K/Ka-Band Very High Data-Rate Receivers: A Viable Solution for Future Moon Exploration Missions

TDANet: An Efficient Solution For Short-Term 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 is crucial for optimizing the network configuration and improving Quality-of-Service (QoS); accurate mobile traffic predictions can help the network operators better configure the network and adapt to the trend of mobile demand. However, improving mobile traffic forecasting is challenging as traffic patterns exhibit strong periodicity while also retaining a certain level of randomness; this makes it difficult to model the sequence correlation. To obtain better mobile traffic predictions, we proposed a deep learning-based predictor called Temporal Dynamics Aware Network (TDANet). TDANet is carefully designed to extract both the global and the local temporal patterns employing recurrent and attention components; a linear module is employed to make the predictions more sensitive to the magnitude of mobile traffic, which makes the model better able to capture fast dynamics. Extensive experiments are conducted on a real-world mobile traffic dataset, and the results show that TDANet outperforms all the baseline models on all evaluation metrics, including both accuracy and complexity.**

***Index Terms*—wireless traffic, forecasting, deep learning**

I. Introduction

A. Background and Related Works

Nowadays, mobile demand has grown rapidly on a global scale and 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]; to better manage the network and improve the services, it is essential to be able to forecast the resource usage in the near future in order to optimize the network operation. There are many applications which require good accuracy of mobile traffic forecasting, such as base station sleeping, admission control and resource allocation and scheduling. Many works have attempted to improve traffic forecasting; early works mainly used statistical models to predict the mobile traffic including exponential smoothing [2], ARIMA [3] and others. In recent years, deep learning models have been proven to be more effective at learning complex non-linear relationships within the time series, and many works started focusing on improving the mobile traffic predictions; a number of popular time series forecasting models have been proposed such as Long Short-Term Memory (LSTM) [4] and Transformer [5]. Even though these models perform generally well, their performance on mobile traffic is quite limited because of the nature of these time series. As is known, mobile demand is strongly related to human activities, hence the time series exhibits strong periodicity while retaining a certain level of randomness. As a consequence, the temporal pattern of mobile demand is a mixture of short-term periodic pattern, long-term periodic pattern and the recent trend affected by unknown factors, resulting in a very complicated behavior which is difficult to be analyzed and predicted. Although current deep learning models are powerful at modelling the temporal correlation within the sequence, it is still hard to capture rapidly changing patterns. Another issue is related to the model design; in many cases complex models are more powerful as a large number of parameters allows them to better fit data distribution. However, we find this is not completely true for short-term time series forecasting in practice; the forecasting model may tend to overfit the regular pattern of time series while ignoring the less typical patterns, which limits the forecasting performance. To obtain accurate predictions, the forecasting model should be able to learn complex patterns with a small number of parameters. To address these challenges, we propose a novel deep learning model called TDANet. TDANet is carefully designed to model complex temporal dynamics using few parameters, and the main contributions are summarized as follows.

- We propose a novel model which improves the mobile traffic forecasting performance by leveraging both the periodic dependencies and the most recent measurements.
- Experiments are carried out on a real-world mobile traffic dataset, and the results show the TDANet outperforms all benchmarks in mobile traffic forecasting with much fewer parameters.

B. Problem Formulation

In this work, the goal is to perform short-term time series forecasting using a model learned from past observations. 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:

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

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 a selected predictor.

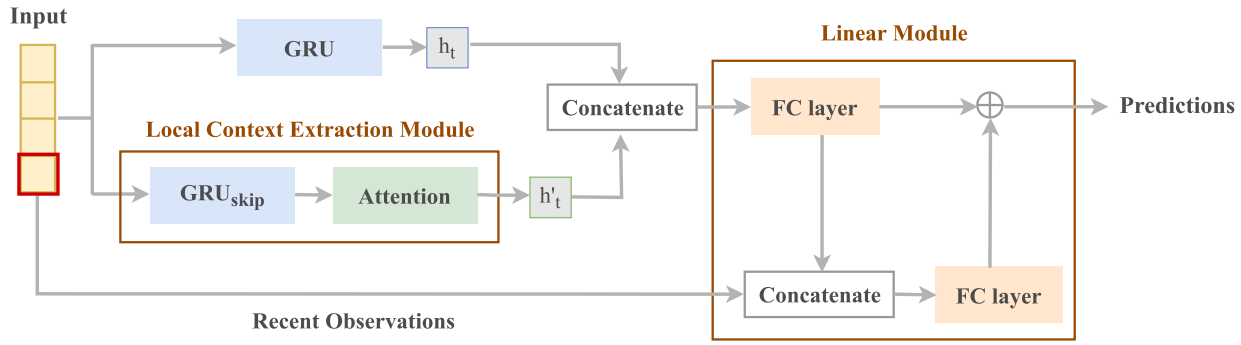

Fig. 1: Architecture of TDANet, where h_t and h'_t are the hidden vectors extracted by GRU and the local context extraction module respectively; the details of the local context extraction module are illustrated in Figure 2.

