting in significant statistical heterogeneity. To address these issues, we rely on the federated proximal (FedProx) framework [19] as global model aggregation strategy, to deal with both system and data heterogeneity issues.

However, when applying FL at the country-scale of an operator network, hundreds of thousands of local agents are simultaneously taking part in the federated training, which creates a significant computational demand. We therefore introduce a clustering-based FL solution, to offset these scalability issues. Each MVNO clusters its associated base stations and a global cluster model is integrated to perform the cluster-wise collaborative learning for each MVNO.

The detailed contributions of this paper are listed below:

- We propose a Federated Proximal Long Short-Term Memory (FPLSTM) framework which can be used by MVNOs for base station-level predictions. Our solution integrates FL with the well known Long Short-Term Memory (LSTM) technique, to enhance the forecasting accuracy of multi-slice network traffic.
- We rely on clustering-based approaches to achieve scalability in terms of network size. To cluster the base stations, different approaches are investigated to determine if the performance can be improved. Specifically, our clustering decision is based on three different factors: *i)* the traffic trend of a slice, *ii)* the geographical coordinates of base stations, those two approaches denoted as Information-based Clustering FPLSTM (IC-FPLSTM), and *iii)* random clustering FPLSTM (RC-FPLSTM).
- To solve the heterogeneity issue in federated networks, our FL global aggregation strategy relies on the FedProx framework, which adds the proximal term to the local sub-problem to limit the effect of local models on the overall global model.
- We conduct an extensive evaluation of our solution on a real-world traffic dataset to evaluate the effectiveness of our proposed approaches compared to the existing state-of-the-art solutions. Our results show that the proposed frameworks (i.e. FPLSTM, IC-FPLSTM and RC-FPLSTM) guarantee a forecasting accuracy comparable to the best centralized solutions in the literature and baseline solutions, while preserving data ownership. In addition, we show that our proposed approaches significantly reduce the communication and computation costs.

The remaining part of this paper is assembled as follows. Section II reviews the related works on traffic forecasting and decentralized solutions in network slicing resource management. Section III introduces the system model and problem statement. In Section IV, we provide the detailed design of our proposed solution, including the dimension scaling of the model and the FPLSTM framework. Next, we articulate our extended clustering approaches in Section V and then discuss the results in Section VI. We summarize our conclusions and offer directions for future work in Section VII.

II. Related Work

A number of studies have investigated the traffic forecasting problems in network slicing, by exploring a wide range of machine learning techniques. The authors of [13] introduce a centralized framework employing the three-dimensional convolutional neural network (3DCNN) method for resource demand forecasting in slices, with the purpose of diminishing the overall resource provisioning costs. Similarly, the authors in [18] establish a modified version of LSTM to study the problem of future demand forecasting for individual slice services. By combining the sequence-to-sequence learning paradigm

TABLE I: SUMMARY OF RELATED WORKS

Ref	Network Slicing	Traffic Forecasting	Per Base Station	Privacy	Scope
[13]	✓	✓	✗	✗	Centralized multi-slice traffic forecasting
[18]	✓	✓	✗	✗	Centralized multi-slice traffic forecasting
[20]	✓	✓	✗	✗	Centralized multi-slice traffic forecasting
[21]	✓	✓	✗	✗	Centralized multi-slice traffic forecasting and resource scheduling
[22]	✓	✓	✗	✗	Centralized multi-slice traffic forecasting and resource allocation
[23]	✓	✓	✗	✓	Decentralized multi-slice traffic forecasting
[8]	✓	✗	✓	✓	Decentralized user association per base station
[14]	✓	✗	✗	✓	Decentralized resource allocation per slice
[24]	✗	✓	✓	✓	Decentralized traffic forecasting per base station
[25]	✗	✓	✓	✓	Decentralized traffic forecasting per base station
[26]	✗	✓	✗	✓	Decentralized traffic forecasting in transportation
Our Work	✓	✓	✓	✓	Decentralized multi-slice traffic forecasting per base station

and a convolutional long short-term memory (S2SConvLSTM) network, high accuracy is achieved for hourly traffic forecasting. For the same objectives, in [20], two common machine learning approaches, deep neural networks (DNN) and LSTM, are explored to enrich an efficient resource reservation strategy in the network slicing paradigm. The authors validate the superiority of their proposed models over a classical Auto-Regressive Integrated Moving Average (ARIMA) approach, based on a real-world traffic footprint.

Traffic forecasting functions have been integrated into resource management frameworks for network slicing. More specifically, the authors in [21] build a collaborative learning framework using LSTM (used for large-timescale hourly traffic forecasting of each slice) and Asynchronous Actor-Critic Agent (A3C) (for small-timescale traffic scheduling in the order of milliseconds) to enable efficient resource utilization while considering performance isolation among slices. Also, in [22], the authors couple LSTM with a heuristic-based solution to maximize the user acceptance rate in their resource management framework, in which LSTM provides the forecast bandwidth requirement of each slice and thus enhances the decision making process of the overall framework. All of the above attempts are centralized forecasting approaches; the data privacy of slice tenants and end-users is not considered. It should also be noted that these studies consider slice-level aggregated traffic rather than traffic at the base station level.

Due to the growing concerns, more and more studies have addressed the privacy issue in multi-stakeholder network slicing, with several proposals of decentralized learning approaches in the resource management model of network slicing. Specifically, the authors in [23] emphasize the efficient forecasting of the service performance of different slices by training the corresponding local models with private datasets at the premises and only sharing the weight matrices of the trained local models to the central node for aggregation. Hence, no private data is exposed. The authors employ an Artificial Neural Network (ANN) driven FL forecasting scheme. In [8], the authors attempt to solve the user association problem at the RAN by using a FL-based two layer model aggregation approach, with a horizontal aggregation for user devices accessing the same type of services, and with a vertical aggregation for different services. Again, in their approach, only the features of the models are transferred for aggregation, instead of local resources, to protect the user privacy.

Similarly, having concerns for the data privacy, the study presented in [14] advocates for an online decentralized learning heuristic framework to find the optimal resource utilization of the overall network, while maximizing the social welfare of all the stakeholders. The proposed solution emphasizes guaranteeing secure business transactions between InP and MVNOs. In all the above-mentioned works based on FL, network heterogeneity issues get little to no attention. It is also worth noting that there is an open question regarding the scalability of the proposed schemes in networks that can involve tens of thousands of base stations.

On top of the conventional FL approach, clustering-based FL has been used in traffic forecasting in the context of wireless and transportation networks to enhance the prediction accuracy [24], [25] or to rectify a scalability issue [26]. More specifically, the authors in [24] first cluster the candidates based on both the average traffic trend and the geographical location of base station using the k-means method. Since an average trend is used instead of the actual data, it is less likely to expose any sensitive information. Collaborative training is then performed in each cluster, followed by intra-cluster and inter-cluster global model aggregation. For a similar purpose, but focusing more on the nature of dynamic networks and data privacy, the work in [25] designs dynamic clustering based on a generative adversarial network by sharing zero raw data and adjusting the clusters based on the network dynamics. To offset the scalability challenge in FL, the authors in [26] couple k-means with FL for traffic forecasting in the transportation system. Specifically, small-scale organizations are clustered based on their location information and then intra-cluster aggregation is performed to update the respective global models. Finally, the optimal global model is selected via the ensemble learning technique, thereby showing a better forecasting performance. Hence, it is reasonable to conclude that clustering-based FL exhibits better performance in terms of forecasting accuracy while addressing scalability issues.

Table I summarizes the contributions in the related works and the positioning of our proposal. We focus on slice traffic forecasting at the level of individual base stations, a critical aspect for MVNOs to enable efficient resource allocation at individual base stations. We propose a clustering-based federated learning approach considering the scalability of the network, that allows MVNOs to collectively and efficiently forecast traffic, benefiting from each others' knowledge while preserving their data privacy.

III. System Model and Problem Statement

In this section, we present our system model and outline the challenges of forecasting mobile traffic per slice and per base station.

A. System Model

We consider a multi-service network that includes an InP and one or more MVNOs sharing the physical and virtual resources of the InP, with $\mathcal{B}$ a set of heterogeneous base stations, managed by the InP. A base station $b \in \mathcal{B}$ could thus cover a macro-, micro-, pico-, or femto-cell. Without loss of generality, we consider that each MVNO offers only one specific type of service, through one slice instance. We assume each slice instance has its own virtual protocol stack [27], which ensures the isolation of the data and control planes.

We use $s \in \mathcal{S}$ to represent a slice instance and $\mathcal{S}$ to represent the set of slice instances. We consider that each MVNO offers its service through a slice instance s on a set of base stations $\mathcal{B}_s \subseteq \mathcal{B}$. The user data traffic on each slice s varies over time. We consider time to be slotted, and we use t to refer to one time interval and $\mathcal{T}$ to refer to a set of time intervals of interest. We denote $v_b^s(t)$ as the generated traffic volume at base station $b \in \mathcal{B}_s$ of slice instance $s \in \mathcal{S}$ at time $t \in \mathcal{T}$.

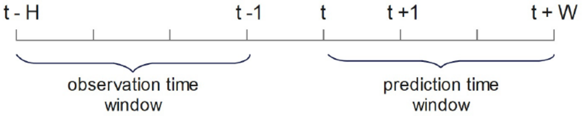


Fig. 1: Observation time window and prediction time window.

Since our focus is on forecasting the mobile traffic volume $v_b^s(t)$, aggregated at the base station level for all the users, we do not model in detail the radio channel and the RAN functions. While the radio channel is an important feature when trying to forecast the individual throughput of each user [12], the problem we tackle is situated at a higher level. In a sliced mobile network architecture, our forecasting framework is part of the management and orchestration (MANO) function, responsible for dynamically initiating, terminating, dimensioning and monitoring a network slice.

