To Be Conservative or To Be Aggressive? A Risk-Adaptive Mixture of Experts for Mobile Traffic Forecasting

To Be Conservative or To Be Aggressive? A Risk-Adaptive Mixture of Experts for Mobile Traffic Forecasting

Shuyang Li*, Enrico Magli*, Gianluca Francini†
*Department of Electronics and Telecommunications Engineering, Politecnico di Torino, Torino, Italy
{shuyang.li, enrico.magli}@polito.it
†Telecom Italia, Torino, Italy
gianluca.francini@telecomitalia.it

***Abstract*—Mobile traffic forecasting plays a key role for optimizing the configuration of network cells. Network operators are very interested in predicting upcoming mobile traffic peaks in an accurate way, in order to improve quality of service via efficient resource allocation. However, forecasting potential peaks is very challenging considering that many peaks occur suddenly for no apparent reason; hence, it is very difficult to determine if a peak will occur in near future based on the temporal dynamics of mobile traffic, potentially leading to inaccurate predictions. To improve the performance of peak prediction, we propose a novel deep learning model called Mixture of Quantiles (MoQ). MoQ employs a mixture of experts model featuring a manager to fuse the predictions of multiple experts. In order to overcome the problem of overly smooth predictions on peaks, the experts are designed to have differentiated forecasting styles from conservative to aggressive. A cooperation mechanism is established through a carefully designed training process, whereby conservative experts are responsible for the forecasting of the off-peak region, and the employed experts are switched to the aggressive ones once the potential increasing trend is detected by manager, which leads to significantly improved peak predictions. Extensive experiments on real-world dataset showcase the effectiveness of the proposed MoQ model, which outperforms all the benchmarks and shows its superior performance in peak forecasting along with excellent interpretability.**

***Index Terms*—mobile traffic forecasting, peak prediction, deep learning**

I. Introduction

According to Ericsson's 2021 annual report, the number of mobile subscriptions will grow from around 8.3 billion by the end of 2022 to around 8.8 billion by the end of 2026 [1]; the growing traffic makes it difficult for Internet Service Providers (ISP) to satisfy stringent quality of service requirements. To maintain the efficiency of mobile networks, the network configuration has to be updated frequently to adapt to the latest trend of mobile demand. However, there is always a delay at observing the latest network key performance indicators (KPI). If the operator employs delayed observations to reconfigure the network, there is a risk that the updated configuration is already outdated and cannot adapt well to the actual traffic pattern; this problem becomes more critical in the presence of traffic peaks, which are critically important towards quality of service.

Even though the prediction of traffic peaks is very important, most of the related research focuses on improving the overall prediction performance instead of the peak forecasting performance [2]–[4]. This latter requires a forecasting model that can capture complex temporal dynamics; deep learning approaches are good candidates as they have proved to be very effective at learning complex non-linear patterns, and many deep learning models have been used to predict time series in recent years [5]. On the other hand, while these models are indeed powerful, they are unable to trade off between the average accuracy and the accuracy of peak prediction. Mobile traffic time series typically exhibit a generally stationary behavior interrupted by sudden strong peaks. Many time series forecasting methods are very good at predicting the stationary parts, but provide inaccurate results in the prediction of peaks. 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 prediction tends 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 overall loss and peak prediction, which has to be handled according to application requirements. Generally for ISPs, it is acceptable that the model makes better peak predictions even though the errors on the non-peak parts are slightly higher, as the peaks are the most valuable parts of mobile traffic.

For tackling these challenges, we propose a novel deep learning model called Mixture of Quantiles (MoQ). MoQ is a point estimate model based on the Mixture of Experts (MoE) framework, which supports 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. Specifically, the main contributions of our work are summarized as follows:

- We propose a novel model called MoQ, which supports various forecasting styles, and features a flexible blending of conservative and aggressive predictions based on recent observations.
- We propose a two-stage training process to address the difficulty of training MoQ model. Indeed, in MoQ each expert must behave in a specific way. To this end, we first pre-train experts with different objective functions to promote their diversity. In the second stage, penalization is applied during training to prevent the problem of imbalanced assignment of experts and encourage the cooperation among experts.
- Experiments are carried out on real-world mobile network dataset, and the results prove that MoQ improves traffic peak prediction significantly. Comparing with baselines, our model is more sensitive to the occurrence of peaks. Visualizing the assigned weights of experts, we observe interpretable cooperation between them, which explains why the model is very effective at adapting to fast changing time series.

