Mobile Traffic Forecasting for Network Slices: A Federated-Learning Approach

Hnin Pann Phyu, Diala Naboulsi, Razvan Stanica

Mobile Traffic Forecasting for Network Slices: A Federated-Learning Approach

Hnin Pann Phyu
Département de Génie Logiciel et TI
École de Technologie Supérieure (ÉTS)
Montreal, Canada
hnin.pann-phyu.1@ens.etsmtl.ca

Diala Naboulsi
Département de Génie Logiciel et TI
École de Technologie Supérieure (ÉTS)
Montreal, Canada
diala.naboulsi@etsmtl.ca

Razvan Stanica
Univ Lyon, INSA Lyon, Inria, CITI
Villeurbanne, France
razvan.stanica@insa-lyon.fr

***Abstract*—Network slicing is one of the cornerstones for next-generation mobile communication systems. Specifically, it enables Mobile Virtual Network Operators (MVNOs) to offer various types of services over the same physical infrastructure owned by an Infrastructure Provider (InP). To satisfy the dynamic user requirements and ensure resource efficiency, MVNOs need to estimate the future traffic demand in advance, to pre-allocate/reconfigure the resources at the base stations. However, this per-slice traffic forecasting exploits information that is clearly sensitive for the MVNOs from a business point of view, and which might even disclose private data regarding some users. Hence, it is vital for MVNOs to ensure data privacy while conducting traffic forecasting. Bearing this in mind, 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 without compromising data privacy. Simultaneously, an InP global model is updated through the aggregation of local models weights. Prediction results obtained by training the models on a real-world dataset indicate that the forecasting performance of FPLSTM is as accurate as state-of-the-art solutions, while ensuring data privacy, computation and communication cost efficiency.**

***Index Terms*—Network Slicing, Traffic Forecasting, Data Privacy, Machine Learning**

I. Introduction

Network slicing is designed to leverage the development of future mobile communication services. It allows Mobile Virtual Network Operators (MVNOs) to offer various services, e.g. enhanced Mobile BroadBand (eMBB), Ultra-Reliable and Low-Latency communication (URLLC), and massive Machine-Type Communication (mMTC), with diverse Quality of Service (QoS) requirements. These services run over the same network infrastructure, operated by an Infrastructure Provider (InP) [1].

Considering the dynamic nature of mobile applications traffic, ensuring an efficient allocation of resources while meeting QoS requirements remains a critical aspect for MVNOs [2]. This is especially the case in the Radio Access Network (RAN) [3].

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 ANR CANCAN (ANR-18-CE25-0011) and the ANR MAESTRO5G (ANR- 18-CE25-0012) projects.

There, both static and dynamic resource allocation strategies are possible. While a static strategy is simple to implement, resources assigned at base stations for a specific slice instance could be unused or in surplus during the slice lifetime. A dynamic strategy allows instead to adapt resource allocation to traffic dynamics [4] and thus enables a more efficient utilization of resources [5].

To this end, an accurate traffic forecast is needed for reconfiguring resources allocation [6]. More precisely, traffic forecasting for individual slices, at each base station, can assist MVNOs to reconfigure resources at each base station, enhancing by that the resources utilization over the RAN.

At this level, different options were proposed in the literature for traffic forecasting [7], [8]: *i)* MVNOs having their own centralized forecasting models, *ii)* MVNOs having their own decentralized forecasting models, and *iii)* MVNOs sharing their data with the InP to train one centralized forecasting model. In this regard, the first approach would incur high communication costs, as large datasets are transferred to the central node. The second approach does not account for possible correlations among base stations and increases the training time for the local models which is more likely to incur additional computation cost.

In the third approach, sensitive and possibly private data is compromised. More precisely, since MVNO slices are attached to diverse types of base stations (i.e. macro, micro, pico, and femto), the dataset shared with the centralised InP model may include either private domain data or personal domain data. Specifically, private domain data covers all the generated/collected data belonging to a private organization, such as aggregated traffic data of macro base stations. Despite having a small chance of exposing the individual user identity through aggregated macro base station traffic, the business strategy, and potential revenues of MVNOs could be revealed.

On the other hand, personal domain data covers all individual or household data which can easily expose the individual identity. For instance, aggregated data from small-cell (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 by any third-party organizations.

