Deep learning based prediction of traffic peaks in mobile networks

Deep Learning Based Prediction of Traffic Peaks in Mobile Networks

Shuyang Li[a,*], Enrico Magli[a], Gianluca Francini[b], Giorgio Ghinamo[b]

[a] *Politecnico di Torino, Corso Duca degli Abruzzi, 24, 10129 TO, Torino, Italy*
[b] *Telecom Italia, Corso Duca degli Abruzzi, 24, 10129 TO, Torino, Italy*

Abstract

To better maintain mobile networks, accurate mobile traffic prediction is crucial for resource allocation. For the adjacent cells, sometimes the mobile traffic concentrates in a single cell, leading to traffic unbalance; in many cases, the detection of unbalance is strongly related to the prediction of traffic peaks, which is extremely difficult because many peaks appear suddenly for no apparent reason. To better predict the peaks and traffic unbalance, we propose two novel mobile traffic predictors. The first is a Mixture of Experts (MoE) model which yields significantly better peak prediction along with excellent interpretability by establishing a cooperation mechanism between different experts, whereas the second predictor is a light-weight Multilayer Perceptron (MLP) which can obtain similar peak forecasting performance but operating in a more flexible way and consuming less computational power. The obtained predictions are then used to aid the predictive detection of traffic unbalance. To this end, we first perform a large-scale analysis of mobile traffic, and then propose different approaches to detect the future traffic unbalance based on the predictions. Extensive experiments are carried on real-world mobile traffic datasets, and the results show the superior performance of our proposed solution at predicting peaks and traffic unbalance.

Keywords: Mobile networks, Mobile traffic forecasting, Traffic unbalance prediction

1. Introduction

In the last decade, mobile traffic has grown rapidly on a global scale because of a massive increase of mobile devices and their applications, along with the development of modern cellular networks. According to Ericsson annual report 2022, mobile data traffic volume is estimated to increase by more than two times in the period 2023–2027, and the mobile video traffic forecasted to grow by almost 30% annually through 2027 [1]; as video streaming applications become increasingly popular, mobile video traffic now accounts for more than half of all mobile data traffic, which makes base stations sustain much higher mobile demand than before. Compared to other mobile services, streaming services have a more strict requirement of the Quality of Service (QoS); how to improve the QoS and maintain the stability of mobile networks are becoming a critical issue for mobile network operators.

Among the network performance indicators of LTE networks, the QoS of mobile users is strongly related to the downlink user throughput whose value is affected by many factors. In practice, the network operator is interested in improving not only the average user throughput of the networks but also the worst 5% user throughput [2, 3]. In mobile networks, cells can be seen as the most fundamental element of the access network layer. The access network layer is composed of elements called evolved Node B (eNodeB) or E-UTRAN Node B; eNodeB is responsible for connecting the mobile devices within its coverage to mobile network [4, 5]. Each eNodeB consists of multiple cells, and these cells are connected targets of mobile device. For a set of cells, it is not rare that the mobile traffic distributes among them in an unbalanced way. In this case, most of the traffic is concentrated in a single cell leading to traffic congestion and reduction of throughput available to the mobile users assigned to that cell; from this point of view, a balanced traffic distribution among cells is preferable as the QoS of the congested cell would be improved. In practical, a set of adjacent cells in a given geographical area can be seen as a small cluster, and it is feasible to move traffic from a congested cell to the adjacent ones if the traffic distributes among them in an unbalanced way.

To configure cells and manage the mobile traffic, Coverage and Capacity Optimization (CCO) and Mobility load balancing (MLB) have been widely used [6, 7]. CCO [8, 9] finds the proper radio parameters to satisfy the requirements of both the coverage and the capacity, whereas MLB [10] modifies the cell-specific offset between neighboring cells adaptively depending on their load difference; if the load difference exceeds the presetting threshold, then the algorithm will be triggered to control the traffic handover between different cells. With these techniques, a mobile network operator can modify the offset related to the han-

*Corresponding author
Email addresses: shuyang.li@polito.it (Shuyang Li), enrico.magli@polito.it (Enrico Magli), gianluca.francini@telecomitalia.it (Gianluca Francini), giorgio.ghinamo@telecomitalia.it (Giorgio Ghinamo)

dover or switch the configuration between predefined ones to encourage traffic redistribution. However, no matter what method is applied to control the traffic handover, there is always a problem related to the decision making. In general, there is a delay of obtaining the latest network measurements, and it is risky to use the delayed observations to reconfigure the network because the updated configuration could be already outdated and may not adapt well to the actual traffic pattern; this problem becomes more critical in the presence of traffic peaks, which are critically important towards QoS. From this point of view, using predictions of future mobile traffic could be more beneficial than using the latest observations. Furthermore, predictions can also enhance the stability of the system which is crucial for network operators.

While accurate mobile traffic predictions are essential, this task is challenging since, in many cases, the time series of mobile traffic parameters exhibit a generally stationary behavior interrupted by sudden strong peaks. Many deep learning-based time series forecasting methods are very good at predicting the stationary parts, but provide inaccurate results in the predictions of peaks, and the reason lies in the fact that forecasting models are often trained to minimize an average loss on the prediction error. Since peaks occur rarely, their predictions tend to be neglected because their effect on the average loss is quite small, whereas the loss is dominated by errors in the stationary parts of the mobile traffic time series; this leads to models typically making conservative predictions most of the time. If the model predicts the future steps in an aggressive way, this would improve the prediction of future peaks but also make the overall performance worse as the aggressive predictions are always risky. Eventually, there is always a trade-off between average loss and peak prediction, which has to be handled according to application requirements. Generally for mobile network operators, it is preferable that the model makes more accurate peak predictions even though the error on the non-peak parts is slightly higher, as the peaks are the most valuable part of mobile traffic and accurate peak predictions allow mitigating QoS degradation promptly.

For tackling these challenges and improving the traffic management of mobile networks, in this paper we propose a new method to detect and adjust the potential traffic unbalance within a cluster of cells; our solution is composed of two different phases: mobile traffic forecasting phase and traffic unbalance detection phase. In the first phase, two deep learning models can be selected to predict the future traffic: Mixture of Quantiles (MoQ) and Flexible Multilayer Perceptron (FMLP). MoQ was initially proposed in [11] aiming at providing accurate mobile traffic predictions while obtaining good peak forecasting performance; in this paper, we use the original implementation of MoQ to conduct experiments, and expand the work to the application of load unbalance prediction. MoQ is built based on MoE framework supporting flexible blending of different forecasting styles, where aggressive and conservative forecasting are adaptively aggregated based on the recent temporal dynamics of the time series. Through the cooperation between experts with diverse characteristics, this model is capable of making better peak predictions while maintaining excellent overall performance and interpretability. Since MoQ is a computational expensive model, we also propose the FMLP light-weight model. FMLP consists of a predictor and a data mapping module, which selects important samples and performs conditional magnitude scaling on the time series, which allows the model to perform forecasting in a flexible way. The behavior of this model can be easily controlled by changing a pre-defined scaling factor, resulting in different levels of aggressiveness in the predictions. In the traffic unbalance detection phase, the predictions of cells are used by a detection algorithm to determine if there would be a potential traffic unbalance; in this paper, two predictive detection approaches are discussed: single-model approach and multi-model approach. Compared to the naive approach which uses the recent observations instead of the predictions, both of the predictive solutions achieve better results on a real-world dataset. Specifically, the main contributions of our work are summarized as follows:

- A predictive approach to detect the potential mobile traffic unbalance among cells combining a mobile traffic predictor with an unbalance detection algorithm.
- A novel mobile traffic forecasting model called MoQ, which features a flexible blending of conservative and aggressive predictions based on recent observations.
- A novel light-weight mobile traffic forecasting model called FMLP, which focuses on learning important patterns; The model behavior can be tuned from conservative to aggressive prediction.
- A large scale analysis is performed to study the traffic behavior within the clusters, and extensive experiments are carried out on real-world mobile traffic datasets, proving the advantages provided by the proposed methods.

This paper is structured as follows. In Section 2, we review previous works related to the corresponding topics. Section 3 and Section 4 presents the details of the proposed MoQ and FMLP models. Section 5 presents the proposed mobile traffic predictor and detection algorithm. In Section 6 we introduce two real-world datasets and perform experiments and analysis. In Section 7, we draw the conclusions and future works.

2. Background and Problem Formulation

2.1. Mobile Traffic Management and Network Configuration

For the topology planning and mobile traffic management, MLB and CCO are the two popular approaches.

CCO solutions focus on adjusting the antenna tilt to optimize the capacity and coverage, affecting the network performance and traffic assignment [12, 13, 14]; how to adjust the antenna tilt efficiently is the major concern in this field [9, 15, 16]. Compared to CCO, MLB does not configure the antennas but modifies the handover region between adjacent cells leading to lower call drop rate and better QoS [10, 17, 18, 19]. Few papers study the problem of detecting the potential traffic load unbalance; in this work, we propose a traffic prediction-based approach to determine if there is a need to redistribute load among neighboring cells in near future.

