Exploring Time Series Variability: A Training-Efficient Mobile Traffic Predictor

Exploring Time Series Variability: A Training-Efficient Mobile Traffic Predictor

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

***Abstract*—Short-term mobile traffic forecasting is an important topic for mobile network operators as accurate predictions are essential for enhancing network efficiency and saving resources. However, predicting mobile demand is difficult because it requires to distinguish different types of patterns, e.g. periodic and short-time behaviours. Recently, deep learning based approaches have been proven effective because of their ability to learn complex non-linear dependencies. Among different architectures, the models built on Recurrent Neural Network (RNN) show impressive performance in sequence modelling and they are widely used in mobile traffic forecasting; however, their rather long training time raises the issue of a high training cost for mobile operators. To better capture the latest trend of mobile demand, mobile operators retrain the forecasting model every week using the recent observations; the training is very expensive and consumes a considerable amount of energy considering that there are hundreds of thousands of cells managed by mobile operators. For mobile operators, it is always desirable to reduce the training cost while retaining the same level of forecasting performance. To address this challenge, in this paper we propose a pure FC-layer deep learning predictor allowing the mobile operators to obtain excellent forecasting performance along with very low training cost. The network is composed of three feature extraction layers, containing RNN-inspired building blocks learning features that are robust to unexpected local variations; these features are then used to make predictions in the last two layers. Extensive experiments are conduct on a real-world dataset, showing that our model obtains almost the same prediction performance as the state-of-the-art RNN-based baseline model, while requiring only 1% of its training time.**

***Index Terms*—mobile traffic prediction, wireless traffic analysis, low-cost-training**

I. Introduction

According to Ericsson annual report 2022, mobile data traffic volume is estimated to increase by more than twice in the period 2023–2027, and the mobile video traffic is forecasted to grow by almost 30% annually through 2027 [1]. To provide high-quality experience to mobile users, mobile operators are required to manage mobile networks in such a way as to meet specific QoS requirements; being able to make accurate short-term mobile traffic forecasts of network KPIs becomes more important than ever. Predicting mobile demand accurately is never an easy task due to the characteristics of mobile traffic over time: one can observe strong periodicity as the wireless demand closely follows human daily activities, whereas traffic is also affected by unexpected events (social events, etc.) adding variability, resulting in complex temporal dynamics. Based on this reason, traditional statistical approaches are not favoured because they are inefficient at modelling non-linear relationships. In recent years, deep learning approaches have shown their superiority in time series forecasting, which have also been used in the mobile traffic scenario [2].

Among deep learning architectures, RNN-based models are very well suited to time series forecasting; however, their training process is rather time-consuming. In RNNs, the calculation of the current hidden state is dependent on the preceding states, which means the process cannot be parallelized. Considering to apply RNN-based models to predict mobile traffic time series, this problem is not negligible since network operators typically retrain the predictor every week in order to capture the latest trend of traffic demand. Generally, a city-wide LTE network can easily cover thousands of cells, and the total amount of cells would go for hundreds of thousands considering all the areas managed by the mobile operators; when there are so many cells in the system, the training of forecasting model would be very expensive and consume a considerable amount of energy, and the situation is more critical when the forecasting model itself has a high training cost. As mobile operators are profit-driven, it is always desired to reduce the training cost without losing the accuracy of forecasting. To tackle this challenge, it is essential to design a model which obtains good prediction performance while enabling fast training. Among the existing forecasting models, N-BEATS [3] is a special one as it obtains good performance using only FC layers, which allows a fast-training; even though its performance is worse than the state-of-the-art RNN-based models, it proves that a model can obtain good forecasting performance using only FC layers along with residual connections. In this paper, inspired by the characteristics of RNN and the design of N-BEATS, we propose a deep learning predictor for mobile traffic forecasting called Variability-Enhanced Network (VEN) which does away with the recurrent model and is completely built on FC layers; to achieve fast training speed of the model, we purposely avoid using computationally expensive mechanisms such as multi-head attention and others. The proposed model has a similar design as N-BEATS but follows a different underlying mechanism: while N-BEATS performs ensemble-style forecasting, our model focuses on modelling the variability of time series. In particular, the proposed architecture allows the network to introduce a certain