Overall, our argument is that MVNOs ought to safeguard

their data only within their premises, not only to secure their business strategy, but also to respect privacy rights of individual users.

One promising solution to overcome those foregoing defects is a decentralized collaborative machine learning scheme, also known as Federated Learning (FL). With respect to our problem, FL holds three unique features: *i)* privacy-preservation, *ii)* communication-efficiency, and *iii)* computation-efficiency [9]. In essence, FL enables a privacy-preserving forecasting framework where MVNOs can collaboratively train their forecasting models without sharing any sensitive data among each other. With this, MVNOs would gain knowledge from their counterparts through collaborative learning, improving their learning performance. In this case, the InP appears as a good candidate for being the centralized coordinator, as it is connected to the different MVNOs. Specifically, the global model is controlled by a global InP agent at the central node and local models are managed by local agents of MVNOs at base stations. It is noteworthy that the InP has a direct interest in improving the forecasting accuracy of the MVNOs, since this would optimize the overall resource demands from their parts.

Keeping in mind the points stated above, the contributions of this paper are twofold. First of all, we propose a privacy-preserving FL-based forecasting algorithm which can be used by MVNOs for base stations level predictions. Our solution integrates FL with a well-known time-series forecasting technique, Long Short-Term Memory (LSTM), to enhance the forecasting accuracy of multi-slice network traffic. Second, we conduct an extensive evaluation of our solution on a real-world traffic dataset to exhibit its effectiveness compared to the existing state-of-the-art solutions. Our results show that the proposed Federated Proximal Long Short-Term Memory (FPLSTM) framework presents a forecasting accuracy comparable to the best centralized solutions in the literature, while preserving data privacy. As a side result, we show that FPLSTM highly reduces the communication and computation costs.

The remaining part of this paper is assembled as follows. Section II reviews related works of traffic forecasting in network slicing. 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. We discuss the results in Section V and conclude the paper in Section VI.

II. Related Works

A considerable amount of studies have investigated traffic forecasting in the context of network slicing. Specifically, the authors in [10] use a modified version of LSTM to study the problem of future demands forecasting for individual slice services. By combining the sequence-to-sequence learning paradigm and convolutional long short-term memory (S2SConvLSTM), high accuracy is achieved for hourly traffic forecasting. In [11], the authors build a collaborative learning framework of LSTM (used for large-timescale hourly traffic forecasting of each slice) and Asynchronous Advantage Actor Critic (A3C), used for small-timescale predictions over intervals in the order of milliseconds, to enable efficient resource utilization while considering performance isolation among slices.

Similarly, the authors of [7] introduce a centralized forecasting framework employing the 3-dimensional convolutional neural network (3DCNN) method for resource demand forecasting over slices, with the purpose of reducing the overall resource provisioning costs. In their case, the authors consider only slice-level aggregated traffic.

The authors in [12] aim at efficiently forecasting service performance of slices while preserving the privacy of each slice. To this end, they employ an Artificial Neural Network (ANN) driven FL forecasting scheme.

Overall, it has been observed that all existing traffic forecasting solutions in network slicing only focus on aggregate slice-level forecasting. In addition, except for the work in [12], all solutions do not take into account privacy issues. In contrast to these studies, we focus in our work on traffic forecasting per slice at the level of individual base stations, a critical aspect for MVNOs to enable efficient resource allocation. We propose a federated learning approach that allows MVNOs to collectively forecast traffic efficiently, benefiting from each others knowledge, while preserving each MVNO's data privacy.

III. System Model and Problem Statement

In this section, we present our system model and introduce the problem of base station level per-slice traffic forecasting.

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. We consider $\mathcal{B}$ as 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 use $s \in \mathcal{S}$ to represent a slice instance and $\mathcal{S}$ to represent the set of slice instances. Besides, we consider 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. In our work, we consider time is slotted and we use t to refer to one time interval and T to refer to a set of time intervals of interest. We denote as $v_b^s(t)$ the generated traffic volume at base station $b \in \mathcal{B}_s$ of slice instance $s \in \mathcal{S}$ at time $t \in \mathcal{T}$.

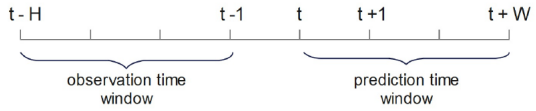


Fig. 1: Observation time window and prediction time window

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 forecast traffic evolution over a prediction window W, between time instants t and $t+W$. Figure 1 visualizes the observation and prediction time windows.