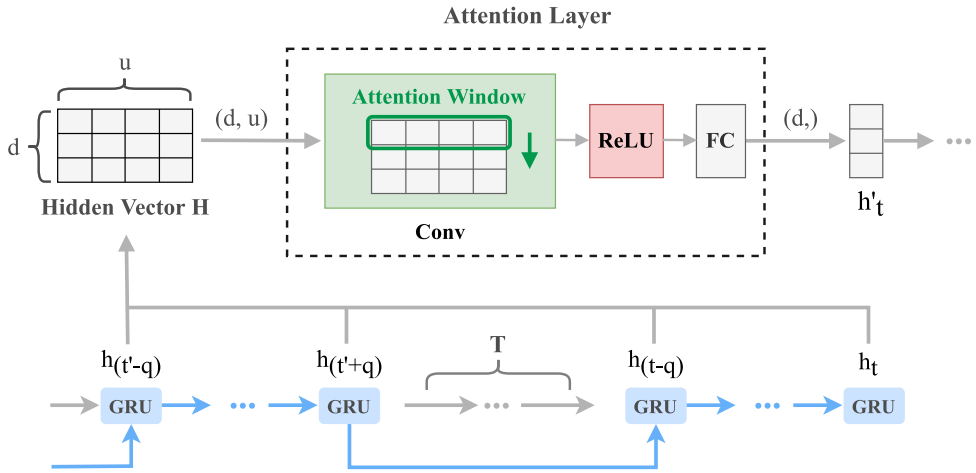

Fig. 2: The local context extraction module. t is the most recent time step, t' is the time step one or more skip periods away from t, T is the skip period and q defines the size of the local context. GRU_{Skip} first calculates the hidden state of the non-skipped time steps, then all the hidden states are fed into the attention layer to obtain the weighted average hidden state h'_t.

II. Forecasting Model

Figure 1 presents the architecture of TDANet. For short-term mobile traffic forecasting, the model is designed for learning the complex temporal dynamics of mobile traffic time series employing few parameters, which allows to obtain accurate mobile traffic forecasting with low computational cost. To perform time series forecasting, TDANet first extracts the global and the local temporal patterns using Gated Recurrent Unit (GRU) [6] and the local context extraction module separately, then the extracted hidden vectors are concatenated and fed into the linear module to make predictions; within the linear module, the first fully connected layer (FC) is responsible for making initial predictions, and these predictions are concatenated with the most recent observations; the concatenated vector is used by the second FC layer to generate linear offsets, finally the final predictions are created by adding initial predictions and linear offsets element-wise. In this section, we present the layers of TDANet in detail.

A. Recurrent Component and Local Context Extraction Module

Once TDANet receives the input, the time series is first fed into the recurrent component GRU and the local context extraction module which is composed of the skip-Gated Recurrent Unit (GRU_{Skip}) and an attention layer. The objective of the components is to extract the temporal pattern of mobile traffic, where GRU focuses on learning the global temporal pattern and the local context extraction module focuses on learning the local temporal pattern. Based on the use of GRU, the model obtains a hidden state h_t extracted by GRU at time stamp t, and this vector summarizes the coarse global information of the sequence.

Compared to the conventional Recurrent Neural Network (RNN), GRU and LSTM [7] are designed to model the long-term dependencies of time series and address the vanishing gradient problem; by applying the gate mechanism, they can better capture the correlation among samples. Even though the

gate mechanism counters vanishing gradients, it is still difficult to learn the temporal dependencies between far away samples when the time series is rather long; if the model cannot capture such long-term dependencies, there is a potential risk of losing the ability to capture periodic patterns which limits the performance in time series forecasting. To alleviate this issue, we propose to use GRU_{Skip} to learn the local patterns in preceding time steps. In practice, in several domains it is often the case that a time series exhibits strong periodicity, e.g. mobile traffic time series or outdoor temperature time series; in this case, a common approach to predict future values is to refer to not only the recent samples but also the samples at past instants $t - nT$ shifted by an integer number of periods T, as the periodic behavior would help the model estimate the future values.