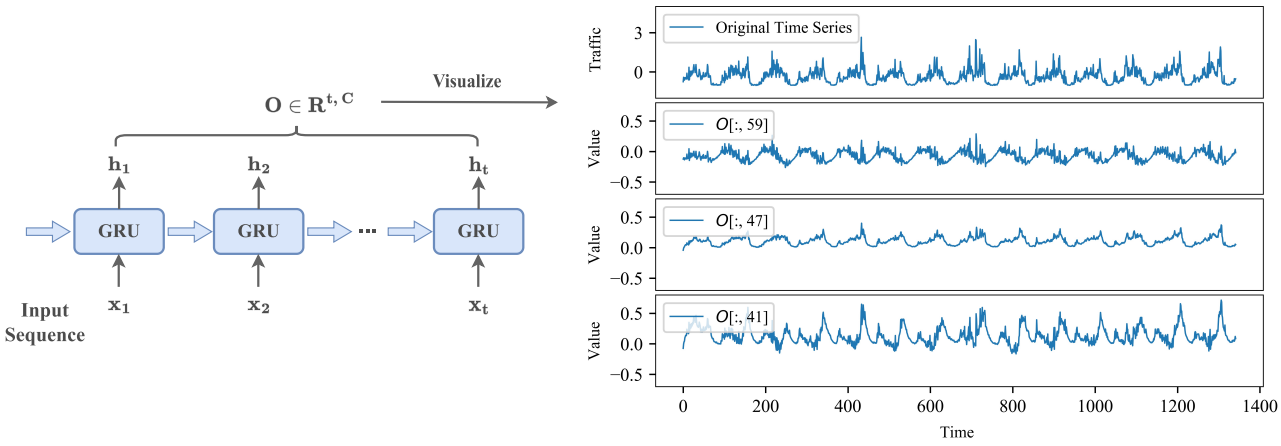

Fig. 1: Visualization of hidden vectors learned by GRU in time series forecasting task; O is the collection of hidden states whose shape is (t, C) where t is the number of input time steps and C is the number of channels of GRU.

level of variability in the time series, on three temporal levels: daily observations, weekly observations and recent observations; on each level, the model generates series with different degrees of variability allowing the model to better describe the temporal dynamics of the input; eventually, the values at the last time step of all generated series are concatenated and used to make predictions. In this way, the proposed model can make accurate mobile traffic predictions while saving a lot of training time. Specifically, the main contributions of our work are summarized as follows:

- We propose a forecasting model called VEN, which adds different degrees of variability to a time series, allowing the network to make improved mobile traffic predictions along with a very low training time.
- In order to reduce the training time as much as possible, the design of VEN uses only FC layers. This design makes the network able to obtain both excellent forecasting performance and very fast training speed.
- We have conducted extensive experiments using a real-world dataset provided the largest telecommunication services provider in Italy, and the proposed model is compared with twelve baseline methods which are popular in time series forecasting field. According to the results, its predictions have state-of-the-art accuracy, while reducing training time by approximately 99%.

II. Problem Formulation

In this work, our goal is to make short-term time series predictions of the downlink usage traffic using a deep learning model trained based on past observations. Assuming we want to predict the future values of downlink mobile traffic with w forecast horizons, a univariate 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:

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

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 the employed deep learning predictor.

III. Methodology

A. Learning from RNN

While RNN-based models are known to be efficient at time series forecasting, their specific learning mechanism is not completely understood yet; Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) are efficient at countering vanishing gradients, but the exact way they learn a model with temporal dependencies is unclear. To get an insight into RNNs characteristics, we have trained a GRU to predict the downlink usage two steps ahead, where the details of the dataset and experiment are discussed in Section IV. As shown in Figure 1, we visualize several extracted hidden vectors O, where each index (GRU channel) represents a specific hidden series learned from the input. If we compare the hidden series and the mobile traffic time series, we can find some similarities between them: hidden series still retains a temporal behavior similar to that of the input series, although they have been processed by GRU leading to different levels of smoothing and noise. By observing the figure, it seems that GRU learns specific ways of scaling and introducing noises to the input series; the hidden featureas created by applying multiple levels of variability provide a better representation of the input space. Following this idea, a non RNN-based architecture may be as good as an RNN if it can learn how to modify the temporal dynamics of series in a gradual way, while the training time could be greatly reduced using only feedforward components; this is the inspiration for this work.