II. Problem Formulation

Quantiles are important for describing the characteristics of a distribution. 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 has led to the use of quantile regression. 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. 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.

In this work, our goal is to perform time series forecasting based on historical observations of mobile traffic. Assuming we want to predict a group of mobile traffic time series, each time series is represented by a matrix $\mathbf{x}_{1:t} = [x_1, x_2, ..., x_t] \in R^{t \times f}$ which consists of historical observations of vectors x_i containing f features, where t is the length of time series; a given set of time series is denoted as $X_{1:t} \in R^{N \times t \times f}$, where N is the size of the group. If the objective is to forecast the future values of these features at the next h steps, the problem can be formulated as

$$\hat{X}_{t+1:t+h} = f(X_{1:t}), \quad (2)$$

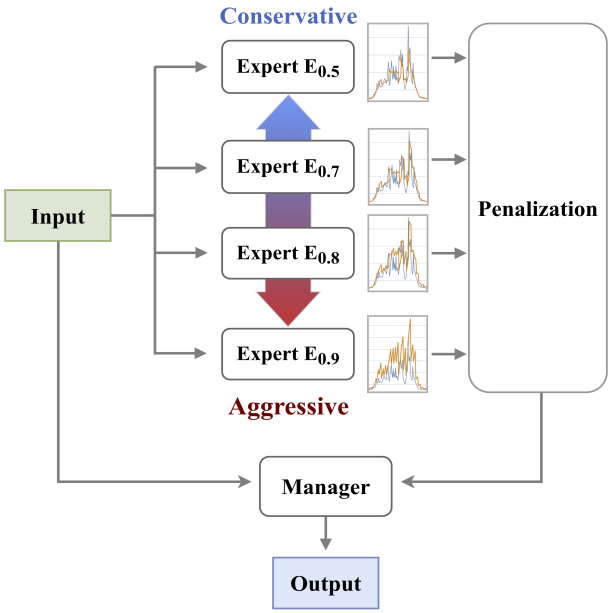

Fig. 1: Architecture of Mixture of Quantiles: expert E_q is trained to minimize a quantile loss having quantile index equals to q. The orange line represents forecasting and the blue line represents ground-truth.

where $\hat{X}_{t+1:t+h} \in R^{N \times h \times f}$ is a matrix of predicted features from time $t+1$ up to time $t+h$, and $f(\cdot)$ is a suitable neural network.

III. Methodology

A. 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. How to use these predictions to maximum effect is the job of manager, who observes the recent temporal behavior of the input features 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.

B. Experts with Various Forecasting Styles

Experts are the key components in MoQ, as their behavior strongly affects the model performance. In this work, LSTNet

[6] is used as the backbone network of each expert. For the MoE framework, the experts are expected to learn different patterns from the same input. If the characteristics of experts differ a lot, each expert specializes in extracting different knowledge patterns, which provides benefit when the manager aggregates these pieces of information. However, the process raises several issues. The first issue is related to the diversity of experts, as there is no guarantee that each expert will learn different things from the same input. If experts act in a very similar way, there is no benefit in using an MoE model. Another issue is about expert assignment; indeed, if we train the MoE model end-to-end from scratch, it is highly likely that the manager will overuse one specific expert. The typical reason is that an expert initially shows superiority because of its weight initialization or the preceding weight update, leading the manager to assign a higher weight to that expert; thus, the expert becomes more reputable due to a more frequent assignment, which eventually results in imbalanced training. For the MoQ, each expert is expected to have different forecasting styles, and this is extremely difficult to achieve if we train experts with the same objective function. To promote diversity among the experts, a specific quantile loss is employed to pretrain each expert individually, and the quantile index is different for the training of each expert. As depicted in Figure 1, MoQ consists of four experts, and these experts are pretrained with the default quantile index 0.5, 0.7, 0.8 and 0.9 respectively; it is also feasible to customize the forecasting behavior by pruning specific experts or changing the default quantile index. 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:

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

where $\hat{X}^{E}_{t+1:t+h} \in R^{k\times N\times h\times f}$. τ is the quantile index, N is the size of mini-batch, h is the forecasting horizon, f is the number of features and k is the number of experts. The next question is how to fuse such predictions in the most effective fashion.