2.2. Time Series Forecasting

Time series forecasting plays a key role in many industrial domains such as climate analysis [20], retail forecasting [21] and traffic modelling [22]. Conventional methods focus on parametric models built based on the knowledge of domain experts including autoregressive model [23]. Recently, deep learning has attracted increasing attention in time series field because it can model complex non-linear relationships [24]. For example, [25] has shown that MLP outperforms the ARIMA autoregressive model in time series prediction [26, 27, 28], and [29] obtains good forecasting performance using Recurrent Neural Network (RNN)-based models. To make more accurate predictions, more complex neural networks are employed in these years including Transformer-based models [30] and Temporal Convolutional Network (TCN) [31]; recently, more advanced forecasting models have been proposed including DLinear [32], Informer [33], Autoformer [34] and the others. Among these works, very few of them investigate how to obtain better peak prediction, which is crucial in many applications.

2.3. Mixture of Experts

MoE model was first proposed in [35], where the authors introduce an ensemble style framework which consists of many sub-networks (experts) and a gating network (manager). In a MoE, each expert is expected to learn different knowledge from training data, and the outputs of experts are mixed by the manager to combine benefits from diverse patterns. This idea has attracted a lot of attention both in deep learning and machine learning fields. [36, 37] build MoE-style models on top of machine learning algorithms such as support vector machines. [38] propose deep MoE models for speech enhancement. In natural language processing, [39] introduces a very large sparsely-gated MoE and obtains significantly better results for machine translation task. MoE has also proven effective for computer vision tasks such as image classification and person re-identification [40, 41]. Another solution is to build layer-level MoE as in [42], which creates a deep learning model composed of multiple MoE layers, where each MoE layer is an individual MoE network, and the layers are stacked together. In the present paper, a model-level MoE is built, introducing a cooperation mechanism enabling MoQ to make good peak predictions.

2.4. Quantile Loss & Quantile Regression

For a real-valued random variable Y, denote as $F(y)$ its cumulative distribution function. Given a quantile index $\tau \in (0, 1)$, the τ-quantile $q^{(\tau)}$ is defined as $q^{(\tau)} = F^{-1}(\tau)$. For example, the 0.5-quantile $q^{(0.5)}$ is the median. For many applications, risk and uncertainty have to be quantified to make optimal decisions. The need of modelling distribution leads to the use of Quantile Regression [43]. The purpose of quantile regression is to predict the conditional τ-quantile $\hat{y}^{(\tau)}_{t+k}$ given the past observations $y_{:t}$ and a predefined quantile index $\tau \in (0, 1)$. Comparing to probabilistic forecasting, quantile regression is more robust because it makes no assumption on the data distribution [44]. In order to predict the quantile, the model has to be trained to minimize the total Quantile Loss (also called pinball loss):

$$QL_\tau(y, \hat{y}^{(\tau)}) = \tau(y - \hat{y}^{(\tau)})_+ + (1 - \tau)(\hat{y}^{(\tau)} - y)_+, \quad (1)$$

where $(\cdot)_+ = \max(0, \cdot)$, and $\hat{y}^{(\tau)}$ is modelled by function $g_\tau(\cdot)$ which can be approximated with a deep learning model. Many works have been done in quantile regression field. [45] proposes a model to make quantile forecasting of the load of smart grid, and a new approach is proposed to determine the prediction intervals. [46] combines quantile regression and multitask learning, and extends the application to spatiotemporal problems. In our work, instead of using quantile predictions to define a prediction interval, quantile predictions are used in an ensemble way, where each expert is trained by minimizing quantile loss with different quantile index τ, resulting in different levels of aggressiveness of forecasting.

3. Mobile Traffic Predictor: MoQ

3.1. Problem Formulation

In this work, we perform short-term time series forecasting using a model learned from past observations, and the predictions are further used to determine if a traffic unbalance event will occur in near future. Assuming we want to predict the future values of univariate time series with w forecast horizons, time series is represented as a vector $\mathbf{x}_{1:t} = [x_1,\ x_2,\ ...,\ x_t] \in R^t$ which consists of past observations of the target mobile network KPI, where x_i is the record collected at time step i and t is the length of sequence; the problem can be formulated as:

$$\hat{\mathbf{x}}_{t+1:t+w} = f(\mathbf{x}_{1:t}), \quad (2)$$

where $\hat{\mathbf{x}}_{t+1:t+w} \in R^w$ is the vector of predictions from time $t + 1$ up to time $t + w$, and $f(\cdot)$ is a selected predictor.

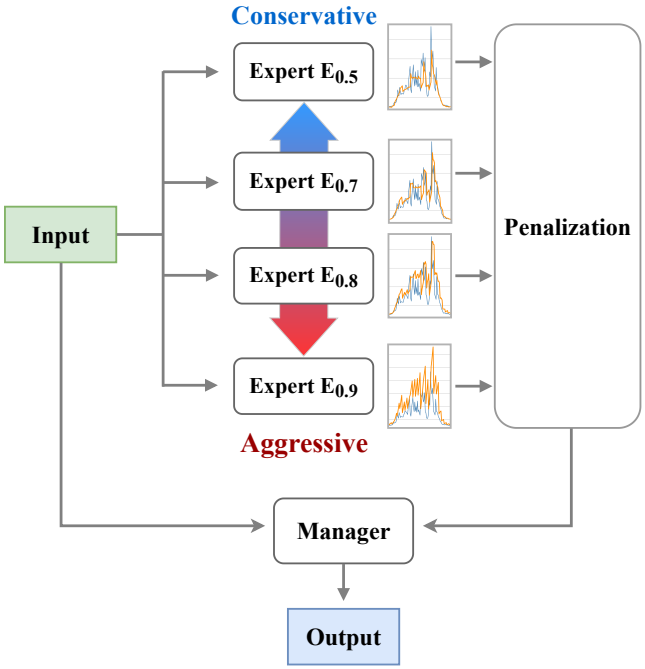

Figure 1: Architecture of Mixture of Quantiles: expert E_q is trained to minimize a quantile loss having quantile index equals to q. The orange line represents forecasting and the blue line represents ground-truth. MoQ combines the prediction of different experts to yield a more accurate prediction, especially around peaks of the time series.

3.2. Mixture-of-Quantiles Architecture

The architecture of MoQ is shown in Figure 1, and it consists of three components:

- *Experts* which are trained to forecast with different aggressiveness levels; the outputs of experts are fed into the manager yielding the final prediction;
- a *Penalization* used to prevent overusing a specific expert and to promote the cooperation among experts;
- a *Manager* aggregating predictions from different experts to output the final forecast.

The idea behind MoQ is to fuse various prediction styles, as illustrated in Figure 1. In MoQ, the input is first fed into the experts pool. By training experts with different objective functions, these expert will have diverse forecasting styles: some of them are going to make aggressive predictions while others will be more conservative. The manager observes the recent temporal behavior of the input series and fuses these predictions based on softmax score; in this way, the model automatically learns when to be conservative and when to be aggressive. The details of MoQ are discussed in the following.

3.2.1. Experts with Various Forecasting Styles

In this work, LSTNet [47] is used as the backbone network of each expert. When applying the MoE framework, how to guarantee that each expert learns different knowledge from the same input is a challenging topic, which would highly affect the model performance; if the experts act in a very similar way, there is no benefit in using an MoE model. To promote diversity among the experts, a specific quantile loss is employed to pretrain each expert individually, and the quantile index is different for each expert. As depicted in Figure 1, MoQ consists of four experts that are pretrained with quantile index 0.5, 0.7, 0.8 and 0.9 respectively. Among the four experts, expert $E_{0.5}$ is the most conservative one, and the prediction becomes more aggressive as the quantile index is increased. Based on that, we can obtain predictors with various forecasting styles in a reliable way. Once the predictions made by each expert are available, the outputs are concatenated as:

$$\hat{\mathbf{x}}^{E}_{t+1:t+h} = [\hat{\mathbf{x}}^{E_{0.5}}_{t+1:t+h}, \hat{\mathbf{x}}^{E_{0.7}}_{t+1:t+h}, \hat{\mathbf{x}}^{E_{0.8}}_{t+1:t+h}, \hat{\mathbf{x}}^{E_{0.9}}_{t+1:t+h}], \quad (3)$$

where $\hat{\mathbf{x}}^{E}_{t+1:t+h} \in R^{k \times N \times h}$, k is the number of experts, N is the size of mini-batch and h is the forecasting horizon. After pretraining using the auantile loss, the weights of experts are frozen and will not be updated during the training of the manager.