B. Variability-Enhanced Predictor

To present the design of VEN, we first introduce its basic building blocks and layers which are shown in Figure 2. The

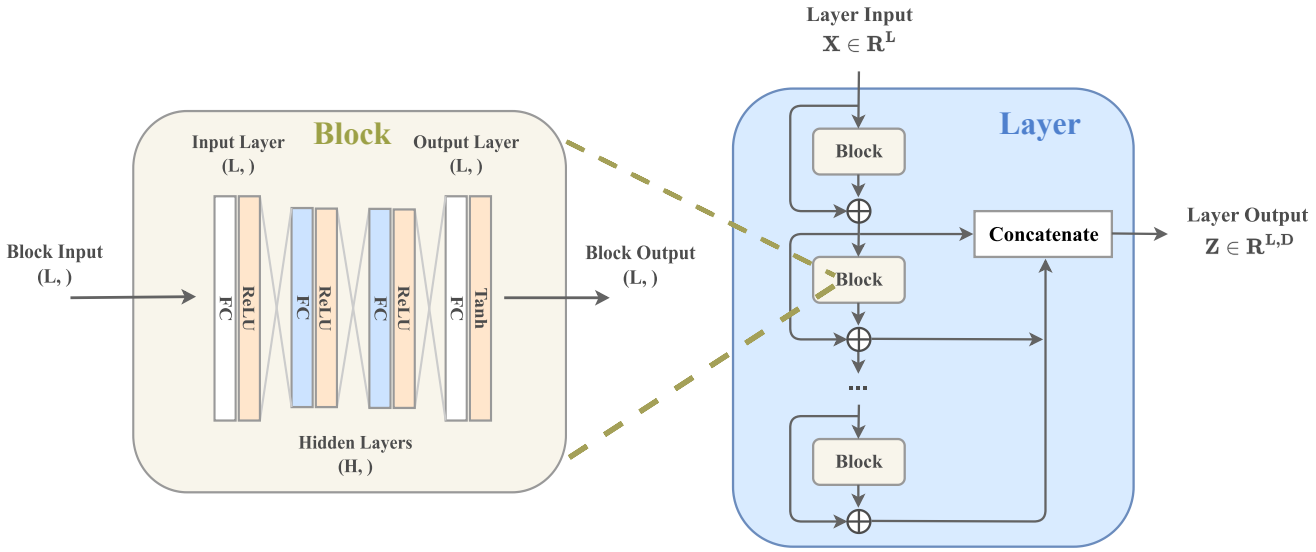

Fig. 2: Basic blocks and layers: a layer is composed of residual connected blocks whose depth is D (number of stacked blocks); each block will generate a new equal-length series which is used to enhance the variability, where L is the length of the block input and H is the number of hidden channels of the blue-coloured FC layers.

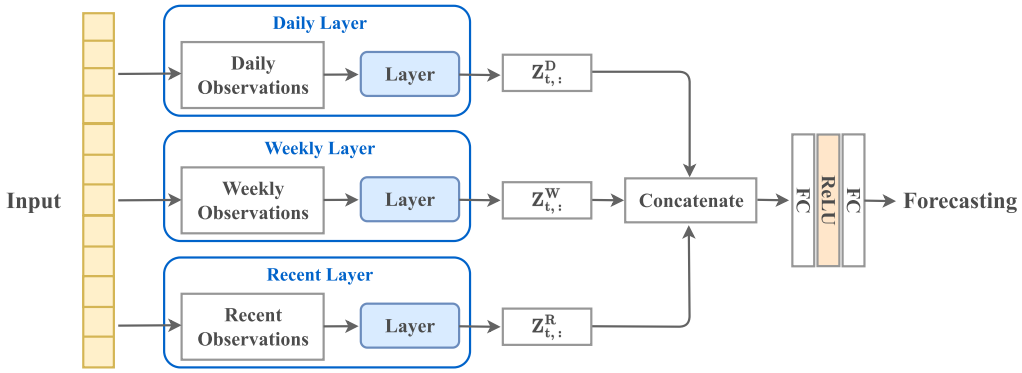

Fig. 3: Architecture of VEN, where $Z^D_{t,:}$, $Z^W_{t,:}$ and $Z^R_{t,:}$ are the values of the corresponding layer outputs at the last time step t.