IV. Algorithm Design

To solve our problem, we introduce the FPLSTM solution, combining an FL approach, known as Federated Proximal, and LSTM. In this section, we start by discussing the number of dimensions in our problem, followed by the presentation of the employed federated learning approach and the proposed model.

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 correlations between these three dimensions, which implies a high system complexity and usually requires a significant training time.

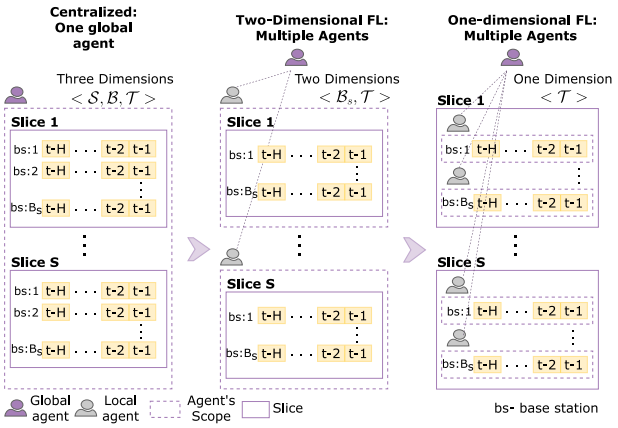


Fig. 2: Problem dimensions in centralized and FL approaches.

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 on the middle of Figure 2. This 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. By that, the system complexity is reduced, as well as the computation load at each local agent, while collaboration between slices is provided by the global agent. However each FL agent covers multiple base stations, so an important communication cost is still apparent, to centrally collect the required data. These centralized and two-dimensional FL approaches discussed above hold both spatial and temporal information.

With communication costs in mind, we further reduce the input of our problem to 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 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 in the realm of the FL framework 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 [13].

B. Federated Learning Approach

We employ the one-dimensional FL approach described in the previous subsection. Accordingly, each local agent trains its own local model with its own local dataset. On the other hand, the global agent aggregates the weights of the local models in order to update its global model. It is worth mentioning that no dataset leaves the base station premises and henceforth data privacy is secured. Figure 3 illustrates the functioning of our proposed FL forecasting framework.

Herein, the overall objective is to minimize the global loss function $GL(.)$ and local loss function $LL(.)$ of global models and local models respectively. Mathematically, the objective can be represented as:

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

where w_J is the weight matrix of the neural network acting as global model J (i.e. 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. Moreover, 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 total number of samples $a = \sum_{s \in \mathcal{S}} \sum_{b \in \mathcal{B}_s} a_b^s$.

It is worth stressing that FL does not require data to be independent and identically distributed (IID). Accordingly, it can be used with MVNOs potentially holding non-IID datasets in practice.

C. Proposed Model

In this section, we present our proposed FPLSTM approach, which 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.

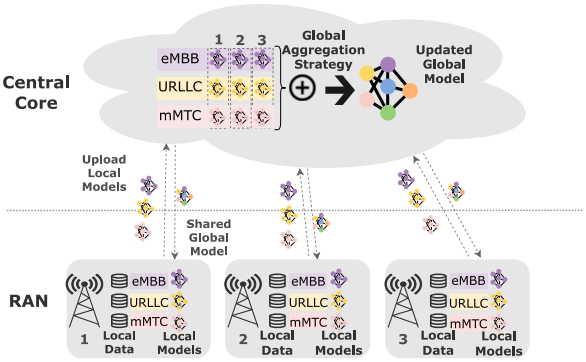


Fig. 3: FL-driven slice traffic forecasting architecture.

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 federated proximal (FedProx) [14] as a global aggregation strategy in our FPLSTM approach. FedProx integrates a proximal term into the local sub-problems, allowing it to handle training data heterogeneity. The complete model training processes of the FPLSTM framework are formally described in Algorithm 1 FPLSTM: `Global Agent(GA)` and Algorithm 2 FPLSTM: `Local Agent(LA)`, respectively.