3.3. Manager

The manager is a gating function that is responsible for aggregating the output of individual experts. The aggregation is performed based on the recent observations of the time series to capture the local temporal behavior. In this case, "local" refers to the most recent p observations, where p is a hyperparameter. In this work, the manager is composed of a fully connected layer and a softmax activation function. The fully connected layer extracts the local patterns of time series, and the manager calculates the softmax score among experts for each future step, which can be formulated as follows:

$$H = FC_w(\mathbf{x}_{-p:t}), \quad (4)$$

$$S = \text{Softmax(H)}, \quad (5)$$

where FC_w is a fully-connected layer parameterized by the weights/biases w, $\mathbf{x}_{-p:t} \in R^{N \times p}$ is the matrix of recent observation and $S \in R^{N \times h \times k}$ is the expert score. The dimension of hidden vector H has to be converted from $N \times hk$ to $N \times h \times k$ before passing it to the activation function. Based on score S, the final prediction $\hat{\mathbf{x}}_{t+1:t+h}$ is made by fusing the weighted output of experts:

$$\hat{\mathbf{x}}_{t+1:t+h} = \sum_{e \in E} S^e \hat{\mathbf{x}}^e_{t+1:t+h}, \quad (6)$$

where E is the set of experts, $\hat{\mathbf{x}}^e_{t+1:t+h}$ and S^e are the corresponding prediction and score for expert e. To acquire the ability of using experts cooperatively, after the experts pretraining, the manager is trained to minimize the Mean Absolute Error (MAE) between the fused prediction and the ground truth. However, this alone is not sufficient to

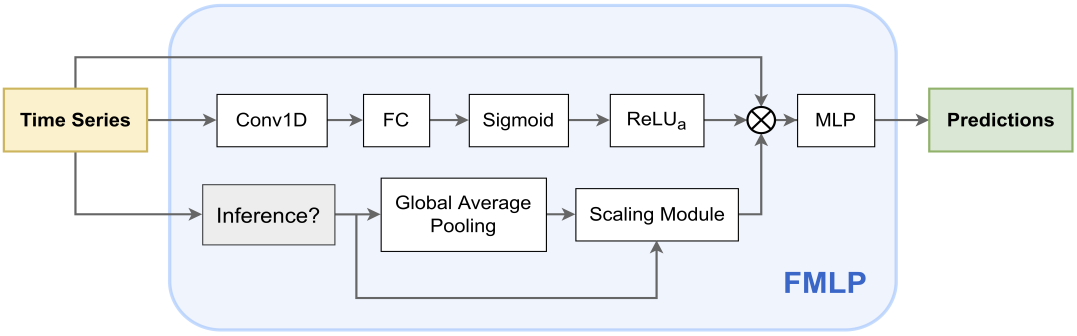

Figure 2: Architecture of FMLP.

guarantee accurate prediction of highly dynamic regions while minimizing the overall loss.

Ideally, the manager should rely on the conservative expert for the forecasting of non-dynamic regions, and on the aggressive experts to predict peaks. However, in most cases, models tend to overuse the most conservative expert which is pretrained to predict the median value, as this yields a low value of the MAE loss. In MoQ, we propose to penalize the manager's nature of being conservative. Specifically, two different penalization methods are introduced.

3.3.1. Penalization Mask

The first option is to use a mask during training, which penalizes conservative experts and encourages the manager to assign higher weights to aggressive experts. The mask is designed as follows:

$$M = \left[\frac{1}{k}, \frac{2}{k}, \ ,..., \ \frac{k}{k}\right], \tag{7}$$

where k is the number of employed experts; the mask is $M = [0.25, 0.5, 0.75, 1.0]$ when we use 4 experts. Each penalization coefficient within the mask corresponds to the expert in the corresponding position of $\hat{\mathbf{x}}^E_{t+1:t+h}$ (see eq. (3)), where the most conservative expert is penalized in the hardest way while the most aggressive expert is not penalized at all. Each expert prediction has to be multiplied by the corresponding mask coefficient before being fed into the manager; this deteriorates the conservative predictions, often making them too low to be used, and only slightly affects the aggressive predictions. In this way, predicting the median value is not the best choice anymore in order to minimize the loss, and the manager is required to learn a smarter way to fuse the results of the individual experts. This is an objective-oriented penalization method, and the mask values can be changed in order to guide the decision of manager in the desired way, thereby controlling the behavior of MoQ.

3.3.2. Penalization via addition of Gaussian Noise

The second approach is to add Gaussian noise to the predictions instead of masking them. The Gaussian noise is designed to have zero-mean and expert-dependent variance. For expert E_τ, the variance of its Gaussian noise is chosen as $\sigma^2_\tau = \alpha \cdot (\frac{1}{\tau} - 1)$, where α controls the spread of the noise variance among the experts. The more conservative the expert, the higher the variance of the Gaussian noise. Comparing to the penalization mask, the addition of Gaussian noise employs only one parameter and does not require the design of a penalization mask; this leads, however, to a somewhat weaker control of the model behavior.

3.3.3. Use of penalization during the training process

It is important to notice that the two introduced penalization methods are only used during the training phase, whereas experts are not penalized during the validation and test phase. Penalization encourages the manager to use aggressive predictors. However, MoQ should make aggressive predictions only when there could be an occurrence of a peak in the upcoming time steps, and not in the stationary part of the time series. To achieve this, we do not apply the penalization for all training samples, but only for the samples whose ground-truth value is much higher than that of the other samples, as these samples have a higher possibility of having peaks in next few steps. Specifically, for each training sample we sum the values of its ground-truth, and only the samples whose value of the summarized ground-truth is in the largest 10% will be penalized. In this way, the manager is forced to learn how to extract local temporal dynamics from recent observations to determine if peak will occur or not.

4. Mobile Traffic Predictor: FMLP

The architecture of FMLP is composed of three components, as shown in Fig. 2.

- an *Information Filter Module* which consists of a convolutional layer, a fully-connected layer and two activation functions. The use of this module is to select important samples from the input and filter the potential noise, eventually creating a mask which weights each sample of the input time series.

- a *Conditional Scaling Module* which is composed of a global average pooling layer and a scaling module. The objective of this component is to perform magnitude scaling for the selected samples based on a pre-defined scaling factor; a mask is created to scale the magnitude of each sample of the input.
- a *MLP* that makes predictions by using the time series masked by the previous modules.

The information filter and conditional scaling modules can be seen together as the "data mapping" module of FMLP. The idea behind FMLP is to design a model whose peak prediction can be customized to have different aggressiveness levels. To this end, the time series is first fed into the two modules to generate two masks selecting the important samples and scaling the magnitude for part of them. According to different setting of the pre-defined scaling factor, the magnitude of the peak predictions can be adjusted in a flexible way; then the original time series is weighted by multiplying these masks and fed into MLP to generate predictions. The details of FMLP are discussed in the following.

4.1. Information Filter Module

This module is used to assign a weight to each sample of the input. The convolutional layer and fully-connected layer are employed to learn the temporal pattern of time series which evaluates the importance of each sample. The extracted hidden vector has the same length as the input, and it is first processed by a sigmoid function to map its value to the range $(0, 1)$; then a parametric ReLU is applied to filter the noise, as:

$$ReLU_\gamma(x) = \begin{cases} 0, & x \leq \gamma \\ x, & x > \gamma \end{cases}, \tag{8}$$

where γ is a pre-defined threshold. Generally, the importance of samples is quite "sparse" as, after the sigmoid, many samples are mapped to very low values such as 0.01. These samples contribute relatively little to the prediction and may be detrimental. Hence, the parametric $ReLU_\gamma$ is used to filter the noise and the parameter γ is set to 0.05 in the experiments.

4.2. Conditional Scaling Module

This module is used to scale the magnitude of the samples of time series in the training phase, but it is not employed in the inference stage. The module first calculates the global average of the time series, then the global average and the original time series are fed into the scaling module to generate a scale mask. Because we want to change the magnitude of the peak prediction without affecting the off-peak part, not all samples need to be scaled. In this case, all the samples whose value is higher than the global average should be multiplied by a pre-defined scaling factor; Due to the nonlinear nature of neural networks, the predictions are robust to the scale change of input series; in this case, even though we scale the magnitude of few samples of input series, the model would still be optimized to make similar predictions. So if we scale the magnitude in training stage and disable the scaling in inference stage, the magnitude of predictions would be scaled in the inverse way without changing the shape of the predictions; the value of the scaling factor should be lower than 1 if we want to increase the magnitude of the peak predictions. For the scenario of time series forecasting, one drawback of the neural network is that the scale of predictions is not sensitive to the scale of the input time series because of the non-linearity introduced by stacking layers [47]. If we decrease the magnitude of peak regions in the training phase, the model will still yield similar predictions during training, but the peak predictions will be increased during the inference phase if we deactivate the scaling module since the weights of layers are optimized to compensate for the magnitude reduction.

4.3. MLP

The MLP used in FMLP is the conventional model proposed by [48], which is composed of three fully-connected layers and ReLU functions. The MLP uses the time series weighted by the two masks to make the predictions.