common way between our model and N-BEATS is that they both use residual connections and block-layer architecture, but their mechanisms are completely different: N-BEATS runs a sequential analysis of the input signal recursively, every time it forecasts part of the predictions and removes the well-modelled part from the previous signal, which performs an ensemble style forecasting. For our model, instead of performing ensemble forecasting, the layers are designed to model the unobserved hidden states of the last observations to better describe the distribution of the predicted random variable. The backbone architecture of VEN consists in layers obtained stacking a number of basic blocks, where each block consists of four FC layers. Blocks are used to generate a bias sequence whose length L is the same as the length of its input, and the blocks are connected in an residual way within the layers; every time a block creates a bias sequence, the generated series is added to the block input to introduce a certain degree of oscillation, eventually creating a variability-enhanced version of the input series. In the design of the block, we choose Rectified Linear Unit (ReLU) as the activation function for the first three FC layers to ensure the gradients can fast propagate across the block, and the last activation function is defined as the hyperbolic tangent function (Tanh) as we need to scale the value range of output to the range of -1 and 1 in order to ensure that every time only a small modification will be made to the block input. By stacking the blocks into a layer, it is feasible to gradually modify the layer input in order to get sequences with different degrees of variability; all the variability-enhanced series are collected and concatenated at the end which forms the output of the layers represented as $Z \in R^{L,D}$, where L is the input length and D is the number of stacked blocks (also known as the depth of the layer). Comparing GRU and the proposed layer, they employ different underlying mechanisms: a GRU will generate 32 hidden series if it has 32 channels, and all hidden series are generated at the same time at each time step by updating

the previous hidden state. Conversely, at each time instant a VEN layer generates a whole new hidden series without any recurring component.

C. Application to Mobile Traffic Prediction

Figure 3 illustrates the architecture of VEN which is composed of three layers and two FC layers. Instead of using a single layer handling the whole input sequence, VEN uses three layers in parallel to process different time slots of the input series. The three layers are daily layer, weekly layer and recent layer which handles daily observations, weekly observations and recent observation respectively. Generally speaking, in time series forecasting the useful information is relatively sparse which means most of the data points do not really help but rather add undesired noise to output. In this case, processing the whole time series may not improve the forecasting performance but indeed significantly increase the processing time; in this work, we manually design three time windows to enhance the ability of VEN to make a trade-off among several temporal dependencies. Mobile traffic time series has complex temporal dynamics whose pattern is seen as a mixture of different terms; there are two important periodic patterns in mobile traffic including the daily pattern and the weekly pattern, where the daily pattern is mainly affected by human's daily schedule and the weekly pattern is dependent on the different between working day and weekend. Aside from these periodic patterns, the recent trend of mobile traffic is also very important considering that recent observations can be more useful to detect upcoming changes in mobile demand. Following this idea, for periodic patterns we only use the samples located at a local window, i.e., the samples just before and after the target instants $t' = t - nT$ with $n = 0, 1, \ldots,$ and T is referred to as the "skip period". For example, t' would be the time steps of 4 PM every day if we want to predict the mobile traffic at 4 PM, and the local window is defined as twice the length of the forecasting horizon, i.e., one hour before and after the target instant t' if we plan to predict mobile demand half an hour in advance. Daily layer and weekly layer use the same local window setting but have different skip periods T; in the daily layer, the skip period is set to 96 time steps which is the number of samples in one day, and the period is increased to 672 time steps (one week) in the weekly layer. In the recent layer, we use the observations in the last 24 hours to capture the recent changes. Once the model receives the outputs of all layers, only the values at the last time step are concatenated and used by FC layers to make predictions; the idea behind this is: the layer output can be seen as a collection of layer input series with different degrees of variability, where the last time step values represent the potential unobserved states of the last observation; the values encode the potential data distribution at the last observed time step allowing the model to learn a mapping between a distribution and predictions, this makes the model robust to the potential noise and achieve better generalization.