Algorithm 1: FPLSTM: `Global Agent(GA)`

Input: $C, \mathcal{N}, \mu, w^0, \Omega$

1 Initially $w_J^c = w^0$, $c = 0$
2 **for** *each communication round* $c = 0, 1, .., C-1$ **do**
3 GA randomly selects Ω of local agents $\mathcal{N}^c \subseteq \mathcal{N}$
4 GA sends w_J^c 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 $w_{J_b^s}^{c+1}$ from all local agents $\mathcal{N}^c$
7 GA updates $w_J^{c+1} = \frac{1}{|\mathcal{N}^c|} \sum_{n_b^s \in \mathcal{N}^c} w_{J_b^s}^{c+1}$
8 GA updates `global_model(`w_J^{c+1}`)` ◁ *No Dataset is required*
9 **end for**

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.

Therewith, the algorithm starts by initializing the global model weight matrix w_J^c with w^0 for $c = 0$ (Line 1). For each communication round, the GA randomly selects a Ω fraction of local agents $\mathcal{N}^c$ from the set $\mathcal{N}$ (Line 3). At the same time, GA shares the latest global weight matrix w_J^c with all the chosen local agents $\mathcal{N}^c$ (Line 4). Also, we denote $n_b^s \in \mathcal{N}^c$ as local agent of slice s associated to base station b. After that, in parallel, each local agent trains its own local model 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 2: FPLSTM: `Local Agent(LA)`

Input: X_b^s, β, $\mathcal{E}$, w_J^c, η, μ, c

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

As for Algorithm 2, the input includes a dataset X_b^s, the batch size β, the local epochs $\mathcal{E}$, the global model weight w_J^c, the learning rate η, the hyper-parameter μ and the global communication round c. Firstly, LA updates its local model weights matrix $w_{J_b^s}^c$ with the latest available global model weights matrix w_J^c (Line 1). Afterwards, LA trains its `local_model(`$w_{J_b^s}^c$`,` X_b^s`)` according to the defined local epochs E and batch size β (Line 2 - Line 8). Specifically, we rely on the Stochastic Gradient Decent (SGD) approach to update the local model weight matrix, where $w_{J_b^s}^*$ is the local weight matrix in which the loss function is minimized or converged, $\bigtriangledown$ is the gradient of loss function $l(w_{J_b^s}, \beta)$ of batch size β with the learning rate η (Line 6). After than, instead of updating the local model weight matrix $w_{J_b^s}^{c+1}$ with the weight matrix $w_{J_b^s}^*$ of minimized local loss function $LL(w_{J_b^s}^*)$, proximal term is integrated into $LL(w_{J_b^s}^*)$ so as to limit the effect of local model update to the overall global model. Then, the local model weights matrix $w_{J_b^s}^{c+1}$ is updated with the minimum weights matrix such as: $w_{J_b^s}^{c+1} \approx argmin_{w_{J_b^s}} h(w_{J_b^s}^*, w_J^c)$ (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 bigger effect than $\mu = 0.01$). Finally, the LA sends the local model weight matrix back to the `Global Agent` (Line 10).

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

V. Evaluation

In this section, we present the evaluation of our methodology. We begin by introducing the dataset we used. Then, we present the benchmarks and performance metrics that we use. Finally, we discuss the obtained results.

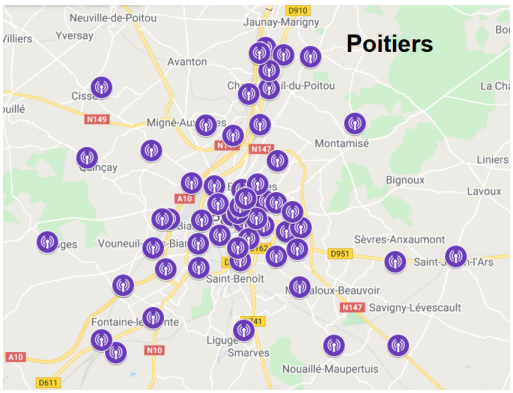


Fig. 4: 4G sites locations in Poitiers, France.

A. Dataset

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

Since network slicing has not been deployed yet in the 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. In fact, those four different applications are used for very different purposes and requisite different bandwidth and latency [15], which is well aligned with the concept of network slicing.