TABLE II: DEFINITION OF SYMBOLS

Notation	Description
$b \in \mathcal{B}$	a base station in the set of heterogeneous base stations
$s \in \mathcal{S}$	an instance in the set of slices
$\mathcal{B}_s \subseteq \mathcal{B}$	set of base stations where a slice instance is attached
$\mathcal{T}$	set of time intervals of interest
$v_b^s(t)$	traffic volume at base station b for slice s at time t
X_b^s	historical traffic dataset at base station b for slice s
$X_b^s(t,H)$	traffic trend for observation window H at time t
$X_b^s(t,W)$	forecast traffic demand over the prediction window W
$\mathcal{N}$	set of local agents in the FL framework
$GL(.)$	global loss function
$LL(.)$	local loss function
w_J^c	weight matrix of the neural network in the global model J at communication round c
$w_{J_b^s}^c$	weight matrix of the neural network in the local model J_b^s at communication round c
μ	hyper-parameter to define the effect of the proximal term

B. Problem Statement

Considering our system model, we study the problem of MVNOs traffic forecasting for each base station. We assume an MVNO stores the historical traffic evolution for its slice instance, at every base station in $\mathcal{B}_s$. Here, X_b^s denotes the private dataset of base station b for slice s. Moreover, we define $X_b^s(t,H) = \{v_b^s(t-H), ..., v_b^s(t-1)\}$ as the historical evolution of the traffic of slice s over base station b between time instants $t-H$ and $t-1$, with H representing an observation window. We define as well $\tilde{X}_b^s(t,W) = \{\tilde{v}_b^s(t), ..., \tilde{v}_b^s(t+W)\}$ to represent the forecast traffic over a prediction window W, between time instants t and $t+W$. Figure 1 visualizes the observation and prediction time windows. We summarize the main variables and symbols in our system in Table II.

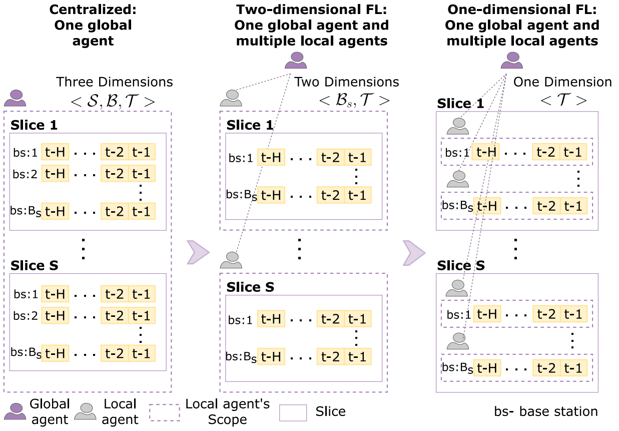
Our objective is to compute a forecast value $\tilde{X}_b^s(t,W)$ as accurate as possible to the real traffic demand. This forecast information can be used by the MVNO when interacting with the MANO function hosted by the InP, to reserve only the amount of resources actually needed by the MVNO during the W time window. In our approach, the prediction is conducted by the MVNO using isolated computation resources, and no raw data is shared with the InP or other MVNOs. Therefore, the InP will not have any direct information regarding the type of service or the end users served by the slices of each MVNO. This results in a higher level of privacy and, as we will show, in reduced communication and computation costs.

IV. Algorithm Design

In this section, we introduce the FPLSTM solution, combining an FL approach, known as the Federated Proximal approach, and LSTM to overcome the issues of privacy and heterogeneity. We begin by discussing the number of dimensions in our problem, followed by the design of the employed federated learning approach.

A. Problem Dimensions

Solving our problem with a centralized approach requires a single global agent operating with three-dimensional objects, represented as $< \mathcal{S}\ ,\mathcal{B}\ ,\mathcal{T}\ >$. In this approach, represented on the left side of Figure 2, the global agent tries to exploit the correlations with respect to these three dimensions, which implies a high system complexity and usually requires a significant training time.



Centralized: One global agent
Two-dimensional FL: One global agent and multiple local agents
One-dimensional FL: One global agent and multiple local agents
Three Dimensions
$< \mathcal{S}, \mathcal{B}, \mathcal{T} >$
Two Dimensions
$< \mathcal{B}_s, \mathcal{T} >$
One Dimension
$< \mathcal{T} >$
Slice 1
Slice S
bs:1 t-H . . . t-2 t-1
bs:2 t-H . . . t-2 t-1
bs:B_S t-H . . . t-2 t-1
Global agent
Local agent
Local agent's Scope
Slice
bs- base station

Fig. 2: Problem dimension comparisons of Centralized and Federated Learning approaches.

Alternatively, the problem can be solved with a two-dimensional FL approach, with multiple local agents, each assigned to one slice instance, and a global agent assuring coordination among them, as exemplified in the middle of Figure 2. This solution eliminates the $\mathcal{S}$ dimension from the vectors and transforms the problem into a two-dimensional problem $<\ \mathcal{B}_s,\ \ \mathcal{T}\ \ >$. Accordingly, all local agents train their local models and coordinate through the global agent, within the collective FL framework. The system complexity is thus reduced, as well as the computation load on each local agent, while collaboration between slices is provided by the global agent. However, each FL agent covers multiple base stations, so data collection needs to be conducted on a centralized server. Similarly, local agents need to operate on the centralized server. As a result, an important communication cost is implied with the two-dimensional FL approach.

Mindful of the communication cost, a one-dimensional FL approach can be considered. Accordingly, the input of the problem is a one-dimensional vector $< \mathcal{T} >$, which practically represents the time series of the traffic demand for a given slice and base station, as shown on the right side of Figure 2. This simplification also reduces the computation load for the local agents, each assigned to a slice instance $s \in \mathcal{S}$ over a single base station $b \in \mathcal{B}_s$. All the local agents then get trained in parallel and coordinate through the global agent. Local agents rely on the classical LSTM method, which is the de facto standard of learning sequential time series data [28].

B. Federated Learning Approach

We employ the one-dimensional FL approach discussed in Section IV-A, so that each local agent trains its own local model with its own local dataset. Meanwhile, the global agent aggregates the weights of the local models in order to update its global model. The overall objective is to minimize the global loss function $GL(.)$ and local loss function $LL(.)$ of the global model and the local model, respectively. Mathematically, this objective can be represented as:

$$min_{w_J}\ GL(w_J) = \sum_{s\in\mathcal{S}} \sum_{b\in\mathcal{B}_s} \rho_b^s\ LL(w_{J_b^s}) \tag{1}$$

where w_J is the weight matrix of the neural network acting as a global model J (LSTM in this case) and $w_{J_b^s}$ is the weight matrix of the local model J_b^s, performing forecasting based on the input dataset X_b^s. Parameter ρ_b^s is the relative impact of each local agent, where $\rho_b^s \geq 0, \forall b \in \mathcal{B}_s$ and $\sum_{s\in\mathcal{S}} \sum_{b\in\mathcal{B}_s} \rho_b^s = 1$. We calculate it as $\rho_b^s = \frac{a_b^s}{a}$, where a_b^s is the total number of samples in dataset X_b^s and a is the overall total number of samples $a = \sum_{s\in\mathcal{S}} \sum_{b\in\mathcal{B}_s} a_b^s$.

C. Proposed Model

The proposed framework consists of two functions: `Global Agent(GA)` and `Local Agent(LA)`. A single GA function runs in the system (at the InP), whereas multiple LA functions are executed, one for each MVNO, at each base station, thereby conducting federated learning for the given communication rounds. Figure 3 illustrates the procedure of an FL-based multi-slice forecasting framework, in which the weight matrix of the different models is made visible with a graph network.

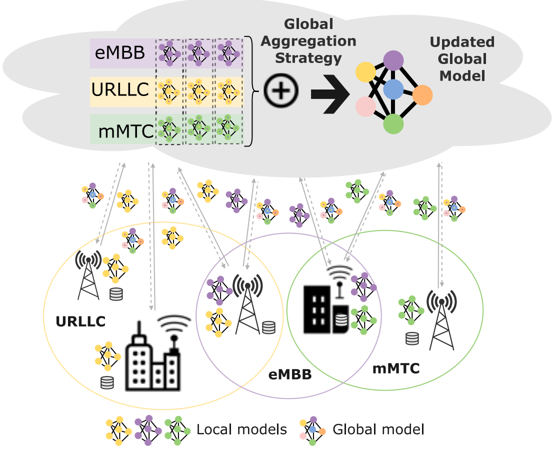


Fig. 3: Federated Learning driven slices traffic demand forecasting architecture.

Our dataset, and sliced network data in general, presents statistical heterogeneity, with non-IID data, as well as system heterogeneity, with base stations showing a diversity in terms of hardware and software. To address this heterogeneity, we use the federated proximal (FedProx) [19] approach as a global aggregation strategy. FedProx integrates a proximal term into the local sub-problems, allowing it to handle training data heterogeneity, unlike other FL methods [29]. The complete model training processes of the FPLSTM framework are formally described in Algorithm 1 `Global Agent(GA)` and Algorithm 2 `Local Agent(LA)`.

The inputs for Algorithm 1 are the number of communication rounds C, the set of local agents $\mathcal{N}$, where each local agent is associated to a slice instance $s \in \mathcal{S}$ over a single base station $b \in \mathcal{B}_s$, the hyper-parameter μ to define the impact of the proximal term on the sub-problem, the initial global weight matrix w^0 with any arbitrary values, and the fraction Ω of local agents to take part in each communication round.

The algorithm starts by initializing the global model weight matrix $w^c_{\mathcal{J}}$ with w^0 for $c = 0$ (Line 1). For each communication round, the GA randomly selects a fraction Ω of local agents, denoted as $\mathcal{N}^c$, from the set $\mathcal{N}$ (Line 3). At the same time, the GA shares the latest global weight matrix $w^c_{\mathcal{J}}$ with all the chosen local agents $\mathcal{N}^c$ (Line 4). We denote $n^s_b \in \mathcal{N}^c$ as the local agent of slice s associated to base station b. Next, each local agent trains its own local model in parallel at each base station to find the local weight for that communication round (Line 5). The detailed steps of the `Local Agent` function are shown in Algorithm 2.