IV. Experiments

A. Dataset

We conduct experiments on a real-world industrial dataset to evaluate the performance of our proposed model. This is a mobile traffic dataset provided by Telecom Italia S.p.A which is the largest Italian telecommunications services provider; the dataset includes the downlink usage data collected from LTE network of a metropolitan city in Italy, describing the downlink mobile demand of 100 cells. This dataset consists of 32300 downlink usage 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 downlink demand for the next two steps (half an hour) and the next four steps (one hour). Notice that the 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. The whole dataset has been split into the train set, validation set and test set with the ratios 80%, 10% and 10% respectively.

B. Benchmarks and Performance Metrics

In the experiments, we have implemented two predictors: the VEN and a single-layer predictor composed of a basic layer and two FC layers. To compare the forecasting performance of the proposed model, twelve baseline approaches are employed, including: MLP [4], TCN [5], Transformer [6], DLinear [7], NLinear [7], N-BEATS [3], LSTM [8], GRU [9], DeepAR [10], MQ-RNN [11], LSTNet [12] and TPA-LSTM [13]. The baseline models cover the most popular approaches in deep learning-based time series forecasting, and they can be divided into two categories: non-RNN-based models (MLP, TCN, Transformer, DLinear, NLinear, N-BEATS) and RNN-based models (GRU, LSTM, DeepAR, MQ-RNN, LSTNet, TPA-LSTM); among these architectures, TPA-LSTM is one of the state-of-the-art forecasting models built based on RNN and temporal attention mechanisms. The models are trained to minimize the Mean Absolute Error (MAE) on train set, and the hyperparameters 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 employed in this paper, by merging train set and validation set, and eventually evaluated on the test set. To evaluate the prediction performance, two metrics are used, namely MAE and Mean Squared Error (MSE); they are defined as follows:

$$\mathrm{MAE} = \frac{1}{\mathrm{n} \times \mathrm{w}} \sum_{\mathrm{i}=1}^{\mathrm{n}} \sum_{\mathrm{j}=1}^{\mathrm{w}} |\mathrm{x}_{\mathrm{t+j}}^{\mathrm{i}} - \hat{\mathrm{x}}_{\mathrm{t+j}}^{\mathrm{i}}| \tag{2}$$

$$\mathrm{MSE} = \frac{1}{\mathrm{n} \times \mathrm{w}} \sum_{\mathrm{i}=1}^{\mathrm{n}} \sum_{\mathrm{j}=1}^{\mathrm{w}} (\mathrm{x}_{\mathrm{t+j}}^{\mathrm{i}} - \hat{\mathrm{x}}_{\mathrm{t+j}}^{\mathrm{i}})^2 \tag{3}$$

where n is the number of time series, w is the number of forecasting horizons, x_{t+j}^i is the ground-truth of time series i at the time step $t + j$ and $\hat{x}_{t+j}^i$ is the corresponding

TABLE I: Comparison of RNNs and single layer predictors: k is the shorthand notation for Thousand; the bold text represents the best one and the underlined text represents the second-best one.

Model	w=2 (up to 30 minutes)				w=4 (up to 60 minutes)			
	MAE	MSE	#Parameters	Training Time	MAE	MSE	#Parameters	Training Time
LSTM	0.295	0.303	265k	20.0 minutes	0.326	0.331	265k	11.4 minutes
GRU	**0.286**	0.298	**50k**	53.4 minutes	0.313	0.330	**50k**	23.0 minutes
Single-Layer Predictor (depth=8)	0.287	0.293	303k	0.9 minutes	0.306	0.305	145k	0.5 minutes
Single-Layer Predictor (depth=16)	0.289	**0.290**	605k	**0.6 minutes**	0.307	0.306	131k	**0.1 minutes**
Single-Layer Predictor (depth=32)	0.287	0.291	581k	0.7 minutes	**0.304**	**0.303**	580k	1.1 minutes