C. Manager

The manager is a gating function which 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:

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

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

where FC_w is a fully-connected layer parameterized by the weights/biases w, $X_{-p:t} \in R^{N\times p\times f}$ 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 $\hat{X}_{t+1:t+h}$ is made by fusing the weighted output of experts:

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

where E is the set of experts, $\hat{X}^e_{t+1:t+h}$ and S^e are the corresponding prediction and score for expert e. To acquire the ability of using experts cooperatively, manager is trained to minimize the Mean Absolute Error (MAE) between the fused prediction and the ground-truth; the details are discussed in the next section.

D. Two-Stage Training and Penalization

Training is the most important part for MoE model; as mentioned previously, the difficulties mainly arise from two problems: (1) how to guarantee the diversity of experts; (2) how to obtain reasonable and interpretable expert assignment without overusing a specific expert. To tackle these issues, a simple single-stage training is insufficient, and in this work we propose to train MoQ in two different stages in order to make the training process successful.

The first training stage focuses on the experts pretraining, where each expert is trained individually with a specific quantile loss in order to promote diversity. Due to the different settings of quantile loss, the experts will not end up being very similar. After pretraining, the weights of experts are frozen and will not be updated during the second stage, so as to avoid undesired behavioral changes and to simplify the training of the manager. The second stage aims to train the manager in order to develop its ability to select and blend experts. If we want to improve the prediction of highly dynamic regions while minimizing the overall loss, selecting proper experts upon detection of different situations is very important; however, this is also tricky for the manager to learn. In the ideal case, the manager should rely more on the conservative expert for the forecasting of non-dynamic regions, and on the aggressive experts when the model needs to predict peaks. However, it is rare to obtain a trained model really working in this way. In most cases, models tend to overuse the most conservative expert which is pretrained to predict the median value. Since peaks are important but relatively rare events in the dataset, the simplest choice for minimizing the overall loss is to avoid making aggressive predictions. In this way, the model loses the ability to capture peaks, which leads to large errors in the dynamic parts while the overall loss is rather low. This process is useless in real applications because it cannot estimate the most significant part of time series, as in the naive but apt metaphor: you will never make mistakes if you

decide to do nothing. In this case, a penalization mechanism is needed against the manager's nature of being conservative. Specifically, two different penalization methods are introduced in this paper.

1) Penalization Mask: The first option is to use a mask during training. This mask acts as a penalization term in the second training stage, with the objective to penalize conservative experts and to encourage 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 $\hat{X}^E_{t+1:t+h}$ (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 S^e 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. The penalization mask is an objective-oriented penalization method, and the mask values can be changed in purpose in order to guide the decision of manager in the desired way, thereby controlling the behavior of MoQ.

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) 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. The reason why we penalize the predictions is to obtain a trade-off between quality of peak prediction and overall loss, as we need to balance the behavior of experts while avoiding having high overall loss. With the penalization, we therefore lower the priority of using conservative predictions. Since the issue of overusing specific experts is addressed, the remaining problem is how to force manager to extract knowledge from the recent observations to guide its fusion. Indeed, we want MoQ to make aggressive predictions only when there could be an occurrence of a peak in the upcoming steps. If we apply penalization to expert predictions for all training examples in a mini-batch, this may lead to higher-than-expected predictions of non-peak parts considering that the conservative experts would be penalized all the time. To overcome this problem, we do not apply the penalization for all training samples, but only for the samples whose ground-truth is much higher than the others. Specifically, for each training sample we sum the values of its ground-truth, and only the samples whose sum is in the largest 10% will be penalized. The rationale is that the selected training samples have a higher possibility of having peaks in next few steps, and the manager is forced to learn how to extract local temporal dynamics from recent observations to determine if peak will occur or not. In this training stage, the model is trained to minimize MAE of final prediction. With the way of penalizing predictions for selected training samples, we introduce inductive bias in the training process, which is intended to inspire the manager to act in our desired way. Comparing to conventional models (e.g., LSTM), MoQ requires more time to train, but the general training time is acceptable as ISPs usually retrain a model every several weeks.

IV. Experiments

A. Dataset

We conduct experiments on a real-world mobile network dataset to evaluate the performance of the proposed model. The dataset is an industrial dataset provided by Telecom Italia S.p.A which is the largest Italian telecommunications services provider. Mobile traffic 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; our target is to predict the mobile traffic for the next hour. Notice that all time series of the 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.

B. 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 (MoQ$^\dagger_\alpha$). We compare the performance of our model against a set of baseline approaches, including: MLP [7], LSTM [8], GRU [9], LSTNet [6], TCN [10], MQ-RNN [11] and Transformer [12]. The baseline models cover the most popular approaches in the time series forecasting field. 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 two datasets employed in this paper, by merging train set and validation set, and eventually evaluated on the test set.