Algorithm 1: `Global Agent(GA)`

```
   Input: $C, \mathcal{N}, \mu, w^0, \Omega$
1  Initially $w^c_{\mathcal{J}} = w^0$ , $c = 0$
2  for each communication round $c = 0, 1, .., C-1$ do
3      GA randomly selects $\Omega$ of local agents $\mathcal{N}^c \subseteq \mathcal{N}$
4      GA sends $w^c_{\mathcal{J}}$ to all chosen local agents $\mathcal{N}^c$
5      /* Chosen Local Agents perform the
          training in parallel at each
          base station (Algorithm 2)       */
6      GA receives $w^{c+1}_{\mathcal{J}^s_b}$ from all local agents $\mathcal{N}^c$
7      GA updates $w^{c+1}_{\mathcal{J}} = \frac{1}{|\mathcal{N}^c|} \sum_{n^s_b \in \mathcal{N}^c} w^{c+1}_{\mathcal{J}^s_b}$
8      GA updates global_model($w^{c+1}_{\mathcal{J}}$) ◁ No
         Dataset is required
9  end for
```

The input for Algorithm 2 includes a dataset X^s_b, the batch size β, the local epochs $\mathcal{E}$, the global model weight $w^c_{\mathcal{J}}$, the learning rate η, the hyper-parameter μ and the global communication round c. First, the LA updates its local model weights matrix $w^c_{\mathcal{J}^s_b}$ with the latest available global model weights matrix $w^c_{\mathcal{J}}$ (Line 1). Afterwards, the LA trains its `local_model(`$w^c_{\mathcal{J}^s_b}$`,` X^s_b`)` according to the defined local epochs E and batch size β (Line 2 - Line 8). We rely on the Stochastic Gradient Descent (SGD) approach to update the local model weight matrix, where $w^*_{\mathcal{J}^s_b}$ is the optimized local weight matrix , and $\bigtriangledown$ is the gradient of loss function $l(w_{\mathcal{J}^s_b}, \beta)$ of batch size β with the learning rate η (Line 6). Next, instead of updating the local model weight matrix $w^{c+1}_{\mathcal{J}^s_b}$ with the weight matrix $w^*_{\mathcal{J}^s_b}$ of minimized local loss function $LL(w^*_{\mathcal{J}^s_b})$, the proximal term is integrated into $LL(w^*_{\mathcal{J}^s_b})$ to limit the effect of local model updates on the overall global model. The local model weights matrix $w^{c+1}_{\mathcal{J}^s_b}$ is then updated with the minimum weights matrix: $w^{c+1}_{\mathcal{J}^s_b} \approx argmin_{w_{\mathcal{J}^s_b}} h(w^*_{\mathcal{J}^s_b}, w^c_{\mathcal{J}})$ (Line 9). Parameter μ in the proximal term is used to control the effect of local updates on the global model (for instance, $\mu = 1$ implies a more significant effect of the local models than $\mu = 0.01$). Finally, the LA sends the local model weight matrix back to the `Global Agent` (Line 10).

Algorithm 2: `Local Agent(LA)`

```
    Input: $X^s_b$, $\beta$, $\mathcal{E}$, $w^c_{\mathcal{J}}$, $\eta$, $\mu$, $c$
1   Initially $w^c_{\mathcal{J}^s_b}$ = $w^c_{\mathcal{J}}$ ◁ Adapt global weight matrix
2   LA trains local_model($w^c_{\mathcal{J}^s_b}$, $X^s_b$) ◁ Feed local
      Datasets
3   Batch ← Split  $X^s_b$ into batch size $\beta$
4   for local epoch $i = 1, ..., \mathcal{E}$ do
5       for each batch $\beta \in$ Batch do
6           $w^*_{\mathcal{J}^s_b} \leftarrow w_{\mathcal{J}^s_b}$ - $\eta \bigtriangledown l(w_{\mathcal{J}^s_b}, \beta)$
7       end for
8   end for
9   Obtain local weight $w^{c+1}_{\mathcal{J}^s_b} \approx argmin_{w_{\mathcal{J}^s_b}} h(w^*_{\mathcal{J}^s_b}, w^c_{\mathcal{J}})$ =
      $LL(w^*_{\mathcal{J}^s_b}) + \underbrace{\frac{\mu}{2} \left\| w^*_{\mathcal{J}^s_b} - w^c_{\mathcal{J}} \right\|^2}_{Proximal\ term}$
10  LA uploads $w^{c+1}_{\mathcal{J}^s_b}$ to GA
```

Next, the GA receives the weight matrices of the updated local models, $w^{c+1}_{\mathcal{J}^s_b}$, and averages them to update the global model weights, $w^{c+1}_{\mathcal{J}}$ (Line 6 - Line 7 in Algorithm 1). Finally, the GA updates the `global_model(`$w^{c+1}_{\mathcal{J}}$`)` with the local models weight matrices $w^{c+1}_{\mathcal{J}}$ for $s \in \mathcal{S}$ and $b \in \mathcal{B}_s$ (Line 8 in Algorithm 1). The algorithm is terminated at the end of all communication rounds or once it converges.

To assess the complexity of our FPLSTM approach, we first observe that the computation complexity of federated learning depends on the number of agents $\mathcal{N}^c$ participating in each round of the federated training as $O(log(\mathcal{N}^c))$ [30]. The computation complexity of each local agent, in order to update the weights matrix by the means of the LSTM model, is $O(W\mathcal{E} + W)$, where W is the total number of parameters in the LSTM model and can be calculated as follow [31]:

$$W = 4IH + 4H^2 + 3H + ZH \qquad (2)$$

where I is the number of LSTM input units, H is the number of hidden units and Z stands for the number of output units. We also empirically evaluate the computing time of our proposed solution (and benchmarks) in Section VI-F.

V. Clustering Solutions

An MVNO generally covers large, country-wide geographical areas, with a number of base stations that can reach hundreds of thousands. Having only one global agent in an FL framework in such a scenario is not scalable, for two reasons: *i)* the high computation load required from the global agent, and *ii)* the high communication delay between local agents and the only global agent in the system. In this section, to offset this scalability issue, we add to the FPLSTM framework a clustering approach, while ensuring forecasting accuracy in line with the best options found in the literature. With this, we first outline the motivation of our clustering approach, and then we provide a detailed description of clustering-based FPLSTM with different decision mechanisms in Algorithm 3 and Algorithm 4.

A. Clustering

Despite the distinct benefits of FPLSTM, our approach presents a practical limitation in terms of scalability, especially with respect to ultra dense network deployment of 5G and beyond networks. We therefore consider designing a clustering-based approach to cope with this scalability issue. This extends the essence of FPLSTM, by clustering local models on an MVNO basis.

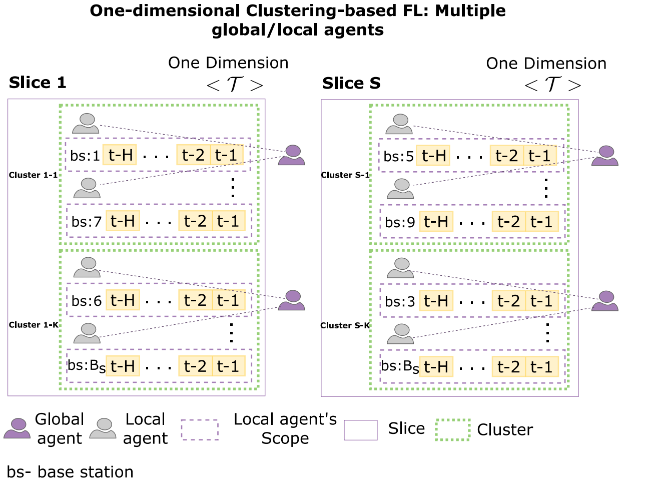


Fig. 4: Problem dimensions of the Clustering-based FPLSTM framework.

Figure 4 and Figure 5 illustrate the framework. Instead of having one global model, the FL process now runs on a per-cluster basis. Therefore, the MVNO first proceeds to clustering the base stations it is attached to. Each cluster is then handled by its own associated global model. As can be seen in Figure 4, once the clustering is done, the forecasting process still operates over one-dimensional vectors $< \mathcal{T} >$. Therefore, a set of base stations belonging to the same cluster conducts the collaborative learning. In fact, this process gives better flexibility to the MVNOs, as their global models could either be run all together in a central node or distributed in nearby edge nodes (further reducing the communication cost of the overall network).

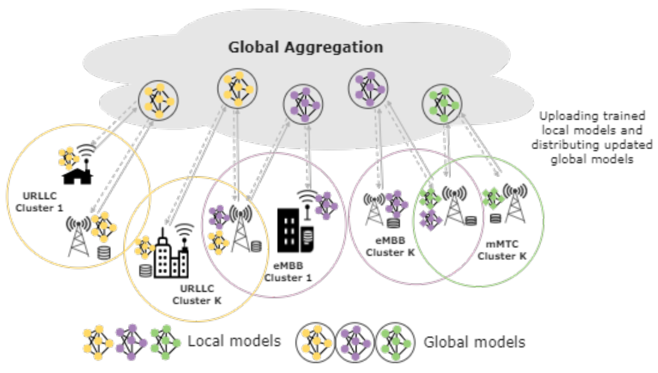


Fig. 5: Clustering-based FPLSTM framework.

We investigate three different clustering schemes: *i)* clustering based on traffic trends, *ii)* clustering based on geographical information, and *iii)* random clustering. Intuitively, the first approach should yield better forecasting accuracy, as more correlated base stations are involved in the corresponding federated training process. The second approach is also expected to lead to good forecasting results, since some traffic correlation is implicit for geographically close base stations. Also, the communication delay between the local agents and the global agent can be minimised in this case, through a careful deployment of the geographical global agents. Conversely, the third approach does not exploit any traffic correlations to improve forecasting accuracy, but still solves the computation scalability problem in an easy way.