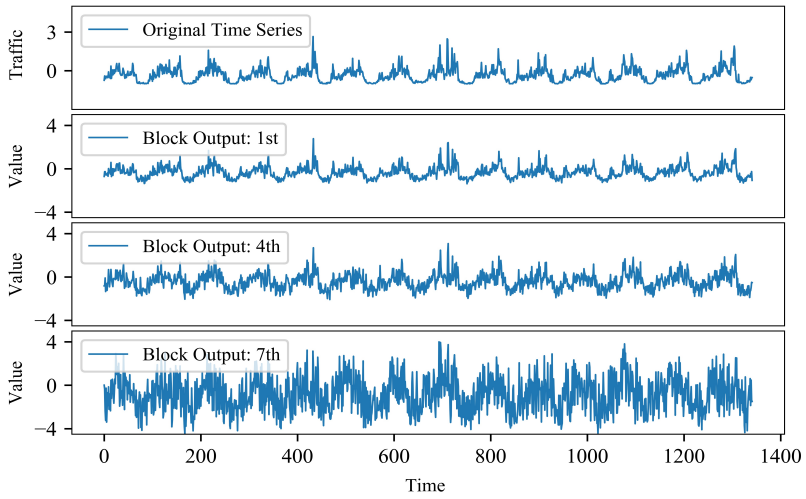

Fig. 4: Visualization of the variability-enhanced series generated by the single-layer predictor (depth=8).

prediction. Besides MAE and MSE, the number of parameters (#Parameters) and training time are also used to evalue the models, where #Parameters is used to evaluate how many parameters are included in the models, which is a good proxy of the complexity of a neural network, and training time is employed to compare the training speed of the methods. In order to compare the performance of the models, for each metric we use the bold text to indicate the best one and the underlined text to indicate the second-best one.

V. Results

To better understand the behavior of the proposed architecture, we first analyze the characteristics of the single-layer predictor; we have trained the single-layer predictors with three different depths and compare their performance with the two most widely used RNNs (LSTM and GRU). Table I evaluates the performance of RNNs and single-layer predictors, and the forecasting performance of single-layer models is better than RNNs from a global point of view: their predictions are as accurate as GRU when we make predictions two steps in advance, and their performance is significantly better than both GRU and LSTM if the forecasting horizon is four; at the same time, single-layer models require much less training time and the price to pay is the larger model size. Comparing the performance of the predictors with different depths, we do not obtain obvious performance improvement by including more blocks; according to this reason, the depth of the layers in VEN is set to 8. Figure 4 visualizes the variability-enhanced series created by the blocks of single-layer predictor (depth=8), the sequences generated at three different depths are illustrated. If we observe the series, the output of the first block is very similar to the original time series and only a little perturbation is added to the input sequence; the introduced perturbation gradually increases as more blocks are used to add oscillation, eventually resulting in a very noisy series (the output of the seventh block). In this case, we can say that the degree of introduced variability is proportional to the depth of the layer. Based on these results, we prove that the behavior of the basic layer follows our desired way, which obtains good forecasting performance by introducing different levels of variability to the input sequence while having a quite fast training speed compared to RNNs.

After discussing the characteristics and effectiveness of our proposed basic layer, we compare the performance of VEN with the twelve baseline approaches, and the results can be found in Table II. According to the results, the RNN-based baseline models generally perform better than non-RNN-based baseline models, and the best two baseline approaches are both RNN-based which are LSTNet and TPA-LSTM; the two models combine RNN architectures with other mechanisms such as temporal attention and skip connections, which

TABLE II: Comparison of the performance of models: k and M are the shorthand notation for Thousand and Million; the bold text represents the best one and the underlined text represents the second-best one.

