DO WE NEED TOO MUCH ATTENTION? A TIME SERIES PERSPECTIVE

Anonymous authors
Paper under double-blind review

ABSTRACT

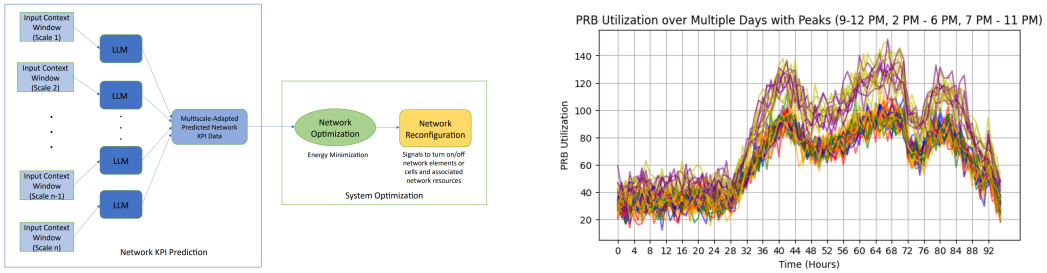
The present work proposes a method for time series prediction with applications across domains, like in agriculture for optimal crop timing, stock market forecasting, and in e-commerce. Studies suggest that with slight modification, Large Language Models (LLMs) can be adapted for time series prediction. In the telecom sector, this approach could help in significant energy conservation during network operations. In this work, various models have been evaluated for this purpose and their performances are compared that includes traditional Machine Learning and Deep Learning methods like ARIMA, RNNs and LSTMs. More recent LLM-based models were also explored such as Chronos, and PatchTST which utilizes fewer attention layers compared to Chronos. It was surprising to observe that among these models, PatchTST achieved the best performance only after fine-tuning. While Chronos is designed for zero-shot forecasting and captures some intricate temporal dependencies, PatchTST's multiscale input helps the model to understand the macro and the micro level trends and therefore might help it perform better than other methods. The results seem to indicate that effective forecasting could be achieved with fewer attention layers when supported by well-engineered input contextual representations.

1 INTRODUCTION

LLMs specialize in capturing the patterns and features of sequential data. Leveraging this ability of LLMs for time series prediction has been proven to work well. `Chronos` (Ansari et al., 2024) is one of the works that has explored this direction. It achieved zero-shot time-series forecasting. Their methods for quantisation of the real values of the time series sequences was a key to directly using the existing LLM models. However, it was found that it failed to capture some global trends when used on our dataset directly without any fine tuning for prediction.

To address this challenge, incorporating a multiscale input with hierarchical contexts of the time series was hypothesized to be beneficial, and experimental results proved it to be correct. `PatchTST` (Nie et al., 2022), which employs a vanilla Transformer encoder as its core architecture (Vaswani, 2017), was used for this purpose. The model takes a multivariate input that includes multiple scales of the same data sequence. This design enabled the model to capture both local and global features, enhancing its understanding of the underlying spatio-temporal dynamics. Details on the previous and related works have been provided in the section §A.1.

In the telecom sector, prediction of Physical Resource Block (PRB) utilization would play a critical role for optimizing network resource allocation processes. This work tries to analyze different forecasting methods to predict the network resource utilization which can further be employed for assessing the network elements required to meet the predicted demand. By leveraging these forecasts, network cells can be dynamically activated or deactivated, optimizing resource allocation and reducing energy consumption. Energy costs for network operations can run into several millions of dollars annually (for example this cost is approximately $1.6 billion dollars in AT&T's network (Dano, 2022)). In addition, on average, 73% of this energy is consumed in the Radio Access Network (RAN) (Kolta, 2021). In our study, on average we have observed a reduction in energy consumption from 27% - 36% across different network configurations based on such AI-based prediction of radio access network utilization which can result in significant cost savings for operators in their networks.



(a) (b)

Figure 1: Diagram (a) illustrates our use case in network energy optimization architecture. In this design, Large Language Models (LLMs) process identical inputs at varying time scales, allowing the Multi-Scale Adapter to integrate both coarse- and fine-grained predictions. This approach enhances decision-making by leveraging information across multiple temporal resolutions. Plot (b) shows the generated dataset for 96 intervals per day (of 50 days)

For the study in this paper, a synthetically generated dataset was employed to analyse different methods for time-series forecasting. This dataset mimicked the PRB utilization in networks using random processes. These processes were modeled to replicate the real-world behavior and trends observed in PRB utilization. It includes variations in usage that correspond to expected fluctuations over time. The dataset generation is explained in detail in the section 2.1.