Algorithm 3: IC-FPLSTM: `Information-based Clustering FPLSTM`

```
Input  : $\mathcal{Y}$, $Z$, $trend$
         // $\mathcal{Y}$ is a set of base station vectors.
         // $Z$ is the maximum number of iterations.
         // $trend$ is a boolean equal to true if
         IC-FPLSTM targets traffic trend clustering
         and false otherwise.
Output: $\mathcal{G}$
         // $\mathcal{G}$ is a set of clusters, where each cluster is
         itself a set of base stations.
if $trend$ = $TRUE$ then
    $\mathcal{Y} = PCA(\mathcal{Y})$
    // Apply PCA for dimensionality reduction of
    high dimensional base station traffic trend vectors
end if
$K$= Elbow($\mathcal{Y}$) //Determine the target number of
  clusters $K$
$\mathcal{C} = \{c_1 \cdots c_K\} \subseteq \mathcal{Y}$ // Intialize centroids set as a
  random subset of $\mathcal{Y}$
for $z : 1 \cdots Z$ do
    $\mathcal{G} = \{\emptyset, ..., \emptyset\}$ // Initialize $\mathcal{G}$ as a set of $K$ empty
    sets
    for $b : 1 \cdots |\mathcal{Y}|$ do
        $k^* = \underset{k}{argmin} \|c_k - y_b\|^2$
        $\mathcal{G}_{k^*} \leftarrow \mathcal{G}_{k^*} \cup \{y_b\}$
    end for
    for $k : 1 \cdots K$ do
        $c_k \leftarrow \frac{1}{|\mathcal{G}_k|} \sum_{y_b \epsilon \mathcal{G}_k} y_b$ ◁ Update the centroids
    end for
end for
```

B. Extended Models

We extend our prior FPLSTM approach by integrating the clustering approach, in which base stations are clustered by the MVNO based on the three schemes described above (i.e., traffic trends, latitude/longitude and random), followed by the FL process. To this end, we denote them as: *i)* Information-based Clustering FPLSTM (IC-FPLSTM) approach, covering for simplicity both clustering based on traffic trends and clustering based on geographical information of base stations, shown in Algorithm 3, and *ii)* Random Clustering FPLSTM (RC-FPLSTM) approach, as shown in Algorithm 4.

As the name implies, the input of Algorithm 3 includes: i) $\mathcal{Y}$ the information of the base stations in $\mathcal{B}_s$, with respect to slice s of the MVNO, ii) Z the maximum number of iterations for the algorithm, and iii) $trend$ a boolean representing the choice of IC-FPLSTM scheme (i.e. slice traffic-based or geographical location-based). More precisely, $\mathcal{Y}$ is a set of base station vectors, with each vector representing for its base station: i) the slice traffic volume over different time slots or ii) its latitude/longitude coordinates. Z represents the number of iterations for which the algorithm should be trained. $trend$ is equal to true if IC-FPLSTM applies slice traffic volume information for clustering the base stations and false if IC-FPLSTM applies the latitude/longitude coordinates instead. The output of the algorithm, $\mathcal{G}$, is a set of clusters, where each cluster is composed of a set of base stations.

In the algorithm, we first check the value of $trend$. If it is set to true, we proceed to reducing the dimensionality of the slice traffic vectors $\mathcal{Y}$, using the principal component analysis (PCA) technique (Lines 1-3). Our clustering approach is based on the k-means clustering method. Before applying it, we thus determine the adequate number of clusters K by using the elbow method (Line 4). Given the desired number of clusters K, the set $\mathcal{C}$ of cluster centroids (i.e. centers of clusters) is initialized randomly forming a subset of $\mathcal{Y}$ (Line 5). With c_k representing the centroid of cluster $\mathcal{G}_k$, the next step assigns each data point y_b to the cluster with the closest centroid, using the euclidean distance (Lines 8-11). This allows to form clusters of base stations that either present similar slice traffic trends or are geographically close. Next, the centroids are updated by computing the average of the points in each cluster (Lines 12-14). The last two steps are repeated Z times.

In the case of Algorithm 4, the inputs are simply the set of base stations $\mathcal{B}_s$ and the desired number of clusters K. The output is the same as in Algorithm 3. Here, we pick the base stations randomly from the set $\mathcal{B}_s$ and place them in a balanced way into each cluster (Lines 2-8). Finally, we put the surplus base stations from $\mathcal{B}_s$ into clusters selected randomly (Lines 9-14).

Algorithm 4: RC-FPLSTM: `Random Clustering FPLSTM`

```
Input  : $\mathcal{B}_s$, $K$ // $\mathcal{B}_s$ is the set of base stations where
         slice $s$ is deployed, $K$ is the desired number
         of clusters
Output: $\mathcal{G}$ // Set of clusters, where each cluster is
         itself a set of base stations
1  $\mathcal{G} = \{\emptyset, ..., \emptyset\}$ // Initialize $\mathcal{G}$ as a set of $K$ empty sets
2  for $k : 1 \cdots K$ do
3      while $|\mathcal{G}_k| < \frac{|\mathcal{B}_s|}{K}$ do
4          $b = Random(\mathcal{B}_s)$ // Pick up a random base
            station $b$ from $\mathcal{B}_s$
5          $\mathcal{G}_k \leftarrow \mathcal{G}_k \cup \{b\}$
6          $\mathcal{B}_s \leftarrow \mathcal{B}_s \setminus \{b\}$
7      end while
8  end for
9  if $\mathcal{B}_s \neq \emptyset$ then
10     for $b : 1 \cdots |\mathcal{B}_s|$ do
11         $k = Random(K)$ // Pick a random $k \leq K$
12         $\mathcal{G}_k \leftarrow \mathcal{G}_k \cup \{b\}$
13     end for
14 end if
```

VI. Evaluation

In this section we present the evaluation of our proposed methodologies. We begin by introducing the nature of the dataset we used, followed by the benchmarks and performance metrics that we study. We then discuss the results obtained, where the benefits of the FL are elaborated from diverse perspectives (i.e. forecasting accuracy, computation, communication efficiency, sample efficiency, heterogeneity, scalability and global agent performance).

A. Dataset

We evaluate the three proposed approaches (FPLSTM, IC-FPLSTM and RC-FPLSTM) using a real-world dataset collected from the Orange 4G network in Poitiers, France. The locations of the 57 cellular sites in the city are shown in Figure 6. The dataset includes the data traffic demand of different mobile applications at each base station, for a period of 10 days in May 2019, with a granularity of 10 minutes.

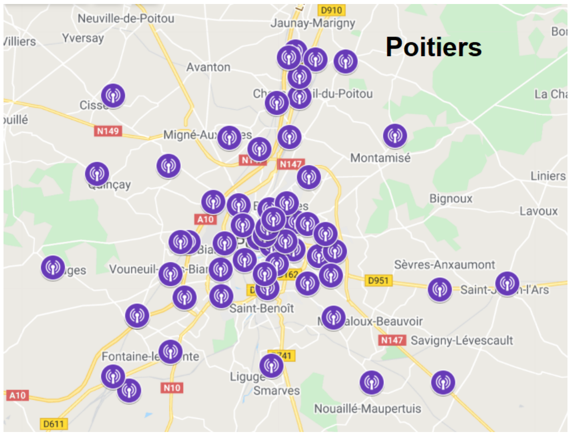


Fig. 6: 4G site locations in Poitiers, France.

Since network slicing has not been deployed yet in real-world commercial networks, we assume slices are deployed on an application-basis, i.e., one application maps to one slice instance. Specifically, Facebook, YouTube, Google, and Instagram have been considered as four different types of slices attached to the base stations. Indeed, these four applications are used for quite different purposes and require different bandwidth and latency [32], which is well aligned with the concept of network slicing.

For the sake of simplicity, we have considered that all four slices are attached to all the 57 base stations. Thus, we end up with 228 local agents in total in our FL approaches. Finally, we note that we use 80% of the data for training of the models, and keep the remaining 20% for testing purposes.

B. Benchmarking

Our ultimate goal is to show the benefits of our proposed schemes compared to state-of-the-art baselines, four of which are described below:

- **Centralized 3DCNN (Deepcog):** Deepcog [13], based on a 3DCNN model, is one of the best known centralized mobile traffic forecasting frameworks. It uses as input a tensor description of network traffic for all the slices in the network. The loss function tries to find a balance between resource overprovisioning and unserviced demands. The whole dataset is shared and aggregated at the centralized server and the model is trained centrally. Our main purpose is to compare the accuracy of Deepcog and FPLSTM, with the former being one of the best centralised solutions in the literature.
- **Decentralized LSTM (DLSTM):** For DLSTM, we train all the local agents in a fully isolated manner, with no collaboration among local agents. We use LSTM as our local models. There is no global model in this case, as it is widely used for time series forecasting. DLSTM guarantees data privacy, as no data leaves the premises. However, new slices could be deployed in the network at any time and DLSTM, as a non-collaborative approach, will need to train new models for those new slices from scratch, a process with high computation costs.
- **Federated Averaging LSTM (FALSTM):** Federated Averaging is an alternative FL solution to our Federated Proximal approach. The training process and problem dimension of FALSTM is the same as that of FPLSTM. The main difference comes from the fact that federated averaging is used as the global aggregation strategy in FALSTM. Simply put, all the weights matrices received from the local models are averaged to update the global model, without using any proximal term. The objective is to verify the expected benefits of the federated proximal approach in managing data heterogeneity.
- **Two Dimensional FL (2DFL):** We employ the two-dimensional FL (as visualized in the middle of Figure 2). We use two-dimensional convolutional neural networks (2DCNN) as our local models and as a global model. Accordingly, the 2DFL model builds on spatio-temporal correlations to derive predictions, while allowing local agents to benefit from each other's knowledge. Since we consider four different slice instances, in this 2DFL approach a total of four local agents and one global agent participate in the training. 2DFL is expected to incur in higher communication costs because data needs to be transferred to a central node to train the local agents (which represent one slice each).

C. Performance Metrics

We rely on the Root-Mean-Square Error (RMSE) to measure the forecasting accuracy performance of the algorithms. All the models are trained with the objective of minimising the RMSE value between predictions and ground truth. In the following, all the average RMSE results refer to the test dataset only.