5. Detection of Traffic Unbalance

5.1. Cluster Creation and Analysis

Due to the network configuration and user movement, mobile traffic often distributes among a cluster of cells in an unbalanced way, and this load unbalance needs to be detected. The successful detection of the potential load unbalance allows the network operators to switch from a configuration optimised for capacity to another suited for redistributing the traffic, leading to improved performance for the congested cell. In this work, a cluster is defined as a set of cells which includes a reference cell (cluster center) plus a few adjacent cells (neighbors). The procedure for creating a cluster is as follows:

1. *Selection of Reference Cell*: A reference cell is more likely to be congested. We choose cells whose 0.9 quantile of its downlink usage time series is higher than 2 Erlang.

2. *Selection of Adjacent Cells*: for the selection of adjacent cells, we consider both the coverage overlap and the served users overlap between the reference and the adjacent cells. We employ the handover data and consider as adjacent cells those whose served users overlap is higher than 20%, and this overlapping setting allows the redistribution of mobile users among cells; among those, the two cells having the largest coverage overlap are selected to form a cluster with the reference cell.

Algorithm 1 Naive Approach

Input: $\lambda_{ref,t}$, R^t_{max}.
Output: L.
if $R^t_{max} \geq 2$ **then**
  if $\lambda_{ref,t} \geq 2$ Erlang **then**
    $L_{t+1} = L_{t+2} = 1$
  else
    $L_{t+1} = L_{t+2} = 0$
  end if
else
  $L_{t+1} = L_{t+2} = 0$
end if

Algorithm 2 Predictive Approach

Input: $\lambda_{ref,1:t}$, $\lambda_{adj1,1:t}$ and $\lambda_{adj2,1:t}$.
Output: L
for Time step $h = t+1$ to $t+2$ **do**
  $\hat{\lambda}_{ref,h} = Predictor(\lambda_{ref,1:t})$
  $\hat{\lambda}_{adja1,h} = Predictor(\lambda_{adj1,1:t})$
  $\hat{\lambda}_{adja2,h} = Predictor(\lambda_{adj2,1:t})$
  $R^h_{max} = \frac{\hat{\lambda}_{ref,h}}{\max_{c \in A} \hat{\lambda}_{c,h}}$
  if $R^h_{max} \geq 2$ **then**
    if $\hat{\lambda}_{ref,h} \geq 2$ Erlang **then**
      $L_h = 1$
    else
      $L_h = 0$
    end if
  else
    $L_h = 0$
  end if
end for

Once we create the clusters, for each cluster we monitor the total ratio R^t_{total}, the maximum ratio R^t_{max} and the downlink usage of the cluster. R^t_{total} is the ratio between the downlink usage of the reference cell and the total downlink usage within the cluster, and R^t_{max} is the ratio between the downlink usage of the reference cell and the maximum downlink usage among its adjacent cells. The ratios are calculated as follows:

$$R^t_{total} = \frac{\lambda_{ref,t}}{\sum_{c \in C} \lambda_{c,t}}, \tag{9}$$

$$R^t_{max} = \frac{\lambda_{ref,t}}{\max_{c \in A} \lambda_{c,t}}, \tag{10}$$

where $\lambda_{ref,t}$ is the downlink usage of the reference cell at time step t, $\lambda_{c,t}$ is the downlink usage of the cell c at time step t, C is the whole cluster and A is the set of adjacent cells. For the time step t, the reference cell is thought as congested and requiring the traffic redistribution if its downlink usage is higher than 2 Erlang and $R^t_{max} > 2$.

5.2. Detection Algorithm

In order to determine if there is a need to redistribute mobile traffic or not, we apply different approaches to detect the traffic unbalance within the cluster based on the downlink usage. The first approach is called "naive" approach and it uses the latest KPI observations of the cells within the cluster to make the decision; this approach is used as the baseline by considering it is quite simple. Another option is the predictive approach which uses the predictions of downlink usage to guide the decision. Either approach provides labels of the next two steps, $L = [L_{t+1}, L_{t+2}]$, where L can be either 0 or 1, and 0 means there is no need to perform traffic redistribution.

The pseudo codes of these approaches are shown as Algorithm 1 and Algorithm 2. Furthermore, the predictive approach can be subdivided into two subcategories based on the applied forecasting models: single-model approach and multi-model approach. Single-model approach uses the same forecasting model to predict the downlink usage for both the reference cell and the adjacent cells, while the multi-model approach uses different models for handling the reference cell and the adjacent cells. For example, Model A+Model B means using Model A for the reference cell and Model B for the adjacent cells. Compared to single-model approach, multi-model approach provides higher flexibility for the decision making; this is discussed in detail in Section 6.3.

6. Experiments and Results

In this section, we conduct experiments on two real-world industrial datasets to evaluate the performance of the proposed approaches; both datasets are provided by Telecom Italia S.p.A which is the largest Italian telecommunications services provider. For the first dataset, the downlink usage data is collected from LTE network of a metropolitan city in Italy, covering 100 cells; this dataset consists of 32300 time series, each time series is recorded during 14 consecutive days, and the traffic profile of each cell is aggregated over 15-minute intervals; the first dataset is used to train and test the forecasting performance of the proposed mobile traffic predictor. The second dataset is much larger; it consists of three months observations of 2713 cells deployed in Piedmont province, Italy, and it is used to perform large scale mobile network analysis and evaluate the performance of the traffic unbalance detector.

6.1. Forecasting of Mobile Traffic Peaks

For the mobile traffic predictor, our target is to predict network parameters for the next 2 steps (30 minutes); all time series of this dataset have been normalized with the standard score normalization, and the normalized time series is calculated by first subtracting the mean value of raw time series and then dividing by the standard deviation.

Metrics	MLP	LSTM	GRU	LSTNet	TCN	MQ-RNN	Transformer	DLinear	Informer	Autoformer	MoQ	$\text{MoQ}_2^\dagger$	$\text{MoQ}_4^\dagger$	$\text{FMLP}_{0.7}$
MAE	0.301	0.295	0.286	**0.282**	0.298	0.290	0.332	0.295	0.343	0.411	0.310	0.297	0.306	0.301
MSE	0.297	0.303	0.298	**0.287**	0.326	0.299	0.342	0.293	0.427	0.481	0.329	0.297	0.310	0.312
Accuracy	67.5%	67.7%	66.1%	67.5%	57.0%	66.2%	69.1%	66.3%	58.7%	56.3%	**77.7%**	75.2%	**77.7%**	75.5%
Sensitivity	36.2%	36.5%	33.1%	36.2%	14.2%	33.4%	39.3%	33.7%	18.3%	13.6%	**58.5%**	52.6%	58.2%	53.5%
#Parameters	188k	265k	50k	102k	61k	141k	1.12M	**5K**	660k	1.14M	409k	409k	409k	189k

Table 1: Performance comparison of different models for mobile traffic prediction and peak classification; k and M are the shorthand notation for Thousand and Million.

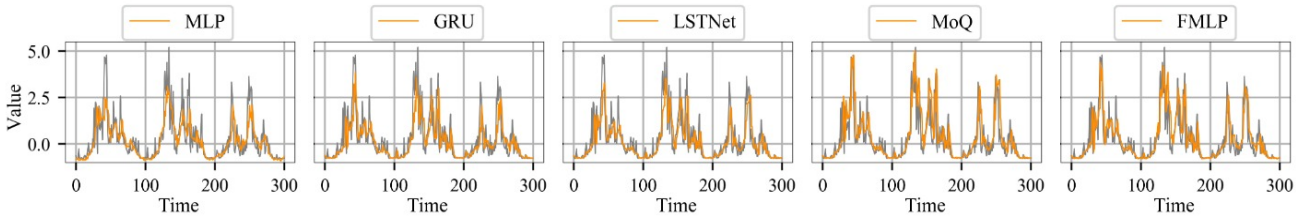

Figure 3: Predictions of downlink usage. The penalization mask of MoQ is employed in the plot and the scaling factor of FMLP is set to 0.7.

6.1.1. Benchmarks and Performance Metrics

Two versions of MoQ are implemented: the MoQ penalized by mask (MoQ) and the MoQ penalized by the Gaussian noise whose variance is controlled by α ($\text{MoQ}_\alpha^\dagger$). The FMLP model is denoted as FMLP_β where β is the predefined scaling factor. We compare the performance of our proposed models against a set of baseline approaches covering the most popular approaches in the time series forecasting field, including: MLP [48], LSTM [49], GRU [50], LSTNet [47], TCN [31], MQ-RNN [51], Transformer [52], DLinear [32], Informer [33] and Autoformer [34]. The hyperparameters of these models have been tuned through grid search based on the performance evaluated on validation set. Once the best configurations have been determined, the models have been retrained on the dataset, by merging training and validation set, and eventually evaluated on the test set. All experiments are conduct on a computer with Intel(R) Core(TM) i7-7700K CPU @ 4.20GHz, 64 GB memory and a single Nvidia TITAN X Pascal 12GB GPU. The operating system is Linux 4.15.0-143-generic, and the model is implemented in Python 3.8.8 with Pytorch 1.7.0.