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

B. Benchmarks and Performance Metrics

We compare our proposed FPLSTM with two other counterparts: Deepcog [7] and fully Decentralized LSTM (DLSTM). More specifically, Deepcog, based on a 3DCNN model, is one of the best known centralized forecasting frameworks proposed in the state of the art. For the DLSTM model, we train all the local agents separately (no global model) and employ the same parameters as our FPLSTM.

We rely on the Root-Mean-Square Error (RMSE) to measure the performance of the algorithms. Therewith, we train all the models with the objective of minimising the RMSE value between predictions and ground truth. All average RMSE results in the following refer to the test dataset only. 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 [16] 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. size of machine learning model or raw dataset size). 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 [17]. 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 use to train our models.

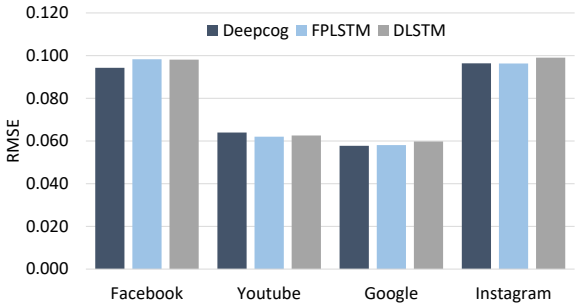


Fig. 5: RMSE values for FPLSTM, Deepcog and DLSTM.

C. Implementation and Overall Results

We implement our proposed FPLSTM solution using the Flower Federated Framework [18]. All the models (i.e. FPLSTM, Deepcog, and DLSTM) are simulated in a Python environment along with open-source TensorFlow libraries. After that, both FPLSTM and counterparts models were trained on the high-performance Linux servers provided by Compute Canada nonprofit organization [19]. Since we have considered four different applications and 57 base stations, there are a total of 228 local models participating in FPLSTM training. Besides, there is one global model to aggregate all the weights of the local models. We have tuned the hyperparameters of our model through more than a hundred time trial-and-error processes and we summarize them in Table I.

We first compare the performance of our FPLSTM solution with its counterparts (i.e. Deepcog and DLSTM). Figure 5 depicts the average RMSE values of FPLSTM, Deepcog, and DLSTM for the four considered slices. As exhibited in this figure, the RMSE values of FPLSTM and DLSTM are indeed generally higher than those obtained by the centralized Deepcog solution, but the difference is rather small (and even favorable to the decentralised solutions for the Youtube slice).

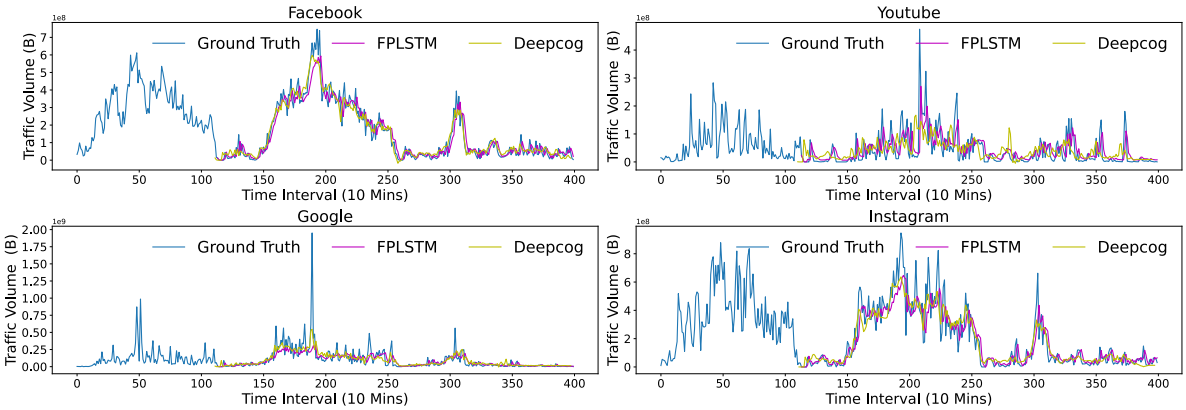


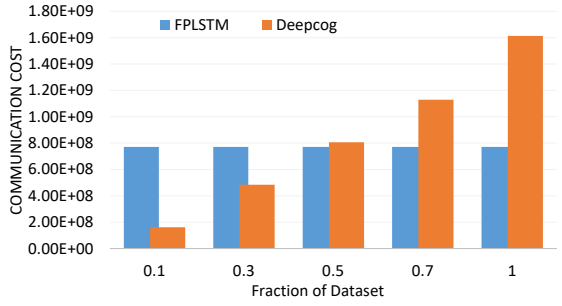
Fig. 6: Forecasting results vs Ground truth for Facebook, Youtube, Google and Instagram for FPLSTM and Deepcog.