TABLE III: LIST OF PARAMETERS

Parameter Description	Value
Number of local agents N	228
Global model	LSTM:64 hidden units, flatten and fully-connected layers
Local model	LSTM:64 hidden units, flatten and fully-connected layers
Percentage of training set	80 % of the dataset
Percentage of testing set	20 % of the dataset
Communication round C	20
Loss functions	RMSE
μ	[0.001, 0.01, 0.1, 0.5, 1]
Fraction of local agents Ω	[0.1, 0.3, 0.5, 0.7, 1]
Local epoch E	200
Batch size β	16 samples
Learning rate η	0.001

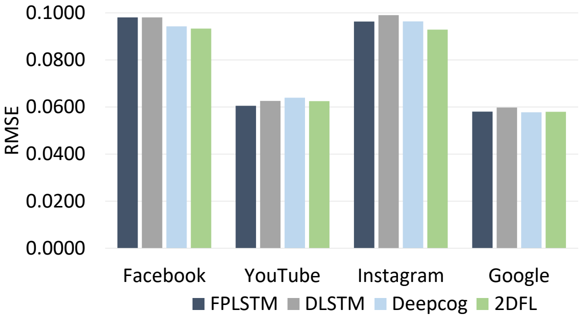


Fig. 7: RMSE values comparisons of FPLSTM, DLSTM, Deepcog and 2DFL.

To further evaluate the effectiveness of our proposed framework, we use two additional metrics: communication cost and computation cost. With this intent, we use the model proposed in [33] for the calculation of the communication cost, expressed as: $2 \cdot C \cdot \mathcal{N} \cdot \Omega \cdot \psi$, where C is the total number of global communication rounds, $\mathcal{N}$ is the total number of local agents, Ω is the local agent selection rate and ψ is the size of the transverse object (i.e., the size of the machine learning model or of the raw dataset). On the one hand, for FPLSTM, ψ is the size of the machine learning model and it can be calculated as $\psi = P_{\psi} \cdot \Upsilon$, where P_{ψ} is the total trainable parameter and Υ is the size, in bits (i.e., 4bits, 8bits, 16bits, 32bits), of the model parameter. For instance, the size of the typical LSTM model with approximately 70 million 32-bits real value parameters is 134.4MB if the input and hidden layer size is 1024 [34]. On the other hand, ψ is the size of the raw datasets for a centralized approach. For the computation cost, we rely on the CPU utilization status of the Compute Canada servers that we used to train our models.

D. Implementations

We implemented our proposed federated approaches using the Flower Federated Framework [35]. All the models (i.e. FPLSTM, IC-FPLSTM, RC-FPLSTM, Deepcog, DLSTM, FALSTM and 2DFL) were executed in a Python environment with open-source TensorFlow libraries. Next, the models were trained on high-performance Linux servers provided by Compute Canada. Since we considered four different applications and 57 base stations, there are a total of 228 local models

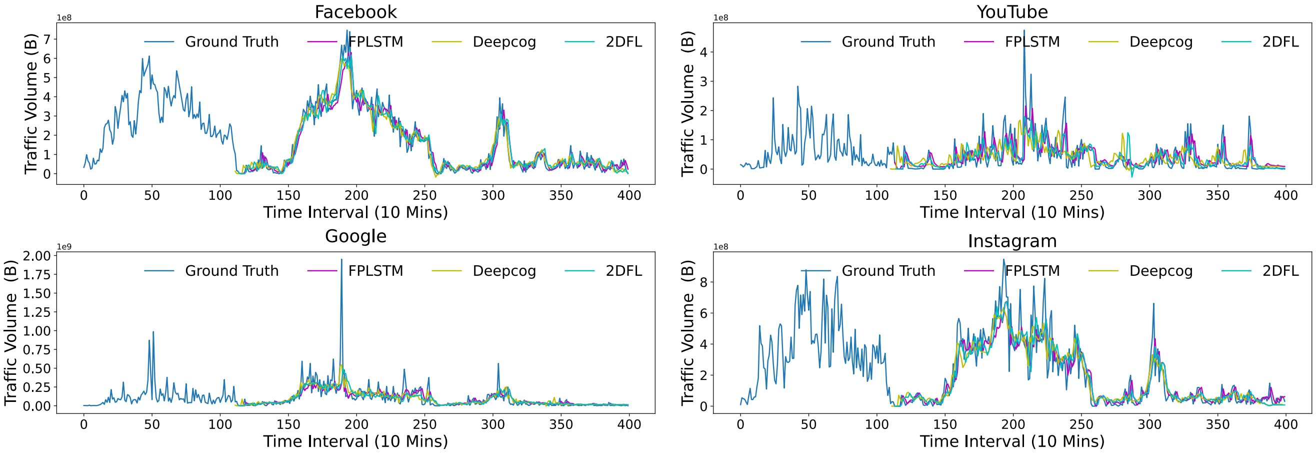


Fig. 8: Forecasting results vs Ground truth for Facebook, YouTube, Google and Instagram for FPLSTM, 2DFL and Deepcog.

participating in the federated training. In addition, there is one global model for the FPLSTM and the 2DFL, and as many global agents as clusters in IC-FPLSTM, RC-FPLSTM, IC-FALSTM. We observed the optimal hyperparameters of our federated models through more than a hundred trial-and-error processes. For brevity, we summarized the hyperparameters of the models associated with our proposed framework in Table III. Finally, we note that, with respect to the clustering approach, we select two components with PCA, as they lead to the best results according to several performance tests.

E. Forecasting Accuracy

We first compare the performance of our FPLSTM solution with its counterparts. Figure 7 depicts the average RMSE values of FPLSTM, Deepcog, DLSTM and 2DFL for the four considered slices. As shown in this figure, the RMSE values of FPLSTM and DLSTM are indeed generally higher than those obtained by the Deepcog and 2DFL solutions, but the difference is rather small and even favorable to the decentralised solutions for the YouTube slice. It is notable that 2DFL exhibits the best RMSE values in the three other applications (i.e. Facebook, Google and Instagram).

Regarding DLSTM, it is fair to say that it shows acceptable results. However, in a real scenario, the number of slices in the network is changing dynamically, based on the service usage and demand [36]. Thus, new slices could be deployed in the network at any time. In such a scenario, the DLSTM, as a non-collaborative approach, would have to train new models for those new slices from scratch. Instead, in federated approaches, new agents can easily obtain an already-trained model from the global agent, removing this cold start problem encountered in DLSTM. Considering this preeminent role of the global model in the context of multi-slice traffic forecasting, we further investigate the performance of the global agent in our FL-based approaches in Section VI-J.

The results of the proposed FPLSTM approach are on pair with a centralised approach such as DeepCog, while not requiring the MVNO to share any potentially sensitive data. To visualize the forecasting performance achieved by FPLSTM, Deepcog and 2DFL, we plot their resulting predicted traffic trend and associated ground truth for different applications in Figure 8. Once again, we can observe that FPLSTM achieves very similar results to Deepcog and 2DFL, following quite well the general trend of the ground truth time series.

Considering the four different slice types, the four applications have a certain regularity in the traffic demand, which is due to the day/night cycle. However, their patterns are quite different. If we consider the two extreme examples, Facebook presents a more complex shape, with a slower variation, while Google traffic presents a simpler shape, but much noisy, with more significant peaks. The two other applications show an intermediate behavior, with Instagram closer to Facebook and YouTube closer to Google. The behavior of RMSE with respect to the application type indicates a higher prediction error for the more complex Facebook shape, and a lower one for the simpler, but noisy, Google traffic. However, the results in Figure 8 indicate that the more frequent and intense peaks in the Google dataset are actually more difficult to predict.

To further appreciate the performance of FPLSTM, we conduct in-depth investigations by varying the observation time window W and the prediction time window H. Table IV and Table V represent the RMSE value comparisons of four applications under different W and H settings for FPLSTM, Deepcog and 2DFL. Since we use 10-minutes granularity, a prediction window $W = 3$ means that the models conduct predictions over the next 30 minutes, and so forth. Similarly, an observation window $H = 2$ means that the models consider the last two 10-minutes intervals to conduct predictions. As it can be seen in Table IV, the performance of our proposed mechanism is even better than that of its counterparts for the larger prediction time windows $W = [5, 10, 15]$. This is really impressive, considering the centralisation used in Deepcog and the supplementary features used by 2DFL.

F. Communication, Computation and Sample Efficiency

Next, we compare the communication cost of FPLSTM and that of the centralized Deepcog and 2DFL approaches. Moreover, we vary the fraction of the dataset used by the training process. We consider the 10-days dataset as a full

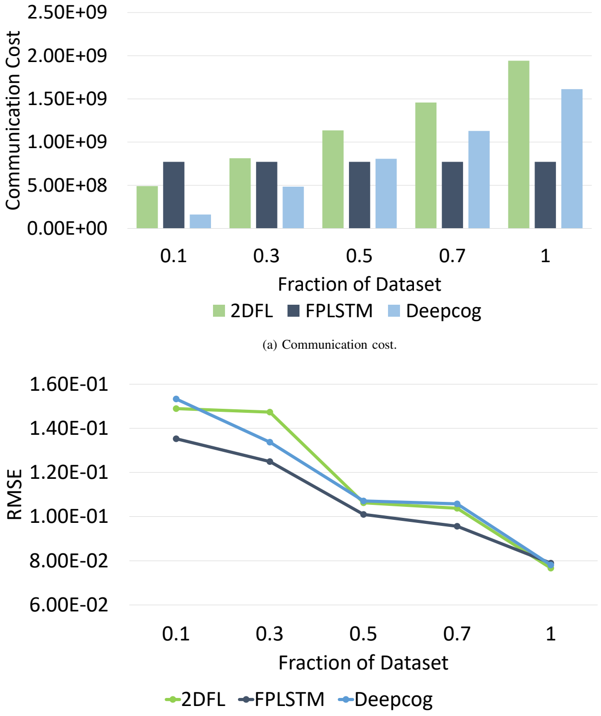
TABLE IV: PERFORMANCE OF FPLSTM VS 2DFL VS Deepcog UNDER DIFFERENT PREDICTION TIME WINDOWS W