Following this idea, GRU_{Skip} is designed to enhance the model's use of long-term dependencies; the architecture of GRU_{Skip} is presented in Figure 2. Instead of feeding all input samples to the recurrent component, we only use the samples which can represent a so-called *local context*, 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". By considering that the mobile traffic demand is strongly related to human activities, the periodic pattern could appear multiple time steps earlier or later than the last observed time step t which leads to a shift on time axis; under this scenario, we prefer to focus on the samples within a predefined window centered around the time steps t', instead of focusing on just the samples at the time steps t'. For example, we would define a window which only includes the past samples from 9 AM to 11 AM if we want to predict the mobile traffic at 10 AM, and this predefined window is called the local context of this specific time step. In the GRU_{Skip} module, we have to define two hyper-parameters: window size q and skip period T; the former defines the size of the local context and the latter defines how many time steps the recurrent component needs to skip. In experiments, the window size is set to 4 which includes the samples one hour before and after the given time step, and the skip period is selected as 96, i.e. the number of samples corresponding to one day which is an obviously typical periodicity for mobile phone usage. For a given time series, the output of GRU_{Skip} is a collection of hidden states $H \in R^{d \times u}$ where d is the hidden dimension of GRU_{Skip} and u is the number of extracted hidden states.

Once the model computes the hidden vector H, the vector is passed to the attention layer to calculate the weighted average h'_t. The attention layer is composed of a convolutional layer (Conv), a rectified linear unit (ReLU) activation function and a FC layer. The reason why we calculate the weighted average h'_t instead of using the hidden state h_t is the following: even though GRU_{Skip} focuses on a specific time slot of each day, there is no guarantee that all past days are equally important, and multiple types of periodic patterns may be found in the time series, such as the daily pattern and the weekly pattern, whose combined effect is not easy to model. Besides the periodic patterns, the most recent values of the time series are also very representative as they reflect the most recent dynamics of the mobile users behavior. Therefore, the future values may follow a complex periodic pattern while also depending on the most recent activities of the users. The hidden state h_t focuses more on the recent trend and downplays the periodic component. In order to optimally weight all factors in the forecast, the attention layer assigns different weights to the extracted hidden states and the weighted average h'_t provides an aggregated hidden state that can better summarize the behaviour of mobile traffic at the target time slot. To achieve this, the attention layer uses a Conv layer as the sliding window to extract the pattern of hidden states; note that the length of the attention window is u and the height is 1, and we slide the window along the hidden dimension d; the vector extracted by the window is process by ReLU and passed to the FC layer to calculate the weighted hidden state. This process is formulated as:

$$h'_{s,t} = \text{FC}(\text{ReLU}(\text{Conv}(\text{H}_{\text{s,1:u}}))) \tag{2}$$

where s is the current sliding position on the hidden dimension, $h'_{s,t}$ is the weighted average of $H_{s,1:u}$ at the sth position of hidden dimension. Eventually, we create a new vector by concatenating the weighted hidden state h'_t extracted by GRU_{Skip} and the hidden state h_t extracted by GRU, and this new vector is fed into the linear module to make the predictions.

B. Linear Module

The last two layers of TDANet are two FC layers which are used to make the final predictions. Compared to the models built based on feedforward neural networks, the RNN-based models show their practical superiority at learning temporal pattern of time series because the strong non-linearity allows them to have a larger search space and better fit to the data distribution. However, their non-linear nature raises an issue that the RNN-based models are not sensitive to the magnitude change of the input; when an increasing trend or a decreasing trend is observed recently, these models would slowly adapt to the change resulting in a delay in making predictions following the recent trend, which limits the time series forecasting performance. To improve the model's ability to capture recent trends, we add linear offsets to the predictions to make them more sensitive to the input scale. To do this, first the concatenated vector learned by the recurrent components is fed into the top FC layer to make the initial predictions $o \in R^w$, then the initial predictions o and the most recent p observations are fed into the bottom FC layer to calculate linear offsets, and the final predictions are obtained by computing the element-wise summation of o and linear offsets. The linear offsets are computed as follows:

$$\begin{aligned} &\text{Input} = \text{Concat}(\mathbf{x}_{\text{-p:t}}, \text{o}) \\ &\text{Linear offsets} = \tanh(\text{FC}(\text{Input})) \end{aligned} \tag{3}$$

the input of the FC layer is created by concatenating the recent observations and the initial predictions, which makes

TABLE I: Comparison of the performance of models; k and M are the shorthand notation for Thousand and Million.