To evaluate the performance, five metrics are used. Initially, we quantify the overall performance of these models in terms of MAE and MSE (Mean Squared Error). Since MAE and MSE do not properly capture the ability of a model to predict peaks in the time series, which is instead very important in our target application, we also employ classification metrics to evaluate the ability of model at predicting peaks. Specifically, for the downlink usage the peak part is defined as the elements of a tensor whose value is higher than a threshold Q defined as a given quantile of this feature. In this setting, a mobile traffic predictor that forecasts downlink usage two time-steps ahead is employed for binary peak classification, in that if the predicted downlink usage has value greater than or equal to the predefined threshold, a peak is detected. In our experiments, the quantile index is defined as 0.95. Based on this, each ground-truth element is either positive (peak) or negative (non-peak), and a model is assessed on the basis of its ability to correctly classify peaks in terms of the sensitivity metric, also known as recall, i.e. the number of correctly identified peaks among the retrieved peaks. Although sensitivity is the most important metric for the target application, we also need to verify that a model does not overestimate a target frequently by generating a lot of false alarms, so the average classification accuracy among classes is also used to quantify the general performance of predictive classification. Besides these metrics, the number of parameters (#Parameters) is also used to evaluate how many parameters are included in the models, which is a good proxy of the model complexity.

6.1.2. Results

We perform forecasting on the test set of the mobile traffic dataset; the performance of the models is reported in Table 1. For this dataset, we observe that RNN-based models obtain better performance compared to the others. LSTNet has the lowest MAE and MSE and it is the best model if we only consider the overall loss. The performance of Autoformer is the worst among the baselines, its peak prediction performance is much worse than the others and the forecasting performance is quite poor. Comparing with the selected baselines, the proposed MoQ models obtain much higher sensitivity. If we compare the performance between the mask version MoQ (MoQ) and the Gaussian noise versions ($\text{MoQ}_\alpha^\dagger$), the difference between them is quite small, and both of them provide a significant improvement of peak predictions. Among these MoQ models, MoQ has the highest sensitivity whose value is 19.2% higher than the highest sensitivity of baselines (39.3%), which means this model is much more capable to predict potential upcoming peaks; MoQ also has the highest classification accuracy, showing that the proposed

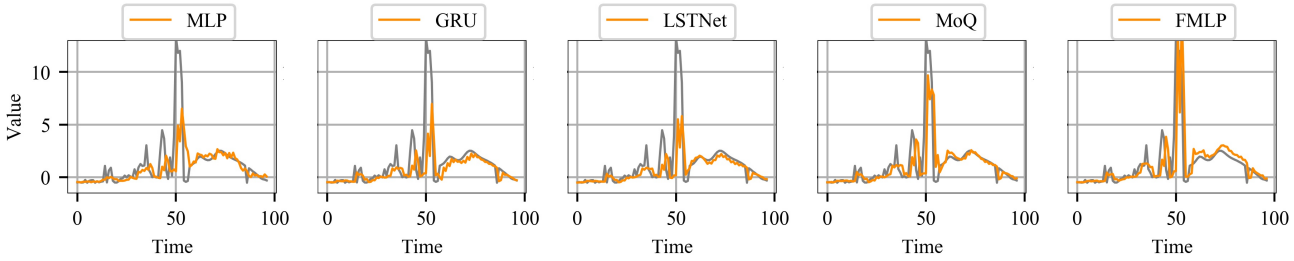

Figure 4: Peak predictions of downlink usage. The penalization mask of MoQ is employed in the plot and the scaling factor of FMLP is set to 0.7.

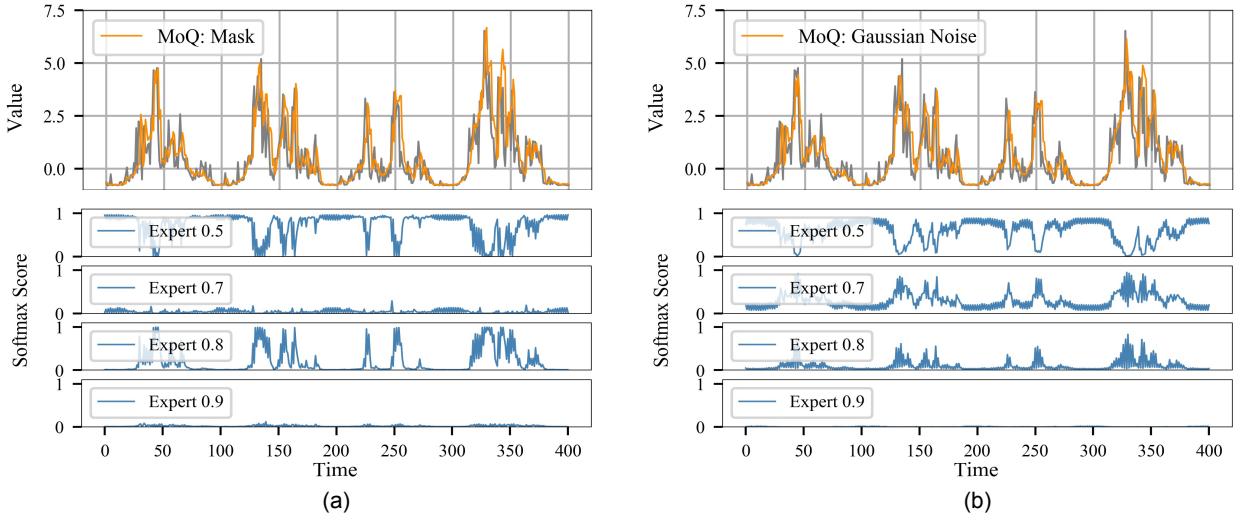

Figure 5: Visualization of the cooperation between experts, the expert scores used to fuse the predictions are visualized. (a) MoQ (mask version); (b) MoQ (Gaussian noise version).

model is able to perform forecasting based on the recent trend which better reflects the traffic pattern. The price to be paid for very good sensitivity is that its MAE and MSE are slightly higher than those of LSTNet, though still to close to those achieved by the best methods. Comparing $\text{MoQ}_4^\dagger$ and $\text{MoQ}_2^\dagger$, we observe that the MAE and MSE are reduced by decreasing the value of α while obtaining slightly lower sensitivity and classification accuracy; by modifying the value of α, it is possible to make a trade-off between peak prediction and overall performance. Comparing to MoQ, FMLP obtains slightly worse forecasting performance, but its performance is still much better than the baselines for the task of mobile traffic peak forecasting. If we compare the complexity of the models, we can find that MoQ is not a very heavy model even though it is built based on the expert system, and it is still lighter than some baseline approaches such as the transformer-based models; to have a faster and lighter version of MoQ, FMLP would be a good candidate as its model size is smaller.

To better understand the behavior of different architectures, we select several models and visualize their forecasting. Figure 3 and Figure 4 illustrate the general traffic predictions and the peak predictions respectively, where Figure 3 refers to the general pattern of downlink usage and Figure 4 refers to a bursty increase of downlink usage in mobile networks. In Figure 3, we can observe that the predictions made by MoQ and FMLP can better follow the trend and peaks of mobile traffic; this behavior is more obvious in Figure 4, where MoQ and FMLP show their superiority in peak prediction of mobile traffic; in this case, the predicted peak is very close to the ground-truth and none of benchmarks is capable of capturing the usage pattern related to the peak in an effective way.

6.2. Analysis of the Characteristics of Mobile Traffic Predictors

6.2.1. MoQ

Besides the advantage of achieving better peak prediction, MoQ also has good interpretability. As mentioned in previous sections, MoQ relies on the cooperation mechanism of experts, where the prediction benefits from switching among predictors with various forecasting styles. To study the contribution of each expert, in Figure 5 we visualize the predictions and the corresponding softmax scores of the four different experts: expert $E_{0.5}$, expert $E_{0.7}$, expert $E_{0.8}$ and expert $E_{0.9}$. Among these experts, expert

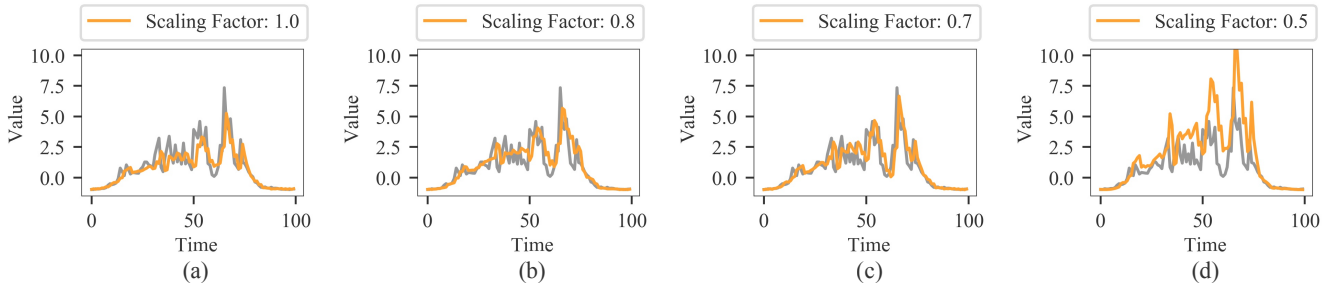

Figure 6: Peak predictions of cell B made by FMLP with respect to different setting of scaling factor, the gray line represents the ground-truth. (a) scaling factor = 1.0; (b) scaling factor = 0.8; (c) scaling factor = 0.7; (d) scaling factor = 0.5.