	Facebook			YouTube			Google			Instagram		
W	FPLSTM	Deepcog	2DFL	FPLSTM	Deepcog	2DFL	FPLSTM	Deepcog	2DFL	FPLSTM	Deepcog	2DFL
1	0.0981	0.0943	0.0933	0.0605	0.0640	0.0625	0.0581	0.0578	0.0580	0.0963	0.0964	0.0929
3	0.1093	0.1077	0.1075	0.0693	0.0735	0.0737	0.0602	0.0606	0.0605	0.1083	0.1086	0.1062
5	0.1149	0.1166	0.1160	0.0722	0.0761	0.0763	0.0609	0.0615	0.0614	0.1125	0.1135	0.1112
10	0.1258	0.1333	0.1331	0.0752	0.0780	0.0782	0.0632	0.0637	0.0635	0.1183	0.1220	0.1193
15	0.1331	0.1420	0.1414	0.0777	0.0794	0.0799	0.0650	0.0657	0.0657	0.1227	0.1278	0.1253

TABLE V: PERFORMANCE OF FPLSTM VS 2DFL VS Deepcog UNDER DIFFERENT OBSERVATION TIME WINDOWS H

	Facebook			YouTube			Google			Instagram		
H	FPLSTM	Deepcog	2DFL	FPLSTM	Deepcog	2DFL	FPLSTM	Deepcog	2DFL	FPLSTM	Deepcog	2DFL
2	0.1030	0.0932	0.0923	0.0615	0.0629	0.0619	0.0594	0.0575	0.0570	0.0962	0.0945	0.0920
5	0.0981	0.0943	0.0933	0.0605	0.0640	0.0625	0.0581	0.0578	0.0580	0.0963	0.0964	0.0929
10	0.0976	0.0948	0.0947	0.0607	0.0642	0.0641	0.0589	0.0583	0.0581	0.0971	0.0968	0.0949
15	0.0982	0.0958	0.0953	0.0609	0.0654	0.0647	0.0590	0.0589	0.0589	0.0976	0.0977	0.0954
20	0.0986	0.0963	0.0964	0.0623	0.0660	0.0658	0.0595	0.0599	0.0601	0.0984	0.0987	0.0968

dataset (i.e. 100%). Thus, a fraction of the dataset of 0.1 represents a 1-day dataset, and so on.



(a) Communication cost.

(b) RMSE.

Fig. 9: Comparisons of FPLSTM, 2DFL and Deepcog under different fractions of training data.

As seen in Figure 9a, the communication cost of FPLSTM is the same for a different fraction of the dataset, as no actual dataset is transferred between nodes, but only the updates of the global and local agents, representing a constant cost. However, the communication costs of Deepcog and 2DFL increase significantly with the fraction of the dataset used in the training process. 2DFL shows the highest communication cost, as it is in fact the hybrid of the decentralized and centralized approaches.

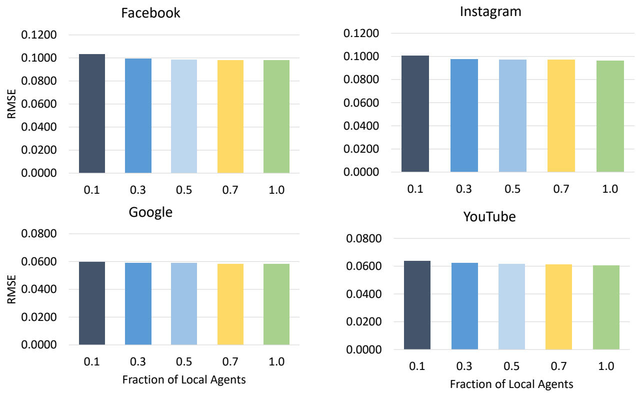


Fig. 10: RMSE value comparison for different fractions of local agents participation for FPLSTM training.

When we only use a small fraction of the dataset for training, the communication costs of both Deepcog and 2DFL actually become lower than that of FPLSTM. However, in these cases, the RMSE values of both Deepcog and 2DFL are less than those of FPLSTM, as shown in Figure 9b. This finding shows that FPLSTM leverages the learning of local models by exchanging knowledge through the global agent, resulting in better sample efficiency than Deepcog and 2DFL. Indeed, Deepcog and 2DFL only achieve slightly better RMSE than FPLSTM when 100% of the training dataset is used, which implies a communication cost that is more than twice as high as that of FPLSTM. It is worth stressing that, although it is the best in forecasting accuracy, 2DFL is probably unrealistic to deploy in a commercial network due to its significantly heavy communication cost.

While FPLSTM shows better communication cost than the benchmarks, there is certainly space for further optimisation. Indeed, the number of local agents participating in each training round has a significant direct impact on the communication overhead [26] and might even transform the global agent in a communication bottleneck. In this light, we investigate the performance of FPLSTM by varying the fraction of local agents participating in the federated training process. We deliberately used different fractions of local agents participation (i.e., 0.1, 0.3, 0.5, 0.7 and 1) in each federated training and generated the results accordingly. Needless to say, a lower fraction of local agents induce lower communication and computation costs. However, those benefits come with the cost of a slight degradation in forecasting accuracy, as not enough local agents take part in the collaborative learning. According to the results shown in Figure 10, our prior arguments are validated. However, it is worth noting that the performance degradation is hardly noticeable up to a fraction of 0.5 of local agents. Hence, to aim at further reducing the overall network cost, using a smaller fraction of local agents could be one of the best alternatives to consider.

Figure 11 depicts the per CPU utilization time of FPLSTM, Deepcog and 2DFL for the four considered slices. Intuitively, since FPLSTM and 2DFL train all of their local models in parallel, they offer better utilization of CPU time than Deepcog, where training is done sequentially. Furthermore, a very light-weight LSTM model with optimal hidden units is used in our FPLSTM, and we adapted a simple 2DCNN model for the 2DFL approach. In contrast, Deepcog has a much more complex model design with three layers of 3DCNNs. The results shown in Figure 11 reveal that the CPU utilization of FPLSTM and 2DFL is much lower than that of Deepcog. Deepcog requires approximately 18 and 9 times the CPU time of 2DFL and FPLSTM, respectively. It is also notable in Figure 11 that CPU utilization of Deepcog model varies for each slice. This is due to the fact that different slices exhibit different temporal patterns, affecting the convergence time of the ML model. The computation time does not seem correlated with the RMSE values obtained for each application.

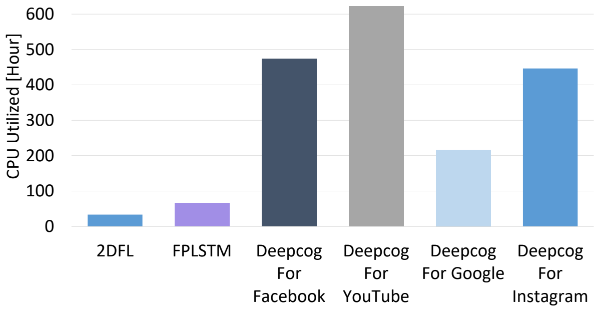


Fig. 11: Per CPU time utilization comparison of FPLSTM, 2DFL and Deepcog.

G. Heterogeneity

The main reason for using a proximal term in our FL approach was the impact we expected from network heterogeneity on forecasting accuracy. Therefore, we evaluate the capabilities of FPLSTM to handle data heterogeneity, by comparing its performance with the one of FALSTM, where federated averaging is applied for global strategy. Figure 12 shows FPLSTM (with different μ values) behaving well in our heterogeneous network environment, and producing a lower RMSE than FALSTM in all scenarios. Just for experimental purposes, we apply different μ values, such as $\mu = [0.001, 0.01.0.1, 0.5, 1]$, in our proximal equation (Line 9 of Algorithm 2). We can see that $\mu = 1$ provides slightly better RMSE values than the other μ values.

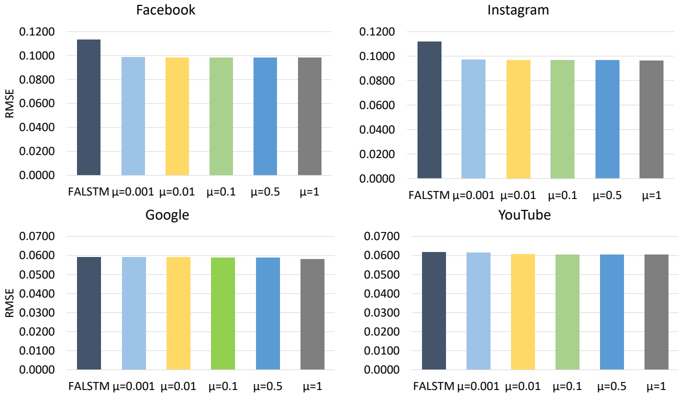


Fig. 12: RMSE values compared to those of FALSTM and FPLSTM (different μ) values.

H. Service Level Agreement Violations

To fully comprehend the benefits of our proposed forecasting solutions in network slicing, we turn our attention to a network-related metric, namely the number of service level agreement (SLA) violations. Practically, an SLA violation occurs when the MVNO reserves too few resources and can not satisfy the user traffic demand. In anticipatory networking, resource reservation is based on the traffic demand forecasting function, so this metric quantifies the impact of our studied forecasting approaches on the network performance.

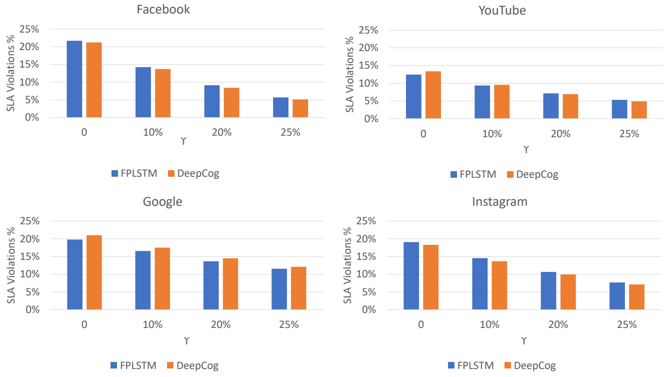


Fig. 13: SLA violations comparison of FPLSTM and DeepCog under different γ.