TABLE I: List of parameters

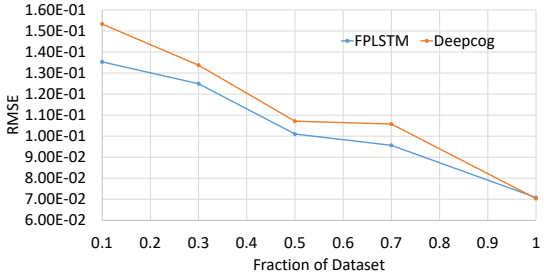
Parameter	Value
Number of Global Agents	1
Number of Local Agents $\mathcal{N}$	228
Global model	LSTM:64 hidden units, flatten and fully-connected layers
Local model	LSTM:64 hidden units, flatten and fully-connected layers
Percentage of training set	80 % of the dataset
Percentage of testing set	20 % of the dataset
Communication round C	20
Loss functions (global and local)	RMSE
Local epoch $\mathcal{E}$	100
Batch size β	16 samples
Learning rate η	0.0001
Decay learning rate	0.01

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 [20]. Thus, new slices could be deployed in the network at any time in the future. In such a scenario, DLSTM, as a non-collaborative approach, needs to train new models for those new slices from scratch. Instead, new agents in FPLSTM can easily obtain an already trained model from the global agent, removing this cold start problem encountered in DLSTM. For this reason, in the following, we merely focus on the performance comparison of FPLSTM and Deepcog. To better visualize the forecasting performance achieved by FPLSTM and Deepcog, in Figure 6 we plot their resulting predicted traffic trend and associated ground truth for different applications. Once again, we can observe that FPLSTM achieves very similar results to Deepcog.

Furthermore, we compare the communication cost of FPLSTM and that of the centralized Deepcog approach. Moreover, we vary the fraction of the dataset used by the training process. Practically, we consider the 10 days dataset as a full dataset (i.e. 100%). Thus, a fraction of the dataset of 0.1 represents a 1-day dataset and so forth. As seen in Figure 7a, 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 cost of Deepcog increases significantly with the fraction of the dataset used in the training process. When a low fraction of the dataset is used, the communication cost of Deepcog actually becomes lower than that of FPLSTM. However, in these cases, the RMSE value of Deepcog is inferior to the one of FPLSTM, as shown in Figure 7b. This finding confirms that FPLSTM leverages the learning of local models by exchanging knowledge through the global agent, exhibiting better sample efficiency than Deepcog. Indeed, Deepcog only achieves a better RMSE than FPLSTM when 100% of the training dataset is used, which implies a communication cost almost two times higher than FPLSTM.



(a)



(b)

Fig. 7: Comparisons of FPLSTM and Deepcog under different fractions of data: (a) Communication cost (b) RMSE.

Figure 8 depicts the utilization time per CPU, in hours, of FPLSTM and Deepcog for the four slices. Intuitively, since FPLSTM trains all the local models in parallel, it induces better CPU utilization time than Deepcog, where training is done sequentially. Moreover, we use a very simple LSTM model in our FPLSTM approach. Conversely, Deepcog has a much more complex model design, with three layers of 3DCNN. The results shown in Figure 8 are the proof of these intuitions, as CPU utilization of FPLSTM is approximately 2 to 6 times better than Deepcog. It is also notable in Figure 8 that CPU utilization of Deepcog model is varied for each slice. This is due to the fact that different slices exhibit different temporal patterns, affecting the convergence time of the ML model.

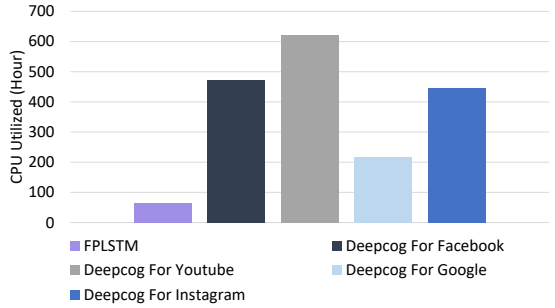


Fig. 8: Per CPU utilization time comparison of FPLSTM and Deepcog.