Metrics	Scaling Factor=1.0	Scaling Factor=0.8	Scaling Factor=0.7	Scaling Factor=0.6	Scaling Factor=0.5
MAE	**0.289**	0.296	0.301	0.381	0.403
MSE	0.309	**0.294**	0.312	0.528	0.608
Accuracy	63.2%	73.3%	75.5%	**84.4%**	83.9%
Sensitivity	27.2%	48.6%	53.5%	**74.6%**	74.0%

Table 2: Performance comparison of FMLPs trained with different scaling factors.

$E_{0.5}$ and expert $E_{0.7}$ are the experts making relatively conservative predictions. In Figure 5(a) we observe that the conservative expert dominates the final prediction for the non-peak part of time series, whereas the prediction style becomes aggressive once our model detects strong temporal dynamics from recent observations. If we focus on the strong oscillating area of this sequence, frequent switching between expert $E_{0.5}$ and expert $E_{0.8}$ can be seen. If we compare Figure 5(a) and Figure 5(b), the learned cooperation pattern is not the same by applying different penalization methods during training, but both of the methods result in similar results; this provides additional flexibility for adjusting the forecasting strategy. From this point of view, MoQ shows great interpretability, and the analysis of the behavior of experts can be used to fine tune the architecture and adapt it to different telecommunication applications.

6.2.2. FMLP

as we discussed in the previous section, the behavior of FMLP can be adjusted by changing the value of scaling factor, which provides us the flexibility of customizing the aggressiveness of forecasting based on the needs of applications; here we discuss the impact of changing scaling factor to the peak predictions. In Figure 3 and Figure 4, we visualize the predictions of another cell made by FMLP; four scaling factors are used to train FMLP: 1.0, 0.8, 0.7 and 0.5. When the scaling factor is set to 1.0, the difference between FMLP and the conventional MLP is minor; hence the predictions in Fig. 6(a) are equivalent to the predictions made by MLP. If we decrease the scaling factor from 1.0 to lower values, we can observe that the predictions of the peak parts have higher magnitude, and that makes the predictions more similar to the real cases. Every time we decrease the scaling factor, what we do is to create a smooth transition from conservative peak predictions to aggressive peak predictions, and this process is very flexible which can be controlled based on the operational strategy of network operators. Table 2 shows the performance of different FMLPs; generally, a lower scaling factor leads to better peak predictions and a slightly higher average loss.

6.3. Predictive Detection of Potential Traffic Unbalance

6.3.1. Mobile Traffic Analysis

In our approach, the predictive detection algorithm is applied to identify the potential traffic unbalance in a reference cell once we have the predictions. To investigate if there is a benefit of using the predictive approach, we create clusters and evaluate the performance of different approaches on them. We consider two types of cells, i.e. 800MHz and 1800MHz cells; the cluster is composed of cells in the same band, following the procedure discussed in Section 5. In the analysis, we use three-months observations of 2713 cells (1059 800MHz cells and 1654 1800MHz cells) and identify 417 reference cells which have high downlink usage.

Table 3: Overview of clusters: congested and non-congested.

	Congested	Non-Congested	Total
800MHz Cells	120 (11.3%)	939 (88.7%)	1059
1800MHz Cells	231 (14.0%)	1423 (86.0%)	1654

According to Table 3, at least 11.3% of 800MHz cells and 14.0% of 1800MHz cells are affected by traffic congestion events after evaluating the downlink usage and the maximum ratio, indicating that traffic redistribution would be very useful in a real-world scenario. To better understand the traffic unbalance issue, we study the behavior

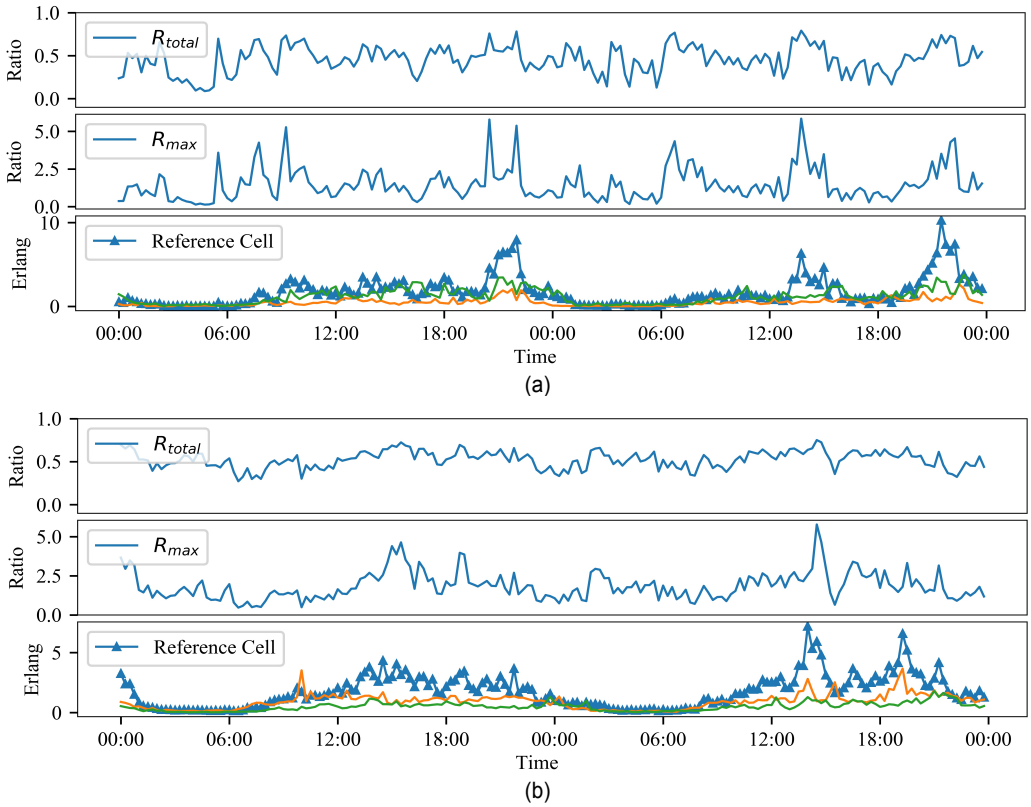

Figure 7: Traffic distribution within the cluster in two consecutive days. The marked line is the downlink usage of the reference cell and the unmarked lines are the downlink usage of its adjacent cells within the cluster. (a) cluster A (800MHz); (b) cluster B (1800MHz).

of the clusters in detail. Figure 7 illustrates the metrics of two clusters in two consecutive days; as we can observe, the downlink usage of the reference cell is much higher than the downlink usage of its adjacent cells in some time slots. In Figure 7(a), congestion can be observed around 9 PM; the congestion pattern is not always periodic as the congestion event is observed in the afternoon without having been observed in the previous day. According to the figure, it is obvious that the mobile traffic distributes in a relatively balanced way for most of the times. However, for certain time slots the reference cell carries much more traffic than the others which would potentially affect the QoS of the mobile users assigned to it. In this case, the prediction of the traffic unbalance would help the mobile traffic management.

6.3.2. Detection of Traffic Unbalance

To compare the performance of different approaches, we set the various prediction models to perform single-model and multi-model approach. For the single-model predictive approach, we use five different deep learning models to predict the downlink usage of the future steps, including MLP, GRU, LSTNet, Transformer, MoQ and FMLP (with two different scaling factors); for the multi-model approach, we use different combinations of models including LSTNet+GRU, MoQ+GRU, MoQ+LSTNet, GRU+MoQ and FMLP+GRU. The nomenclature for the multi-model approach is of the kind "$\text{Model}_{\text{ref}} + \text{Model}_{\text{adj}}$", for example MoQ+GRU means using MoQ for the reference cell and GRU for the adjacent cells. By using different predictive approaches, it is feasible to evaluate how peak forecasting performance affects the prediction performance of mobile traffic unbalance. To evaluate the classification performance, four metrics are calculated:

- *Balanced Accuracy*: the classification accuracy whose value equals to the mean value of the classification accuracy of each class.
- *Accuracy (Non-Congested)*: the classification accuracy of the non-congested class.
- *Accuracy (Congested)*: the classification accuracy of the congested class.
- *F-score*: this is the harmonic mean of precision and recall which evaluates the overall performance of a classifier. Its value ranges between 0 and 1, where 1 indicates a perfect classifier.