Figure 13 shows the percentage of time intervals where the MVNO encounters an SLA violation, using FPLSTM and DeepCog. Of course, if the MVNO uses no margin and reserves precisely the amount of predicted resources, even a very low forecasting error produces an SLA violation. This results in SLA violations 20% of the time. For this reason, we use a parameter, γ, which represents an over-provisioning margin the MVNO will consider with respect to the predicted traffic value. This highly reduces the number of SLA violations, for both DeepCog and FPLSTM. The two approaches actually give very similar results, showing once again that a well designed FL solution can be on par with a state of the art centralised approach.

Analysing the four applications individually, we notice that the Google slice, with more frequent and significant peaks, produces more SLA violations. Even for higher values of the γ over-provisioning margin, the number of SLA violations is more than double when compared to the other slices. However, at the same time, this is the only slice for which FPLSTM outperforms DeepCog with respect to this metric.

I. Scalability

Regardless of its solid performance, as explained in Section V, FPLSTM can run into scalability issues if only one single global model is used for an entire network of local agents. Thus, we put forward the clustering-based FPLSTM in which base stations are clustered based on their traffic trend, their location or at random. Accordingly, in Figure 14, we compare the extended clustering-based techniques, IC-FPLSTM (Traffic Trend), IC-FPLSTM (Latitude/Longitude) and RC-FPLSTM, with our prior FPLSTM proposal. As expected, IC-FPLSTM approaches, in general, show better performance than FPLSTM and RC-FPLSTM. This is mainly because the datasets used by local agents in IC-FPLSTM at each cluster are somewhat correlated and developing a corresponding global model with such data accelerates the convergence and increases the performance of IC-FPLSTM. Since the bars in Figure 14 are very close, we also plot in the figure the percentage of improvement of IC-FPLSTM (Traffic Trend) over FPLSTM, denoted as α. This shows that the performance of the former approach is approximately 0.4% to 1.5% better than that of the latter. It is fair to say that the results of clustering-based approaches are able to perform as good as or even better than FPLSTM in most cases.

Interestingly, RMSE values of RC-FPLSTM are only slightly higher than their counterparts (even slightly better than FPLSTM in Facebook slice). The random clustering approach has two operational advantages: *i)* it does not require storing any additional information, such as traffic trends, and *ii)* the clustering algorithm itself is straightforward. While we focus on the better performing IC-FPLSTM in the following, the results obtained for RC-FPLSTM demonstrate that even a simple clustering step can efficiently manage any scalability issues in our FPLSTM framework.

The two IC-FPLSTM flavors, the one based on traffic trend and the one based on geographical coordinates, perform very similarly, with a slight advantage for the traffic-trend version. Therefore, in the following, we focus on the results of the traffic-trend solution, as the best performing one. However, we also need to consider that traffic-trend clustering is more challenging, since it requires storing historical data and potentially presents data privacy issues. Nevertheless, the objective of this work is to show that clustering-based approaches can be efficiently included in the forecasting framework. The precise choice and parameters of the clustering process will depend on the operational context.

While IC-FPLSTM is a good rival for FPLSTM in terms of forecasting accuracy, we measure the CPU time of IC-FPLSTM to really state if it is an efficient solution for the large-scale network. In this regard, Figure 15 depicts the per CPU time utilization comparison of FPLSTM and IC-FPLSTM for different number of local agents. In general, the CPU time of IC-FPLSTM is much preferable to that of the FPLSTM approach, as IC-FPLSTM presents 4-6 times better CPU utilization than FPLSTM. Specifically, the CPU utilization of FPLSTM increases significantly, reaching 65 hours, while the CPU utilization of IC-FPLSTM increase gradually, only 15 hours with 228 local agents. This demonstrates that the proposed clustering-based approach can scale our federated proximal approach to an actual real-world deployment.

The communication cost of IC-FPLSTM is the same as the one of FPLSTM if the corresponding global models reside at the central core node. On the other hand, if each global agent is placed at the edge, which is located in close proximity to local agents, IC-FPLSTM would bring better communication efficiency than FPLSTM. However, the communication model we use in this work does not account for the placement of the agents in the networks, so we can not evaluate the impact of the clustering approach in this case.

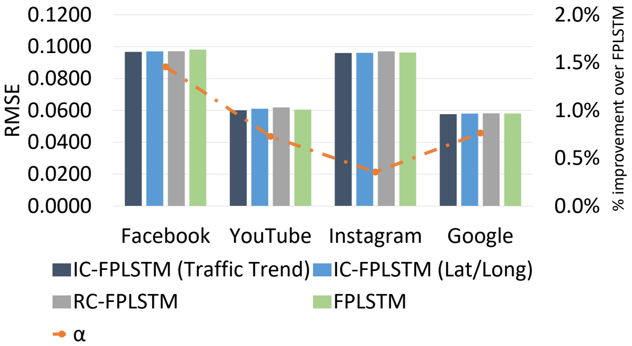


Fig. 14: RMSE values comparisons of IC-FPLSTM (Traffic Trend), IC-FPLSTM (Latitude/Longitude), RC-FPLSTM and FPLSTM.

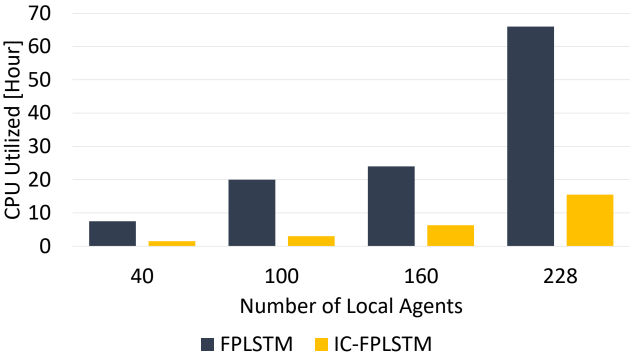


Fig. 15: Per CPU time utilization comparison of FPLSTM and IC-FPLSTM under different number of local agents.

J. Global Agent Performance

Very few studies focusing on FL evaluate the performance of the global model. However, in our case, the performance of the global model is important. As explained, slice activation and deactivation are dynamic, so our system faces a cold start issue, where local agents, associated with slice instances, need to perform forecasting without actually having significant historical data. In this vein, to strengthen the effectiveness of our extended mechanism, we evaluate not only the local models performance but also the one of the global model.

Therewith, Figure 16 depicts the global model performance comparison of the traffic-trend based IC-FPLSTM and FPLSTM in terms of average RMSE over four different slice instances. In general, the global model performance of IC-FPLSTM is 6% better than FPLSTM. The superior performance of IC-FPLSTM can be attributed to the clustering-based approach that we adopt. Recall that the local models of IC-FPLSTM are grouped based on their temporal similarity before performing the collective training. With this, it is sensible to highlight that if the global models are updated by local models with more similarity, better RMSE is achieved at global model.

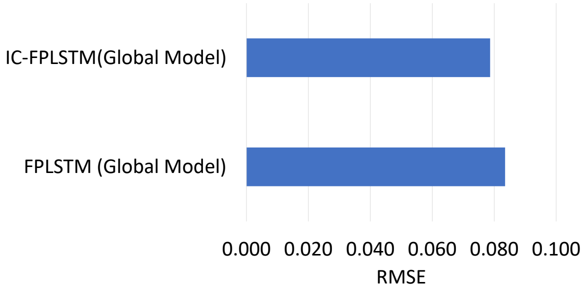


Fig. 16: Global model performance comparison of IC-FPLSTM (traffic trend) and FPLSTM.

VII. Conclusion

In this paper, we propose the federated learning-based FPLSTM framework, in which MVNO local models are allowed to train with their own datasets and only share the weight of the models with a central entity. This allows each MVNO to gain knowledge from their peers and leverage the training of their respective models, while respecting data privacy. Our results, obtained using real mobile network data, show that FPLSTM obtains a similar forecasting accuracy when compared with centralised solutions, with a reduced communication and computation cost. However, these costs can still be prohibitive in ultra dense network deployments of beyond 5G networks. Therefore, we further propose the IC-FPLSTM clustering approach to cope with the large-scale network and computation cost efficiency. Based on a series of simulations on a real-world dataset, the advantages of IC-FPLSTM in terms of accuracy, scalability and robustness are clear; the IC-FPLSTM outperforms the contemporary state-of-the-art centralized solutions and baseline models. Overall, we can state that our proposed approach is able to achieve similar accuracy to a centralized approach, while ensuring data privacy, scalability, data heterogeneity, sample efficiency, communication, and computation efficiency. While the proposed methods are well grounded for multi-slice traffic forecasting, we plan to integrate them into a resource management framework to further appreciate their feasibility in the resource efficiency perspective of network slicing.

References

[1] NGMN, "Description of Network Slicing Concept," *NGMN 5G Project Requirements & Architecture – Work Stream E2E Architecture*, vol. 1, pp. 1–7, January 2016.

[2] GSMA, "From Vertical Industry Requirements to Network Slice Characteristics," August 2018.

[3] Ericsson, "Network Slicing: A Go-To-Market Guide to Capture the High Revenue Potential," Tech. Rep., 2021.

[4] T. Shimojo, M. R. Sama, A. Khan, and S. Iwashina, "Cost-efficient Method for Managing Network Slices in a Multi-service 5G Core Network," *Proceedings of IFIP/IEEE International Symposium on Integrated Network and Service Management (IM)*, pp. 1121–1126, 2017.

[5] S. O. Oladejo and O. E. Falowo, "5G Network Slicing : A Multi-Tenancy Scenario," *Proc. of Global Wireless Summit (GWS)*, pp. 88–92, 2017.