To evaluate the performance, four metrics are used. Initially, we quantify the overall performance of the models in terms of MAE and Mean Squared Error (MSE). Since MAE and MSE do not properly capture the ability of a model to predict

TABLE I: Performance comparison of different models for overall mobile traffic forecasting and peak classification.

Metrics	MLP	LSTM	GRU	LSTNet	TCN	MQ-RNN	Transformer	MoQ	$\text{MoQ}_2^\dagger$	$\text{MoQ}_4^\dagger$
MAE	0.304	0.295	0.292	**0.288**	0.395	0.299	0.325	0.312	0.302	0.319
MSE	0.289	0.296	0.290	**0.283**	0.448	0.305	0.326	0.337	0.295	0.322
Accuracy	68.3%	68.1%	67.9%	66.7%	59.5%	64.7%	67.1%	75.9%	74.3%	**76.4%**
Sensitivity	37.7%	37.1%	36.6%	34.1%	19.5%	30.1%	35.0%	54.5%	50.6%	**55.7%**

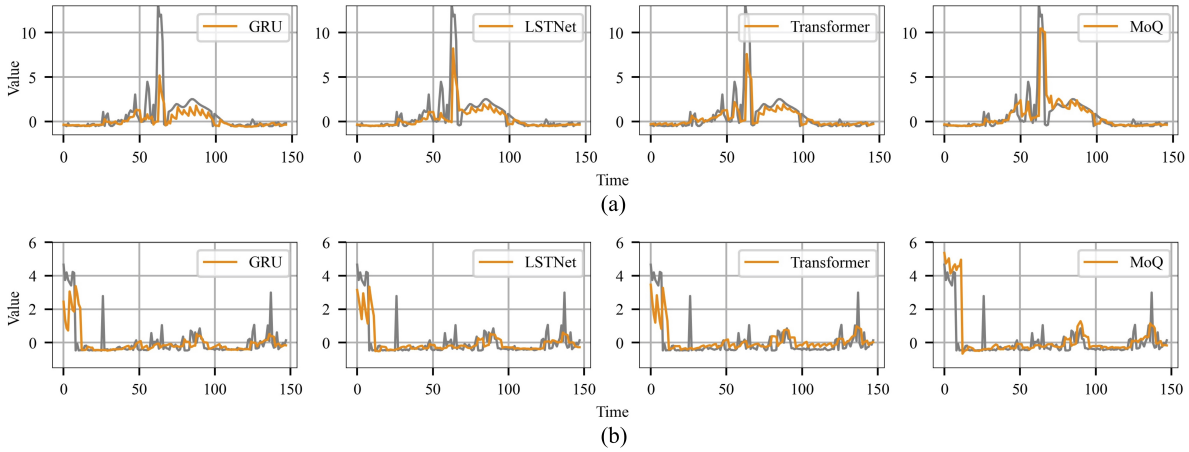

Fig. 2: Peak predictions of mobile traffic, the gray line represents the ground-truth and the orange line represents the forecasting. The penalization mask of MoQ is employed. (a) cell A; (b) cell B.

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 mobile traffic 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 mobile traffic four time-steps ahead is employed for binary peak classification, in that if the predicted mobile traffic 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 classification accuracy is also used to quantify the general performance of predictive classification.

C. Results

We evaluate the performance of these models on test set, and the results are reported in Table I. 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 TCN is the worst among the baselines, its overall performance and peak prediction are both unsatisfactory. 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, $\text{MoQ}_4^\dagger$ has the highest sensitivity whose value is 18% higher than the highest sensitivity of baselines (37.7%), which means this model is more capable of predicting potential peaks in upcoming steps; $\text{MoQ}_4^\dagger$ also has the highest classification accuracy, that means the proposed model is able to perform forecasting based the recent trend instead of overestimating the targets all the time. 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. To better understand the behavior of different architectures, we select several models and visualize their forecasting. Figure 2 illustrates peak predictions performed by different models, where Figure 2(a) and Figure 2(b) refer to the mobile traffic collected from two different cells of mobile network. In Figure 2(a), GRU makes very poor forecasting of the peak. Transformer and LSTNet can make better predictions comparing to GRU, but the predictions are still significantly underestimated. The proposed MoQ model