In the experiments, 20 clusters (10 at 800MHz and 10 at 1800MHz) are employed to compare the performance of different approaches. For each cell within a cluster, we first normalize its observations of three months and split them into many sub-series; each sub-series contains 1344 records

		Balanced Accuracy	Accuracy (Non-Congested)	Accuracy (Congested)	F-score
	Naive Approach	85.1%	91.9%	78.3%	0.787
	Predictive Approach				
Single-M	MLP-based	86.9%	92.4%	81.5%	0.810
	GRU-based	86.7%	92.4%	81.1%	0.808
	Transformer-based	85.1%	92.2%	78.2%	0.788
	LSTNet-based	87.3%	92.1%	82.4%	0.813
	MoQ-based	87.4%	91.4%	83.3%	0.811
	FMLP-based (scaling factor=0.6)	86.5%	90.5%	82.5%	0.797
	FMLP-based (scaling factor=0.7)	87.1%	91.9%	82.2%	0.809
Multi-M	LSTNet + GRU	87.5%	91.5%	83.5%	**0.814**
	MoQ + LSTNet	88.1%	90.1%	86.0%	0.813
	MoQ + GRU	**88.2%**	89.4%	87.0%	0.811
	GRU + MoQ	84.9%	**93.8%**	76.0%	0.791
	FMLP (scaling factor=0.6) + GRU	88.0%	87.4%	**88.6%**	0.801
	FMLP (scaling factor=0.7) + GRU	87.7%	90.8%	84.7%	0.812

Table 4: Comparison of different approaches; Single-M and Multi-M are the abbreviations for single-model approach and multi-model approach.

	Balanced Accuracy	Accuracy (Non-Congested)	Accuracy (Congested)	F-score
Naive Approach (downlink usage)	**85.1%**	**91.9%**	**78.3%**	**0.787**
Naive Approach (connected users)	67.9%	69.1%	66.6%	0.540
MLP-based (downlink usage)	**86.9%**	**92.4%**	**81.5%**	**0.810**
MLP-based (connected users)	67.6%	68.5%	67.4%	0.541

Table 5: Comparison of different features.

corresponding to the observations of two weeks. For each sequence, the forecasting model first predicts the normalized value of the downlink usage for the next two steps, then the real value (in Erlang) of the predicted downlink usage is obtained by denormalizing the predictions using the corresponding mean and standard deviation, and it is used by the predictive approach for decision making; the ground-truth of the predictive classification is generated using the future downlink usage (in Erlang) of the reference cell and the adjacent cells, and the details can be found in Section 5.1. The results are shown in Table 4.

In Table 4, the best two models are the multi-model approach MoQ+GRU and FMLP (scaling factor=0.6)+GRU as they have much higher accuracy of detecting the congestion events while obtaining good overall performance. For instance, compared to the naive approach, MoQ+GRU achieves 3.1% higher average classification accuracy and 8.7% higher accuracy of classifying the potential congestion; even though the use of prediction models introduces additional complexity (Table 1), the cost is still acceptable for mobile operators considering the improvement of congestion detection is significant. From the global point of view, predictive approaches are better than the naive approach especially for detecting the future mobile traffic unbalance. Among the single-model approaches, the MoQ-based approach is the best one as we care more about the sensitivity to the potential congestion, and its performance is slightly better than the LSTNet-based approach. The performance of the single model FMLP-based approach is similar to LSTNet-based and MoQ-based approaches when the scaling factor is set to 0.7; when we apply multi-model approach with FMLP, the performance of FMLP+GRU is better if the scaling factor equals to 0.6. As we discussed, the ability of MoQ and FMLP of predicting peaks makes them very good candidates for predicting the rapid increment of mobile demand. However, using only the single model could be sub-optimal in the scenario of traffic redistribution. Due to the fact that MoQ and FMLP are more capable of making aggressive predictions, they tend to make higher predictions for both the reference cell and the adjacent cells; because the maximum ratio measures the relatively difference between the downlink usage of the reference cell and its adjacent cells, in this case we can purposely obtain a higher ratio by using a more conservative predictor to predict the adjacent cells, which makes the classifier more sensitive to the future load unbalance. As the result, the approaches MoQ+GRU and FMLP+GRU show great performance in detecting the traffic unbalance. Besides improving the ability to detect traffic unbalance, we can also make the classifier focus more on the balanced case. The approach GRU+MoQ uses a conservative predictor for the reference cell and a more aggressive predictor for the adjacent cells, which leads to the highest classification accuracy on the non-congested class (93.8%), but with much worse performance on other metrics. Compared to the naive approach and the single-model approach, the

multi-model approach shows better performance and provides higher flexibility, as it can be adjusted based on the needs of different operating strategies of network operator. Considering the results of both single-model solutions and multi-model solutions, MoQ is seen as a better choice compared to FMLP; even though FMLP can achieve slightly better results in multi-model case (FMLP 0.6+GRU), the low scaling factor would lead to much worse general forecasting performance which is not desired (Table 2). In this case, FMLP is a good candidate designed for applications which have higher requirements of computation speed and model complexity.

Besides the comparison of algorithms, our attention should also be focused on the choice of the network KPIs used to perform the decision making. Among the network KPIs, the average number of connected users is another important variable which reflects the mobile demand indirectly. Compared to the downlink usage, this variable exhibits smaller variation over time which makes it much easier to be predicted, and it is interesting to study if the predictive approach performs better using as input the number of connected users with respect to the downlink usage. Table 5 compares the performance of the approaches using different variables. In the table, it is obvious that the average number of connected users is not a good choice for guiding the decision of traffic redistribution, as it cannot measure the mobile traffic correctly. The reason behind the result is simple: the mobile traffic is not always proportional to the number of connected users by considering some users would not generate too much traffic; in this case, even though a cell has a high number of connected users, there is no guarantee that the cell also has a high downlink demand. From this point of view, using the downlink usage would be the better choice.

7. Conclusion and Future Works

In this work we have proposed a prediction-based approach to detect potential mobile traffic unbalance among cells. The proposed approach is composed of a mobile traffic predictor selecting from MoQ and FMLP, and a traffic unbalance detection algorithm. Compared to other forecasting models, MoQ and FMLP provide much more accurate peak prediction of mobile traffic. MoQ achieves great forecasting performance along with excellent interpretability. As MoQ is a "heavy" model, we also propose a lightweight FMLP model. For the traffic unbalance detection, we propose two methods including single-model predictive approach and multi-model predictive approach. We first perform a large scale mobile traffic analysis by using more than 2700 cells deployed in northern Italy; extensive experiments are conducted on real-world dataset to evaluate the performance of the proposed detection approach, and the results show the superiority of our approach in traffic unbalance detection.

For future work, it would be interesting to integrate the proposed approach with mobile network optimization approaches such as MLB and CCO.

Acknowledgement

This work is supported by the PhD research program of TIM S.p.A (Italy).

References

[1] Ericsson, Ericsson annual report 2022 (2023).

[2] A. Awada, B. Wegmann, I. Viering, A. Klein, A joint optimization of antenna parameters in a cellular network using taguchi's method, in: 2011 IEEE 73rd Vehicular Technology Conference (VTC Spring), IEEE, 2011, pp. 1–5.

[3] A. Awada, B. Wegmann, I. Viering, A. Klein, A mathematical model for user traffic in coverage and capacity optimization of a cellular network, in: 2011 IEEE 73rd Vehicular Technology Conference (VTC Spring), IEEE, 2011, pp. 1–5.

[4] K. Sultan, H. Ali, Z. Zhang, Call detail records driven anomaly detection and traffic prediction in mobile cellular networks, IEEE Access 6 (2018) 41728–41737. doi:10.1109/ACCESS.2018.2859756.

[5] D. Kim, B. Shin, D. Hong, J. Lim, Self-configuration of neighbor cell list utilizing e-utran nodeb scanning in lte systems, in: 2010 7th IEEE Consumer Communications and Networking Conference, IEEE, 2010, pp. 1–5.

[6] H. Klessig, A. Fehske, G. Fettweis, J. Voigt, Improving coverage and load conditions through joint adaptation of antenna tilts and cell selection rules in mobile networks, in: 2012 International Symposium on Wireless Communication Systems (ISWCS), IEEE, 2012, pp. 21–25.

[7] R. Kwan, R. Arnott, R. Paterson, R. Trivisonno, M. Kubota, On mobility load balancing for lte systems, in: 2010 IEEE 72nd vehicular technology conference-fall, IEEE, 2010, pp. 1–5.

[8] S. Fan, H. Tian, C. Sengul, Self-optimization of coverage and capacity based on a fuzzy neural network with cooperative reinforcement learning, EURASIP Journal on Wireless Communications and Networking 2014 (1) (2014) 1–14.

[9] V. Buenestado, M. Toril, S. Luna-Ramírez, J. M. Ruiz-Avilés, A. Mendo, Self-tuning of remote electrical tilts based on call traces for coverage and capacity optimization in lte, IEEE Transactions on Vehicular Technology 66 (5) (2016) 4315–4326.