[6] A. J. Gonzalez, J. Ordonez-Lucena, B. E. Helvik, G. Nencioni, M. Xie, D. R. Lopez, and P. Grønsund, "The isolation concept in the 5g network slicing," in *European Conference on Networks and Communications (EuCNC)*, 2020, pp. 12–16.

[7] W. Guan, H. Zhang, and V. C. Leung, "Slice Reconfiguration based on Demand Prediction with Dueling Deep Reinforcement Learning," in *IEEE Global Communications Conference (GlobeCom)*, 2020, pp. 1–6.

[8] Y.-J. Liu, G. Feng, Y. Sun, S. Qin, and Y.-C. Liang, "Device Association for RAN Slicing Based on Hybrid Federated Deep Reinforcement Learning," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 12, pp. 15 731–15 745, 2020.

[9] Y. Kim, S. Kim, and H. Lim, "Reinforcement Learning based Resource Management for Network Slicing," *MDPI Applied Sciences*, vol. 9, no. 11, 2019.

[10] G. Sun, K. Xiong, G. O. Boateng, G. Liu, and W. Jiang, "Resource Slicing and Customization in RAN with Dueling Deep Q-Network," *Elsevier Journal of Network and Computer Applications*, vol. 157, no. 102573, 2020.

[11] C. Marquez, M. Gramaglia, M. Fiore, A. Banchs, and X. Costa-Pérez, "Resource Sharing Efficiency in Network Slicing," *IEEE Transactions on Network and Service Management*, vol. 16, no. 3, pp. 909–923, 2019.

[12] N. Bui and J. Widmer, "Data-Driven Evaluation of Anticipatory Networking in LTE Networks," *IEEE Transactions on Mobile Computing*, vol. 17, no. 10, pp. 2252–2265, 2018.

[13] D. Bega, M. Gramaglia, M. Fiore, A. Banchs, and X. Costa-Perez, "DeepCog: Optimizing Resource Provisioning in Network Slicing with AI-Based Capacity Forecasting," *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 2, pp. 361–376, 2020.

[14] H. Zhao, S. Deng, Z. Liu, Z. Xiang, J. Yin, S. Dustdar, and A. Zomaya, "DPoS: Decentralized, Privacy-Preserving, and Low-Complexity Online Slicing for Multi-Tenant Networks," *IEEE Transactions on Mobile Computing*, vol. 21, no. 12, pp. 4296–4309, 2022.

[15] S. Redana, O. Bulakci, C. Mannweiler, L. Gallo, A. Kousaridas, D. Navrátil, A. Tzanakaki, J. Gutiérrez, H. Karl, P. Hasselmeyer, A. Gavras, S. Parker, and E. Mutafungwa, "5G PPP Architecture Working Group - View on 5G Architecture, Version 3.0," June 2019.

[16] V. Sciancalepore, K. Samdanis, X. Costa-Perez, D. Bega, M. Gramaglia, and A. Banchs, "Mobile Traffic Forecasting for Maximizing 5G Network Slicing Resource Utilization," *Proc. IEEE Conference on Computer Communications (Infocom)*, May 2017.

[17] J. Mei, X. Wang, and K. Zheng, "An Intelligent Self-sustained RAN Slicing Framework for Diverse Service Provisioning in 5G-beyond and 6G Networks," *Intelligent and Converged Networks*, vol. 1, pp. 281–294, December 2020.

[18] C. Zhang, M. Fiore, and P. Patras, "Multi-service Mobile Traffic Forecasting via Convolutional Long Short-Term Memories," *Proc. IEEE International Symposium on Measurements & Networking (M&N)*, pp. 1–6, 2019.

[19] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated Optimization in Heterogeneous Networks," in *Proc. Conference on Machine Learning and Systems (MLSys)*, 2020.

[20] J. B. Monteil, J. Hribar, P. Barnard, Y. Li, and L. A. Dasilva, "Resource Reservation within Sliced 5G Networks: A Cost-Reduction Strategy for Service Providers," *Proc. IEEE International Conference on Communications Workshops (ICC Workshops)*, 2020.

[21] M. Yan, G. Feng, J. Zhou, Y. Sun, and Y.-C. Liang, "Intelligent Resource Scheduling for 5G Radio Access Network Slicing," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 8, pp. 7691–7703, 2019.

[22] T. V. K. Buyakar, H. Agarwal, B. R. Tamma, and A. A. Franklin, “Resource Allocation with Admission Control for GBR and Delay QoS in 5G Network Slices,” *Proc. International Conference on Communication Systems and Networks (COMSNETS)*, pp. 213–220, 2020.

[23] B. Brik and A. Ksentini, “On Predicting Service-oriented Network Slices Performances in 5G: A Federated Learning Approach,” in *Proc. IEEE 45th Conference on Local Computer Networks (LCN)*, November 2020, pp. 164–171.

[24] C. Zhang, S. Dang, B. Shihada, and M. S. Alouini, “Dual Attention-based Federated Learning for Wireless Traffic Prediction,” *Proc. IEEE Conference on Computer Communications (Infocom)*, 2021.

[25] Y. Kim, E. A. Hakim, J. Haraldson, H. Eriksson, J. M. B. Da Silva, and C. Fischione, “Dynamic Clustering in Federated Learning,” *Proc. IEEE International Conference on Communications (ICC)*, pp. 16–21, 2021.

[26] Y. Liu, J. J. Yu, J. Kang, D. Niyato, and S. Zhang, “Privacy-Preserving Traffic Flow Prediction : A Federated Learning Approach,” *IEEE Internet of Things Journal*, vol. 7, no. 8, pp. 7751–7763, 2020.

[27] H. Hirayama, Y. Tsukamoto, S. Nanba, and K. Nishimura, “RAN Slicing in Multi-CU/DU Architecture for 5G Services,” *IEEE Vehicular Technology Conference (VTC Fall)*, 2019.

[28] S. Yang, X. Yu, and Y. Zhou, “LSTM and GRU Neural Network Performance Comparison Study: Taking Yelp Review Dataset as an Example,” in *International Workshop on Electronic Communication and Artificial Intelligence (IWECAI)*, 2020, pp. 98–101.

[29] K. Hu, Y. Li, M. Xia, J. Wu, M. Lu, S. Zhang, and L. Weng, “Federated Learning: A Distributed Shared Machine Learning Method,” *Complexity*, no. 8261663, 2021.

[30] M. Zhang, E. Wei, and R. Berry, “Faithful Edge Federated Learning : Scalability and Privacy,” *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 12, pp. 3790–3804, 2021.

[31] H. Sak, A. Senior, and F. Beaufays, “Long Short-Term Memory Based Recurrent Neural Network Architectures for Large Vocabulary Speech Recognition,” *arXiv*, 2014.

[32] C. Marquez, M. Gramaglia, M. Fiore, A. Banchs, C. Ziemlicki, and Z. Smoreda, “Not All Apps Are Created Equal: Analysis of Spatiotemporal Heterogeneity in Nationwide Mobile Service Usage,” in *Proc. 13th International Conference on Emerging Networking EXperiments and Technologies (CoNEXT)*, 2017, p. 180–186.

[33] Q. Xia, W. Ye, Z. Tao, J. Wu, and Q. Li, “A Survey of Federated Learning for Edge Computing: Research Problems and Solutions,” *High-Confidence Computing*, vol. 1, no. 1, p. 100008, June 2021.

[34] Z. Que, Y. Zhu, H. Fan, J. Meng, X. Niu, and W. Luk, “Mapping Large LSTMs to FPGAs with Weight Reuse,” *Journal of Signal Processing Systems*, vol. 92, no. 9, pp. 965–979, 2020.

[35] D. J. Beutel, T. Topal, A. Mathur, X. Qiu, T. Parcollet, and N. D. Lane, “Flower: A Friendly Federated Learning Research Framework,” *arXiv*, 2020.

[36] Y. Abiko, T. Saito, D. Ikeda, K. Ohta, T. Mizuno, and H. Mineno, “Flexible Resource Block Allocation to Multiple Slices for Radio Access Network Slicing Using Deep Reinforcement Learning,” *IEEE Access*, vol. 8, pp. 68 183–68 198, 2020.

Hnin Pann Phyu received the M.Sc. degree in communication networks and services from the Institut Mines-Telecom, France, in 2016, and the M.Eng. degree in telecommunications from the Asian Institute of Technology, Thailand, in 2016. From 2016 to 2020, she was a Network Strategist and Architect Engineer with Telenor, Myanmar. She is currently pursuing a Ph.D. degree in Software and IT Engineering with École de Technologie Supérieure University, Montreal, Canada. Her current research interests include next-generation mobile communication systems, machine learning, big data analytics, resource management and network slicing.

Razvan Stanica is an associate professor at INSA Lyon, France, and a research scientist with the Inria Agora team of the CITI laboratory. He obtained a M.Eng. degree and a Ph.D. in computer science, both from INP Toulouse, France, in 2008 and 2011 respectively. His research interests include wireless mobile networks, with a special focus on communication networks in urban environments.

Diala Naboulsi is an Assistant Professor at the École de Technologie Supérieure (ÉTS), Canada. Before that, she held a Research Professional position in the Ultra-TCS research chair, at ÉTS, Canada, and a Research Associate position and a Postdoctoral Researcher position at Concordia University, Canada. She was also a Visiting Researcher at Ericsson, Canada. She held Course Lecturer positions at McGill University, Canada and Concordia University, Canada. She obtained the Ph.D. degree in Computer Science from INSA Lyon, France in 2015. As part of a double degree program, she received in 2012 the M.Sc. degree in Computer Science from INSA Lyon, France and the M.Eng. degree in Telecommunications from the Lebanese University, Lebanon. Her research interests are in mobile networks, virtualized networks, and wireless networks. She holds an NSERC Discovery Grant (2020-2026) on network slicing in future mobile networks. She has collaborations with several providers in communications technology such as Ultra Intelligence & Communications and Octasic.