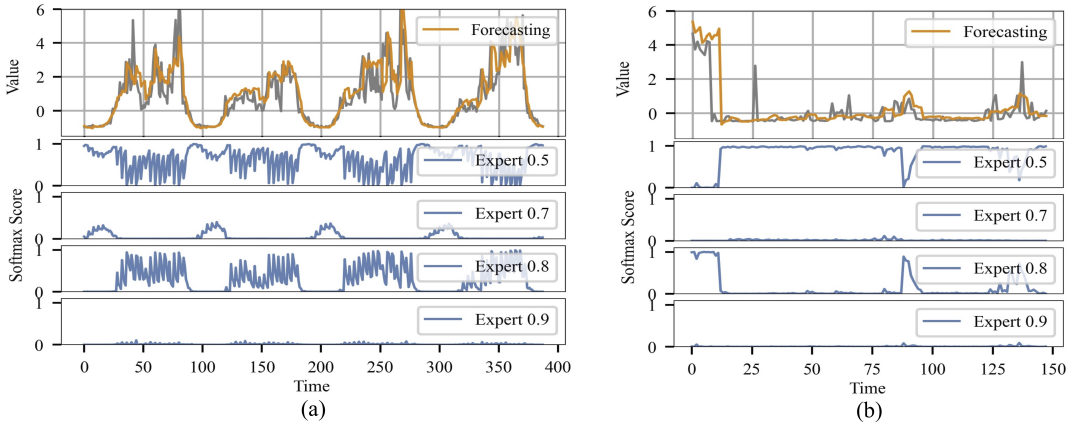

Fig. 3: Visualization of the cooperation between experts: the expert scores used to fuse the predictions are visualized for two different cells, the gray line represents the ground-truth and the orange line represents the forecasting.

can better estimate peaks in the time series, and the predicted peak is very close to the ground-truth. In Figure 2(b), none of benchmarks is capable of capturing the usage pattern related to the peak on the left; this leads to the risk of overload if the predictions are used to adaptively configure mobile networks, which could eventually lead to a cell shutdown. Benefiting from the aggressive prediction of MoQ, the risk of overload is decreased since we can predict peaks in a more reliable way.

Besides the advantage of making better peak prediction, our model 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 3 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 $E_{0.5}$ and expert $E_{0.7}$ are the experts which make relatively conservative predictions. In Figure 3(a) we observe that conservative experts dominate the final prediction for the non-peak part of time series, and the prediction style turns to be 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. This cooperation mechanism is more obvious in the series depicted in Figure 3(b), where expert $E_{0.5}$ is responsible for the forecasting of MoQ for most of the time, except upon the occurrence of several peaks. In this case, our model shows great interpretability, and the analysis of the behavior of experts can be used to fine tune the architecture and adapt it to different application fields.

V. Conclusion

In this work we have proposed MoQ, a novel MoE-style model for mobile traffic forecasting. The model is designed to address the problem of peak prediction in mobile traffic prediction. Through an ad-hoc training procedure, MoQ yields experts with various forecasting styles, where some experts make conservative predictions while the others make aggressive predictions, and a manager fuses the available predictions. The results show that the proposed model outperforms all baselines for the prediction of peaks; the results are also interpretable in terms of the individual contribution of each expert and their cooperation pattern.

Acknowledgement

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

References

[1] Ericsson, "Ericsson annual report 2021," 2022.

[2] W. Jiang, "Cellular traffic prediction with machine learning: A survey," *Expert Systems with Applications*, p. 117163, 2022.

[3] H. D. Trinh, L. Giupponi, and P. Dini, "Mobile traffic prediction from raw data using lstm networks," in *2018 IEEE 29th Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*. IEEE, 2018, pp. 1827–1832.

[4] C. Zhang and P. Patras, "Long-term mobile traffic forecasting using deep spatio-temporal neural networks," in *Proceedings of the Eighteenth ACM International Symposium on Mobile Ad Hoc Networking and Computing*, 2018, pp. 231–240.

[5] B. Lim and S. Zohren, "Time-series forecasting with deep learning: a survey," *Philosophical Transactions of the Royal Society A*, vol. 379, no. 2194, p. 20200209, 2021.

[6] G. Lai, W.-C. Chang, Y. Yang, and 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.

[7] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016.

[8] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[9] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014.

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

[11] R. Wen, K. Torkkola, B. Narayanaswamy, and D. Madeka, "A multi-horizon quantile recurrent forecaster," *arXiv preprint arXiv:1711.11053*, 2017.

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