[10] Y. Yang, P. Li, X. Chen, W. Wang, A high-efficient algorithm of mobile load balancing in lte system, in: 2012 IEEE Vehicular Technology Conference (VTC Fall), IEEE, 2012, pp. 1–5.

[11] S. Li, E. Magli, G. Francini, To be conservative or to be aggressive? a risk-adaptive mixture of experts for mobile traffic forecasting, in: 2023 IEEE International Conference on Communications (ICC), IEEE, 2023.

[12] N. Dandanov, H. Al-Shatri, A. Klein, V. Poulkov, Dynamic self-optimization of the antenna tilt for best trade-off between coverage and capacity in mobile networks, Wireless Personal Communications 92 (1) (2017) 251–278.

[13] R. Razavi, S. Klein, H. Claussen, Self-optimization of capacity and coverage in lte networks using a fuzzy reinforcement learning approach, in: 21st Annual IEEE International Symposium on Personal, Indoor and Mobile Radio Communications, IEEE, 2010, pp. 1865–1870.

[14] T. Cai, G. P. Koudouridis, C. Qvarfordt, J. Johansson, P. Legg, Coverage and capacity optimization in e-utran based on central coordination and distributed gibbs sampling, in: 2010 IEEE 71st Vehicular Technology Conference, IEEE, 2010, pp. 1–5.

[15] N. Dandanov, S. R. Samal, S. Bandopadhaya, V. Poulkov, K. Tonchev, P. Koleva, Comparison of wireless channels for antenna tilt based coverage and capacity optimization, in: 2018 Global Wireless Summit (GWS), IEEE, 2018, pp. 119–123.

[16] C. Yanyun, H. Alexis, X. Hui, Y. Xingxiu, Coverage and capacity optimization for 4g lte networks using differential evolution, in: 2018 5th IEEE International Conference on Cloud Computing and Intelligence Systems (CCIS), IEEE, 2018, pp. 640–645.

[17] R. Kwan, R. Arnott, R. Paterson, R. Trivisonno, M. Kubota, On mobility load balancing for lte systems, in: 2010 IEEE 72nd Vehicular Technology Conference - Fall, 2010, pp. 1–5. doi:10.1109/VETECF.2010.5594565.

[18] R. Nasri, Z. Altman, Handover adaptation for dynamic load balancing in 3gpp long term evolution systems, arXiv preprint arXiv:1307.1212 (2013).

[19] N. Zhang, S. Zhang, S. Wu, J. Ren, J. W. Mark, X. Shen, Beyond coexistence: Traffic steering in lte networks with unlicensed bands, IEEE wireless communications 23 (6) (2016) 40–46.

[20] M. Mudelsee, Trend analysis of climate time series: A review of methods, Earth-science reviews 190 (2019) 310–322.

[21] A. L. Loureiro, V. L. Miguéis, L. F. da Silva, Exploring the use of deep neural networks for sales forecasting in fashion retail, Decision Support Systems 114 (2018) 81–93.

[22] R. Yu, Y. Li, C. Shahabi, U. Demiryurek, Y. Liu, Deep learning: A generic approach for extreme condition traffic forecasting, in: Proceedings of the 2017 SIAM international Conference on Data Mining, SIAM, 2017, pp. 777–785.

[23] G. E. Box, G. M. Jenkins, G. C. Reinsel, G. M. Ljung, Time series analysis: forecasting and control, John Wiley & Sons, 2015.

[24] P. Lara-Benítez, M. Carranza-García, J. C. Riquelme, An experimental review on deep learning architectures for time series forecasting, International Journal of Neural Systems 31 (03) (2021) 2130001.

[25] C. Hamzaçebi, D. Akay, F. Kutay, Comparison of direct and iterative artificial neural network forecast approaches in multiperiodic time series forecasting, Expert systems with applications 36 (2) (2009) 3839–3844.

[26] P.-H. Kuo, C.-J. Huang, A high precision artificial neural networks model for short-term energy load forecasting, Energies 11 (1) (2018) 213.

[27] I. Koprinska, D. Wu, Z. Wang, Convolutional neural networks for energy time series forecasting, in: 2018 international joint conference on neural networks (IJCNN), IEEE, 2018, pp. 1–8.

[28] A. Tsantekidis, N. Passalis, A. Tefas, J. Kanniainen, M. Gabbouj, A. Iosifidis, Forecasting stock prices from the limit order book using convolutional neural networks, in: 2017 IEEE 19th Conference on Business Informatics (CBI), Vol. 01, 2017, pp. 7–12. doi:10.1109/CBI.2017.23.

[29] L. Kuan, Z. Yan, W. Xin, C. Yan, P. Xiangkun, S. Wenxue, J. Zhe, Z. Yong, X. Nan, Z. Xin, Short-term electricity load forecasting method based on multilayered self-normalizing gru network, in: 2017 IEEE Conference on Energy Internet and Energy System Integration (EI2), IEEE, 2017, pp. 1–5.

[30] N. Wu, B. Green, X. Ben, S. O'Banion, Deep transformer models for time series forecasting: The influenza prevalence case, ArXiv abs/2001.08317 (2020).

[31] S. Bai, J. Z. Kolter, V. Koltun, An empirical evaluation of generic convolutional and recurrent networks for sequence modeling, arXiv preprint arXiv:1803.01271 (2018).

[32] A. Zeng, M. Chen, L. Zhang, Q. Xu, Are transformers effective for time series forecasting? arxiv 2022, arXiv preprint arXiv:2205.13504.

[33] H. Zhou, S. Zhang, J. Peng, S. Zhang, J. Li, H. Xiong, W. Zhang, Informer: Beyond efficient transformer for long sequence time-series forecasting, in: Proceedings of AAAI, 2021.

[34] H. Wu, J. Xu, J. Wang, M. Long, Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting, Advances in Neural Information Processing Systems 34 (2021) 22419–22430.

[35] R. A. Jacobs, M. I. Jordan, S. J. Nowlan, G. E. Hinton, Adaptive mixtures of local experts, Neural computation 3 (1) (1991) 79–87.

[36] R. Collobert, S. Bengio, Y. Bengio, A parallel mixture of svms for very large scale problems, in: T. Dietterich, S. Becker, Z. Ghahramani (Eds.), Advances in Neural Information Processing Systems, Vol. 14, MIT Press, 2002.

[37] M. I. Jordan, R. A. Jacobs, Hierarchical mixtures of experts and the em algorithm, Neural computation 6 (2) (1994) 181–214.

[38] S. E. Chazan, J. Goldberger, S. Gannot, Speech enhancement using a deep mixture of experts, arXiv preprint arXiv:1703.09302 (2017).

[39] N. Shazeer, A. Mirhoseini, K. Maziarz, A. Davis, Q. Le, G. Hinton, J. Dean, Outrageously large neural networks: The sparsely-gated mixture-of-experts layer, arXiv preprint arXiv:1701.06538 (2017).

[40] Z. Ge, C. McCool, C. Sanderson, P. Corke, Subset feature learning for fine-grained category classification, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, 2015, pp. 46–52.

[41] R. T. Mullapudi, W. R. Mark, N. Shazeer, K. Fatahalian, Hydranets: Specialized dynamic architectures for efficient inference, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 8080–8089.

[42] D. Eigen, M. Ranzato, I. Sutskever, Learning factored representations in a deep mixture of experts, arXiv preprint arXiv:1312.4314 (2013).

[43] R. Koenker, G. Bassett Jr, Regression quantiles, Econometrica: journal of the Econometric Society (1978) 33–50.

[44] J. W. Taylor, A quantile regression neural network approach to estimating the conditional density of multiperiod returns, Journal of Forecasting 19 (4) (2000) 299–311.

[45] W. Zhang, H. Quan, D. Srinivasan, Parallel and reliable probabilistic load forecasting via quantile regression forest and quantile determination, Energy 160 (2018) 810–819.

[46] F. Rodrigues, F. C. Pereira, Beyond expectation: Deep joint mean and quantile regression for spatiotemporal problems, IEEE transactions on neural networks and learning systems 31 (12) (2020) 5377–5389.

[47] G. Lai, W.-C. Chang, Y. Yang, H. Liu, Modeling long-and short-term temporal patterns with deep neural networks, in: The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval, 2018, pp. 95–104.

[48] I. Goodfellow, Y. Bengio, A. Courville, Deep learning, MIT press, 2016.

[49] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural computation 9 (8) (1997) 1735–1780.

[50] J. Chung, C. Gulcehre, K. Cho, Y. Bengio, Empirical evaluation of gated recurrent neural networks on sequence modeling, arXiv preprint arXiv:1412.3555 (2014).

[51] R. Wen, K. Torkkola, B. Narayanaswamy, D. Madeka, A multi-horizon quantile recurrent forecaster, arXiv preprint arXiv:1711.11053 (2017).

[52] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, I. Polosukhin, Attention is all you need, in: Advances in neural information processing systems, 2017, pp. 5998–6008.