2 EXPERIMENTS

Our experiments utilized multiple models to generate datasets for future 6G networks. During the development of Digital Twins for 6G networks, we identified energy optimization as a critical challenge for building sustainable networks (ITU, 2023). To predict the utilization of each network element, we selected PRB Utilization as the primary metric and designed a system, as illustrated in Figure 1(a). In this paper, we present the challenges associated with time series prediction using LLMs and evaluate the performance of alternative non-attention-based models. Section 2.1 provides a detailed description of the dataset generation process, while Section 2.2 offers a brief overview of the models employed in the experiments. Finally, Section 2.3 outlines the experimental setup utilized in this study.

2.1 DATASET

To model realistic trends in PRB utilization, the generated synthetic dataset, called *PRB Data*, follows a combination of predefined daily utilization and variance patterns. This reflects typical network behavior which can include predicable variations at different times of the day. The mean and variance of the distribution used to sample the value for each time step were derived based on the hour of the day and the day of the week, which aligns with the observed trends in real-world scenarios. For example, residential areas exhibit higher utilization during mornings and evenings on weekdays, while weekends show higher usage due to increased activity in the residential area.

The mean utilization at each hour was calculated using piecewise functions incorporating smooth transitions and sinusoidal components to capture peak and off-peak periods. Variance was modeled to reflect higher variability during peak hours and lower variability during off-peak periods, having a range of 1 to 16. Random noise sampled from Gaussian distributions scaled 0 was added to both means and variances, respectively, to introduce day-to-day variability. The range of PRB values is 0 to 120. When the utilization value crosses 100, it indicates that more capacity cells need to be turned on to meet the high demand.

To achieve different temporal resolutions and account for varying scales within each sequence, different numbers of intervals per day were selected. In this case, the intervals chosen were 96, 48, 24, 12, 8, 6, and 4. Total of 7 different scales. The final dataset shape is (number of sequences, sequence length, number of scales). In this work, it is (10K, 512, 7) for the input context which has

information of 512 time steps for every sequence and the corresponding output shape is (10K, 96, 7), which essentially means predicting the next 96 time steps for each of the seven scales. Figure 1(b) is the plot of the generated time series data for 50 consecutive days. In this plotting example, the number of intervals considered for a day was 96. Hence, there are 96 values in the time series sequence for a day. It can be seen the variations in the demand through the course of the day are modeled reasonably well, according to our notions and understanding of the PRB utilization in the industrial area.

Further, for investigating the anomaly robustness of the model, another dataset, called *PRB Data-Robust* was generated which includes sudden spikes for a few minutes, sudden outages for some hours and sustained periods of abnormally high or low utilization for more than an hour. This was done to reflect the real-life scenarios, where there could be power cuts, fluctuations, or increase in demand of the network resources due to some events like live sports broadcasts.

2.2 MODELS

Time series prediction can be performed using statistical, machine learning, and deep learning models. We experimented with `ARIMA`, `RNN`, `LSTM`, `Chronos`, and `Patch-TST`, which have demonstrated effectiveness in forecasting network element utilization. Each model offers unique advantages in handling time-dependent data.

`ARIMA` is a well-established statistical method that models time series data through autoregression, differencing, and moving averages. By tuning its parameters (p, d, q), `ARIMA` can effectively capture linear patterns and trends in univariate time series data. Deep learning models such as `RNNs` and `LSTMs` are designed to learn temporal dependencies in sequential data. `RNNs` use recurrent connections to retain information over time, while `LSTMs` incorporate memory cells and gating mechanisms to mitigate vanishing gradient issues, enabling them to learn long-term dependencies more effectively. `Chronos` applies language model architectures, such as `T5`, to time series forecasting by tokenizing data and training models with minimal architectural modifications. This allows `Chronos` to perform well in both zero-shot and in-domain forecasting tasks without extensive fine-tuning. `PatchTST` introduces an innovative approach by using patching and channel-independence, capturing both local semantic structures and long-range dependencies in time series data. By processing each univariate series separately, `PatchTST` reduces computational complexity while improving performance in multivariate forecasting.