	w=2 (up to 30 minutes)			w=4 (up to 60 minutes)		
Model	MAE	MSE	#Parameters	MAE	MSE	#Parameters
MLP	0.301	0.297	188k	0.316	0.312	188k
LSTM	0.295	0.303	265k	0.326	0.331	265k
GRU	0.286	0.298	50k	0.313	0.330	50k
DeepAR	0.314	0.313	265k	0.358	0.359	265k
MQ-RNN	0.290	0.299	141k	0.319	0.320	174k
TCN	0.298	0.326	61k	0.403	0.416	64k
LSTNet	0.282	0.287	102k	0.302	0.307	102k
Transformer	0.332	0.342	1.12M	0.382	0.390	1.12M
TDANet (ours)	**0.278**	**0.284**	**33k**	**0.295**	**0.298**	**36k**

the FC layer aware of both the recent magnitude and the predicted waveform, reducing the difficulty of generating the linear offsets. The final predictions $\hat{\mathbf{x}}_{t+1:t+w}$ are obtained by summing the two terms:

$$\hat{\mathbf{x}}_{t+1:t+w} = o + \text{Linear offsets} \tag{4}$$

III. Experiments and Results

A. Dataset

We conduct experiments on a real-world mobile network dataset to evaluate the performance of the proposed model. This is an industrial dataset provided by Telecom Italia S.p.A which is the largest Italian telecommunications services provider. The downlink usage data is collected from LTE network of a metropolitan city in Italy, covering 100 cells. This dataset consists of 32300 downlink usage time series which measure the downlink demand of cells, 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 usage 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.

B. Benchmarks and Performance Metrics

To compare the forecasting performance of the proposed model, a set of baseline approaches are employed, including: MLP [8], LSTM [7], GRU [6], LSTNet [9], TCN [10], DeepAR [11], MQ-RNN [12] and Transformer [13]. The baseline models cover the most popular approaches in deep learning-based time series forecasting. 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 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 Mean Absolute Error (MAE) and Mean Squared Error (MSE); they are defined as follows:

$$\text{MAE} = \frac{1}{\text{n} \times \text{w}} \sum_{\text{i}=1}^{\text{n}} \sum_{\text{j}=1}^{\text{w}} |\text{x}_{\text{t+j}}^{\text{i}} - \hat{\text{x}}_{\text{t+j}}^{\text{i}}| \tag{5}$$

$$\text{MSE} = \frac{1}{\text{n} \times \text{w}} \sum_{\text{i}=1}^{\text{n}} \sum_{\text{j}=1}^{\text{w}} (\text{x}_{\text{t+j}}^{\text{i}} - \hat{\text{x}}_{\text{t+j}}^{\text{i}})^2 \tag{6}$$

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 prediction. Besides MAE and MSE, the number of parameters (#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.

C. Results

We evaluate the performance of these models on test set, and the results are reported in Table I. Among the baseline approaches, we observe that the RNN-based models such as GRU, MQ-RNN and LSTNet obtain better performance compared to the others. Comparing to the baseline models, TDANet shows its superiority in mobile traffic prediction and outperforms all of them on all metrics for both 2 steps forecasting and 4 steps forecasting. Besides the excellent forecasting performance, TDANet is also very light-weight as it only has around 33k parameters. LSTNet is the best baseline model having the lowest MAE and MSE; its forecasting performance is slightly worse than TDANet, but it has 102k parameters which is approximately three times as many as TDANet. Another comparison can be made between TDANet and GRU: even though TDANet consists of two GRU components, its architecture can extract important features more efficiently, which means the GRU components do not require many parameters to learn the correlation of time series, eventually leading to even fewer parameters than baseline GRU model. From this point of view, the design of TDANet allows to model and predict mobile traffic in a more effective way with much fewer parameters, and the small model size makes it feasible to deploy TDANet on mobile or embedded devices whose memory is quite limited; hence, it is possible to use TDANet at the edge of a cellular network system to satisfy the