The above results demonstrate that the proposed FPLSTM approach is able to achieve an accuracy similar to a state of the art centralized solution, while keeping sensitive and private data on the premises of the MVNOs. At the same time, we show that FPLSTM is much more efficient, both communication and computation-wise.

VI. Conclusion

In this paper, we address the multi-slice traffic forecasting problem, to facilitate intelligent and anticipatory resource management. Therewith, we propose the federated learning-based FPLSTM framework, where the MVNOs' local models are allowed to train with their own dataset and only share parameters of the models. With the help of a global agent managed by the InP, each MVNO gains knowledge from peers and leverages it in training the local agents, while respecting data privacy. According to a series of experiments on a real-world dataset, the performance of FPLSTM is shown to be as accurate as that of a state-of-the-art centralized solution, while ensuring data privacy, and improving communication and computation efficiency. Speaking of which, our results demonstrate that federated learning in general, and our FPLSTM approach in particular, is appropriate for base station level traffic forecasting in a sliced network architecture.

As future mobile networks are expected to rely on ultra dense deployments, scalability remains a critical issue for resources management solutions. Consequently, our future works include the extension of our FPLSTM solution to tackle the scalability issue under the umbrella of anticipatory resource management in network slicing.

References

[1] V. P. Kafle, Y. Fukushima, P. Martinez-Julia, and T. Miyazawa, "Consideration on Automation of 5G Network Slicing with Machine Learning," in *Proc. 10th ITU K*, Santa Fe, Argentina, 2018.

[2] W. Guan, H. Zhang, and V. C. Leung, "Slice reconfiguration based on demand prediction with dueling deep reinforcement learning," in *Proc. IEEE Globecom*, Taipei, Taiwan, 2020.

[3] 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.

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

[5] 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.

[6] 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.

[7] D. Bega, M. Gramaglia, M. Fiore, A. Banchs, and X. Costa-Perez, "DeepCog: Cognitive Network Management in Sliced 5G Networks with Deep Learning," in *Proc. IEEE Infocom*, Paris, France, 2019.

[8] 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*, pp. 1–14, 2021.

[9] A. Imteaj, U. Thakker, S. Wang, J. Li, and M. H. Amini, "A Survey on Federated Learning for Resource-Constrained IoT Devices," *IEEE Internet of Things Journal*, vol. 9, no. 1, pp. 1–24, 2022.

[10] C. Zhang, M. Fiore, and P. Patras, "Multi-service Mobile Traffic Forecasting via Convolutional Long Short-term Memories," in *Proc. IEEE M&N*, Catania, Italy, 2019.

[11] 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.

[12] B. Brik and A. Ksentini, "On Predicting Service-oriented Network Slices Performances in 5G: A Federated Learning Approach," in *Proc. IEEE LCN*, Sydney, NSW, Australia, 2020.

[13] S. Yang, X. Yu, and Y. Zhou, "LSTM and GRU Neural Network Performance Comparison Study: Taking Yelp Review Dataset as an Example," in *Proc. IWECAI*, Shanghai, China, 2020.

[14] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated Optimization in Heterogeneous Networks," in *Proc. MLSys*, 2020.

[15] C. Marquez, M. Gramaglia, M. Fiore, A. Banchs, and X. Costa-Perez, "How should i slice my network? a multi-service empirical evaluation of resource sharing efficiency," *Proceedings of the Annual International Conference on Mobile Computing and Networking, MOBICOM*, pp. 191–206, 2018.

[16] Q. Xia, W. Ye, Z. Tao, J. Wu, and Q. Li, "A Survey of Federated Learning for Edge Computing: Research Problems and Solutions," *Elsevier High-Confidence Computing*, vol. 1, no. 1, 2021.

[17] 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.

[18] D. J. Beutel, T. Topal, A. Mathur, X. Qiu, T. Parcollet, and N. D. Lane, "Flower: A Friendly Federated Learning Research Framework," *ArXiv*, vol. abs/2007.14390, 2020.

[19] "Compute Canada Research Portal," https://www.computecanada.ca/.

[20] 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.