2.3 EXPERIMENT SETUP

As outlined in Section 2.1, the dataset used for this study comprises time series data with a context length of 512, provided as input to the models discussed in Section 2.2. The prediction length was set to 96 time steps. Out of 10,000 samples, 8,000 samples were allocated for training, while 2,000 samples were used for validation and testing. The dataset structure was designed to ensure a balanced evaluation of model performance across different architectures.

`ARIMA`, `RNN`, `LSTM`, and `Chronos` processed each channel individually, as they lack multi-channel processing capability. In contrast, `PatchTST` efficiently handles multi-channel inputs, allowing it to process the entire dataset simultaneously. This distinction influences computational efficiency and overall model accuracy, as models that process data sequentially may face increased training time and reduced scalability. For `ARIMA`, an empirical analysis determined that (p, d, q) as $(10, 1, 0)$ configuration performed best. The `RNN` model achieved optimal results with two recurrent layers and a learning rate of 0.001. The `LSTM` model was structured with 50 hidden layers and a single output layer. These configurations were finalized after evaluating performance trade-offs, balancing accuracy and computational cost. The `Chronos` model, built on the pre-trained `T5` architecture with 700M parameters, was applied for zero-shot inference. The same experiment was conducted on the `PatchTST` (602K parameters) with seven channels simultaneously. Based on the results we tried fine-tuning `PatchTST`, to improve forecasting accuracy by adapting to domain-specific time series patterns.

Experiments on Google Colab with T4 GPUs and an NVIDIA A100 40GB GPU ensured efficient execution, reduced training time, and enabled comprehensive model evaluation for time series forecasting.

3 RESULTS AND DISCUSSION

Table 1 summarizes the results of all experiments conducted, with the Mean Squared Error (MSE) being used as the evaluation metric. Among the tested models, `PatchTST`, after fine-tuning, achieved the best performance with an MSE of 8.00, showcasing its ability to retain local semantic information due to its patching technique and its ability to capture longer history at the same time. For further experimentation, another dataset containing anomalies was used to evaluate the same fine-tuned `PatchTST` model. It achieved an MSE of 10.86, demonstrating its adaptability to anomalous patterns. Also, for comparison, the same datasets were used for evaluating the pre-trained PatchTST model. It is observed that there is a huge improvement in the performance of the model after fine-tuning as compared to the pre-trained model, from 468.70 MSE on the anamolous dataset and 465.58 MSE on the non-anomalous data to 10.60 and 8.00 MSE respectively.

`Chronos`, another transformer-based model, performed poorly, likely due to the lack of fine-tuning on the synthetic dataset. The pretrained `Chronos` model was used directly for forecasting, which might have limited its ability to capture specific global trends present in the dataset. Although `Chronos` was designed to be able to perform zero-shot forecasting.

`LLM` and `RNN` were also evaluated, yielding MSE values of 56.22 and 68.54, respectively. These models outperformed `Chronos`, possibly because they were trained directly on the synthetic dataset, allowing them to learn the dataset-specific temporal patterns. `ARIMA`, a statistical time series forecasting model, produced an MSE of 186.84, reflecting its limitations in handling the complex temporal structure present in the dataset.

Dataset	*PRB Data*						*PRB Data-Robust*	
Model	ARIMA	RNN	LSTM	Chronos	PatchTST	PatchTST (FT)	PatchTST	PatchTST (FT)
MSE	186.54	68.54	56.22	181.59	465.58	**8.00**	468.70	**10.60**

Table 1: The table above presents all experiment results. "FT" indicates the fine-tuned model, with the best results in **bold**. The *PRB Data-Robust* dataset, containing additional anomalies, was used to evaluate model robustness (see Section 2.1).

4 CONCLUSION AND FUTURE WORKS

In Section 3, we emphasize the role of multi-channel inputs in time series forecasting and examine recent deep learning advancements aimed at enhancing accuracy while reducing computational demands. Our findings highlight a key challenge in applied deep learning: specialized forecasting models, such as Chronos, do not always outperform more general approaches. Despite its claimed zero-shot capabilities, Chronos struggles to capture complex temporal dependencies, resulting in a performance gap. In contrast, non-attention-based networks achieve comparable accuracy, challenging the notion that attention mechanisms are crucial for high-quality predictions. Moreover, PatchTST, with fewer attention layers and parameters, performs exceptionally well, questioning the growing reliance on increasingly complex models.