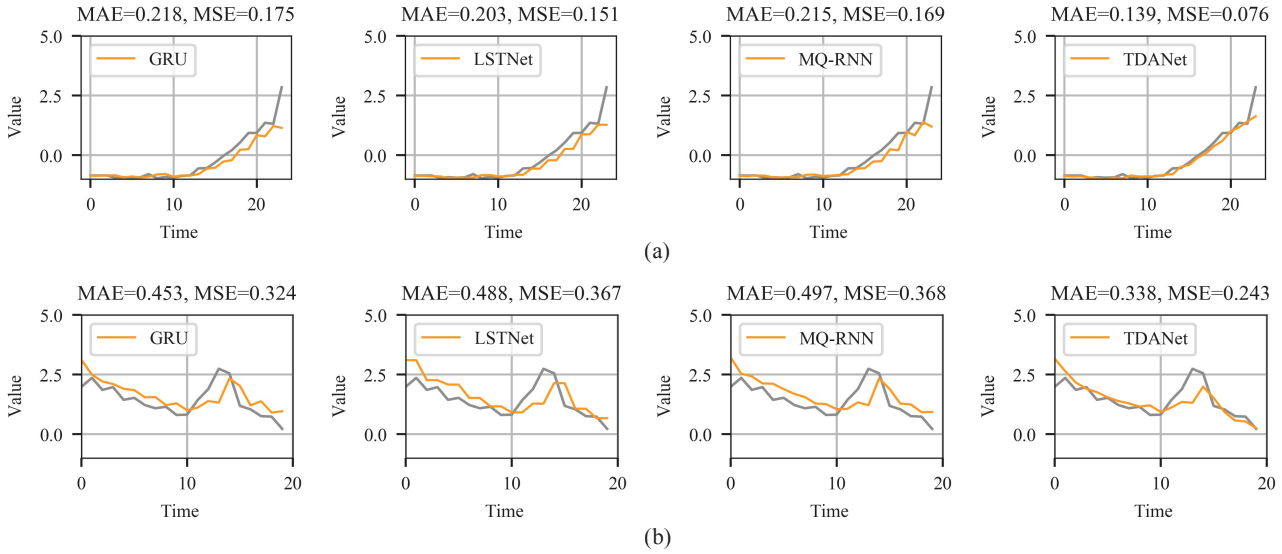

Fig. 3: Comparison of the predictions made by RNN-based models. The gray line is the ground truth and the orange line is the forecasting; the y-axis is the normalized value of downlink usage. (a) increasing trend of mobile traffic; (b) decreasing trend of mobile traffic.

specific mobile traffic forecasting needs of some applications, and further improve the quality of experience for end users. As we discussed in the previous sections, the RNN-based models are not sensitive to the recent trend of time series, which is a drawback limiting their forecasting performance. In Figure 3, we visualize the forecasting of four RNN-based models; two scenarios are considered: increasing and decreasing trend of mobile traffic. In the figure, it is obvious that the RNN-based baseline models cannot adapt very well; on the other hand, the proposed model can better follow the fast changes in the time series which leads to much lower MAE and MSE.

IV. Conclusion

Accurate mobile traffic forecasting plays a key role in network configuration and QoS improvement. Due to the nature of mobile traffic, it is very difficult to model both its long-term dependencies and short-term randomness. The widely used deep learning models have limited performance of mobile traffic forecasting while using a large number of parameters. To address these issues, we propose a novel model called TDANet which can capture the complex temporal pattern with much fewer parameters; two recurrent components and an attention layer are used to extract the global and local temporal patterns, and the linear components are employed to overcome the weakness that RNN-based models are not sensitive to the input dynamics. Experiments are carried out on a real-world mobile traffic dataset, and the results show a clear superiority of TDANet in mobile traffic forecasting.

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] D. Tikunov and T. Nishimura, "Traffic prediction for mobile network using holt-winter's exponential smoothing," in *2007 15th International Conference on Software, Telecommunications and Computer Networks*. IEEE, 2007, pp. 1–5.
[3] F. Xu, Y. Lin, J. Huang, D. Wu, H. Shi, J. Song, and Y. Li, "Big data driven mobile traffic understanding and forecasting: A time series approach," *IEEE transactions on services computing*, vol. 9, no. 5, pp. 796–805, 2016.
[4] Y. Zhu and S. Wang, "Joint traffic prediction and base station sleeping for energy saving in cellular networks," in *ICC 2021-IEEE International Conference on Communications*. IEEE, 2021, pp. 1–6.
[5] Z. Chen, E. Jiaze, X. Zhang, H. Sheng, and X. Cheng, "Multi-task time series forecasting with shared attention," in *2020 International Conference on Data Mining Workshops (ICDMW)*. IEEE, 2020, pp. 917–925.
[6] 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.
[7] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.
[8] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016.
[9] 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.
[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] 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.
[12] R. Wen, K. Torkkola, B. Narayanaswamy, and D. Madeka, "A multi-horizon quantile recurrent forecaster," *arXiv preprint arXiv:1711.11053*, 2017.
[13] 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.