		w=2 (up to 30 minutes)				w=4 (up to 60 minutes)			
	Model	MAE	MSE	#Parameters	Training Time	MAE	MSE	#Parameters	Training Time
Non-RNN-Based	MLP	0.301	0.297	188k	**0.1 minutes**	0.316	0.312	188k	**0.2 minutes**
	TCN	0.298	0.326	61k	24.7 minutes	0.403	0.416	64k	27.2 minutes
	Transformer	0.332	0.342	1.12M	57.8 minutes	0.382	0.390	1.12M	24.6 minutes
	DLinear	0.294	0.290	5k	0.3 minutes	0.316	0.308	11k	0.3 minutes
	NLinear	0.298	0.293	**3k**	0.3 minutes	0.319	0.306	**5k**	0.4 minutes
	N-BEATS	0.298	0.306	471k	0.9 minutes	0.323	0.321	471k	1.2 minutes
RNN-Based	LSTM	0.295	0.303	265k	20.0 minutes	0.326	0.331	265k	11.4 minutes
	GRU	0.286	0.298	50k	53.4 minutes	0.313	0.330	50k	23.0 minutes
	DeepAR	0.314	0.313	265k	11.7 minutes	0.358	0.359	265k	11.6 minutes
	MQ-RNN	0.290	0.299	141k	28.4 minutes	0.319	0.320	174k	22.9 minutes
	LSTNet	0.282	0.287	102k	47.6 minutes	0.302	0.307	102k	128.7 minutes
	TPA-LSTM	**0.278**	0.284	69k	218.0 minutes	**0.296**	0.305	115k	138.6 minutes
Our	VEN (depth=8)	0.279	**0.281**	425k	1.3 minutes	0.298	**0.301**	549k	1.5 minutes

improves the forecasting performance significantly. However, these mechanisms are usually computationally slowly and this makes LSTNet and TPA-LSTM have longer training time compared to other methods; besides this problem, LSTNet and TPA-LSTM are still seen as two state-of-the-art short-term forecasting models. For the non-RNN-based models, we can find that some models have very short training time; for example, the training time of MLP, NLinear and DLinear is less than half a minute; but at the same time their forecating performance is not that good. Even though Transformer and TCN are not built on RNNs, but their training time is much longer considering that they either use a multi-head attention mechanism or stack many dilated convolutional layers. Compared to the best baseline model TPA-LSTM, our model has almost the same performance; if we check the table, we can find that the difference in forecasting performance between VEN and TPA-LSTM is minor, but the training time of VEN is much shorter than TPA-LSTM: VEN requires approximately 1.5 minutes for training and the required training time of TPA-LSTM is hundreds of minutes. In this case, VEN obtains excellent forecasting performance while using only 1% training time of the best baseline approach. Even though N-BEATS is also a very fast model, its performance is much worse than ours.

VI. Conclusion

In this work, we have proposed VEN, a low-cost mobile traffic forecasting model aiming at making accurate short-term forecasting while reducing the training time of the network. Inspiring by the characteristics of RNNs, the backbone layer of the network has been designed to gradually introduce variability into the input time series, and the generated variability-enhanced sequences can be seen as different representations of the input sequence suffering from various degrees of variation, which better describes the input space of the mobile traffic time series. To make VEN more efficient, we carefully design three layers and each of them focuses on one specific pattern, including daily pattern, weekly pattern and recent trend. We conduct extensive experiments on a real-world mobile traffic dataset, and the results show that our model achieves excellent prediction performance while having a very fast training speed: its predictions are as accurate as the predictions of state-of-the-art forecasting models (RNN-based), while its training time is approximately 99% lower.

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] W. Jiang, "Cellular traffic prediction with machine learning: A survey," *Expert Systems with Applications*, p. 117163, 2022.
[3] B. N. Oreshkin, D. Carpov, N. Chapados, and Y. Bengio, "N-beats: Neural basis expansion analysis for interpretable time series forecasting," *arXiv preprint arXiv:1905.10437*, 2019.
[4] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016.
[5] 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.
[6] 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.
[7] A. Zeng, M. Chen, L. Zhang, and Q. Xu, "Are transformers effective for time series forecasting?" *arXiv preprint arXiv:2205.13504*, 2022.
[8] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.
[9] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint arXiv:1412.3555*, 2014.
[10] D. Salinas, V. Flunkert, J. Gasthaus, and T. Januschowski, "Deepar: Probabilistic forecasting with autoregressive recurrent networks," *International Journal of Forecasting*, vol. 36, no. 3, pp. 1181–1191, 2020.
[11] R. Wen, K. Torkkola, B. Narayanaswamy, and D. Madeka, "A multi-horizon quantile recurrent forecaster," *arXiv preprint arXiv:1711.11053*, 2017.
[12] 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.
[13] S.-Y. Shih, F.-K. Sun, and H.-y. Lee, "Temporal pattern attention for multivariate time series forecasting," *Machine Learning*, vol. 108, pp. 1421–1441, 2019.