Looking ahead, we aim to integrate self-improving systems that adapt to changing conditions with minimal human oversight. This involves developing models that continuously learn from new data, enhancing performance without extensive retraining. Future work will focus on real-world datasets, optimizing existing techniques, and exploring innovative methods to reduce reliance on computationally intensive architectures. Addressing challenges in large-scale data processing and improving model interpretability will be crucial for advancing time series forecasting, particularly within digital twin frameworks for more effective simulation and optimization. These efforts will help bridge the gap between theoretical advancements in deep learning and their real-world applications.

Ultimately, selecting the right number of model parameters, number of attention heads, and network depth is essential for optimizing performance based on the specific forecasting task. A well-balanced approach can lead to more efficient, customized AI designs that minimize energy and memory usage while adapting to domain-specific requirements, ultimately enhancing the effectiveness of AI-driven forecasting systems.

REFERENCES

Abdul Fatir Ansari, Lorenzo Stella, Caner Turkmen, Xiyuan Zhang, Pedro Mercado, Huibin Shen, Oleksandr Shchur, Syama Sundar Rangapuram, Sebastian Pineda Arango, Shubham Kapoor, et al. Chronos: Learning the language of time series. *arXiv preprint arXiv:2403.07815*, 2024.

M Dano, Sep 2022. URL https://www.lightreading.com/sustainability/how-at-t-s-network-chief-hopes-to-cut-a-1-6b-electricity-bill.

Abhimanyu Das, Weihao Kong, Rajat Sen, and Yichen Zhou. A decoder-only foundation model for time-series forecasting. *arXiv preprint arXiv:2310.10688*, 2023.

ITU, Nov 2023. URL https://www.itu.int/dms_pubrec/itu-r/rec/m/R-REC-M.2160-0-202311-I%21%21PDF-E.pdf.

Emanuel Kolta, June 2021. URL https://data.gsmaintelligence.com/api-web/v2/research-file-download?id=60621137&file=300621-Going-Green-efficiency-mobile.pdf.

Yuqi Nie, Nam H Nguyen, Phanwadee Sinthong, and Jayant Kalagnanam. A time series is worth 64 words: Long-term forecasting with transformers. *arXiv preprint arXiv:2211.14730*, 2022.

A Vaswani. Attention is all you need. *Advances in Neural Information Processing Systems*, 2017.

Shiyu Wang, Haixu Wu, Xiaoming Shi, Tengge Hu, Huakun Luo, Lintao Ma, James Y Zhang, and Jun Zhou. Timemixer: Decomposable multiscale mixing for time series forecasting. *arXiv preprint arXiv:2405.14616*, 2024.

A APPENDIX

A.1 RELATED WORKS

Time series forecasting has seen significant advancements with deep learning models, particularly Transformer-based architectures. Traditional methods are increasingly overshadowed by these models, which can handle complex patterns and dependencies across large datasets. `Chronos` (Ansari et al., 2024) proposes using language models for time series forecasting, employing minimal architectural changes. It treats time series as sequences and applies time series tokenization, demonstrating impressive zero-shot performance across diverse datasets. However, `Chronos` primarily focuses on univariate forecasting and may face limitations in capturing multivariate relationships without additional modifications.

Meanwhile, `PatchTST`(Nie et al., 2022) introduces a patching mechanism that divides time series into smaller subsequences, improving computational efficiency while preserving long-term dependencies. This method is effective in multivariate settings, with independent channels that address the challenge of handling multiple features in time series data. The model's ability to group time steps into subseries-level patches makes it well-suited for tasks requiring long look-back windows. However, `PatchTST` struggles to fully capture interactions between different time series features, a gap that could hinder its application in complex forecasting tasks.

In parallel, `TimeMixer`(Wang et al., 2024) and `TimesFM`(Das et al., 2023) introduce complementary approaches for improving forecasting accuracy. `TimeMixer` uses a multiscale mixing architecture to decompose time series into fine- and coarse-grained components, capturing both detailed seasonal fluctuations and broader trends. This allows for a more nuanced understanding of temporal patterns, though it can be computationally expensive. On the other hand, `TimesFM` leverages a foundation model for time series forecasting, demonstrating strong zero-shot performance by utilizing a diverse set of datasets for pretraining. While it excels in zero-shot settings, `TimesFM`'s reliance on a fixed context length may limit its ability to adapt to varying granularities of time series data. `Chronos` and `PatchTST`, highlight the lesser complexity and potential of deep learning in time series forecasting, compared to other approaches.