Fine-Tuned MLP-Mixer Foundation Models as data-driven Numerical Surrogates?

Imran Nasim
IBM UK; University of Surrey
imran.nasim@ibm.com
i.nasim@surrey.ac.uk

João Lucas de Sousa Almeida
IBM Research Brazil
joao.lucas.sousa.almeida@ibm.com

Abstract

Scientific Machine Learning (SciML) has significantly advanced climate science by enabling precise forecasting of complex dynamical systems. While state-of-the-art models excel in domain-specific tasks, recent advancements in time series-based foundation models seek to replicate the success seen in natural language processing and computer vision. This study investigates whether a "small" MLP-Mixer-based foundation model, Tiny Time Mixers (TTMs), can be fine-tuned to accurately forecast complex real-world dynamical systems while adhering to practical resource and cost constraints. Our findings suggest that TTMs are sensitive to the dynamical characteristics present in the training data, particularly in terms of amplitude and periodicity, yet significant variations in forecast accuracy were observed within the same training distribution. These results highlight the need for further adaptation of TTMs to enhance their robustness in specialized SciML forecasting tasks.

1 Introduction

In recent years, Scientific Machine Learning (SciML) has rapidly emerged as a transformative approach in climate science, particularly in forecasting complex dynamical systems. Within the SciML toolkit, there exist a vast number of state-of-the-art architectures such as: Neural ordinary differential equations (NODEs) which model time-evolving systems by learning continuous-time dynamics directly from data [3], Physics-informed neural networks (PINNs) which embed physical laws into the learning process enabling models to predict behavior consistent with known scientific principles while also leveraging data-driven approaches [11], Deep Neural Operators (DeepONets) which generalize neural networks to learn operators mapping between function spaces [8]. Although each of these methods has shown significant promise in domain specific tasks, there is growing interest in foundation models (FMs) for time series forecasting which aim to replicate the success seen in NLP and vision domains [2, 16, 14]. Unlike single-task designs, FMs serve as a versatile base that can be adapted through transfer learning (TL) to perform various downstream tasks with minimal additional data. Recent time-series FMs like TimesFM [5], Lag-llama [12], Chronos [1], and Moment [7] have demonstrated strong zero-shot forecasting capabilities. However, these models often require substantial computational resources, making them less practical for specialized SciML forecasting tasks. This study addresses a key question: *Can a "small" time series-based foundation model be fine-tuned to accurately forecast a complex real-world dynamical system under practical resource and cost constraints?* To this aim, we consider the recently developed "small" pre-trained MLP-Mixer based foundation model Tiny Time Mixers (TTMs) [6]. Despite its size, this model has demonstrated state-of-the-art performance in zero and few-shot forecasting of multivariate time-series data and as such there has been interest in applying such MLP-Mixer architectures within the field of SciML [10]. In this study, we fine-tune the TTM model using real-world temperature data within a hyperparameter optimization framework and evaluate its performance on downstream forecasting tasks. Our findings suggest that forecast accuracy is sensitive to the amplitude and periodicity of

D3S3: Data-driven and Differentiable Simulations, Surrogates, and Solvers @ NeurIPS 2024.

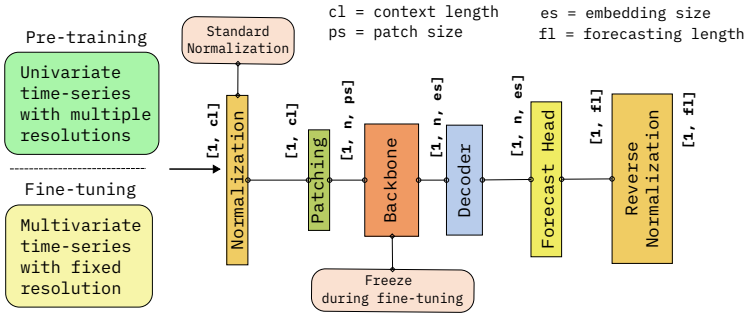


Figure 1: The general neural architecture of Tiny Time Mixer.

the dynamics present in the training data, with notable deviations even within the same training distribution. These results suggest that while TTMs hold promise, further adaptation is required to enhance their robustness in complex SciML scenarios.

2 Methods

Foundation Model: TTM is an architectural improvement over TSMixer [4], a deep learning model designed for multivariate time series forecasting using historical data. This architecture consists of two main components: time-mixing MLPs and feature-mixing MLPs, which are shared across all time steps and capture covariate information, respectively. Temporal projection is used to map the time series from its original input length to the target forecast length, while residual connections allow for deeper architectures and improved training efficiency. Normalization layers ensure stable and efficient learning, and 2D normalization is applied on both time and feature dimensions due to the presence of time-mixing and feature-mixing operations. A schematic of the neural architecture for TTM is presented in Figure 1.
Data: We obtained real world climate data from the ERA5-Land dataset [9]. Specifically, the data consists of hourly 2-meter temperature measurements for 2022 & 2023, at geographical locations corresponding to four large cities in the United States (Miami, New York, Los Angles and Seattle), see Appendix A for further details. This resulted in 17522 time steps for each time series data.
Fine-tuning: We restored the pre-trained model in our codebase and re-trained just the head decoder which is usually smaller compared to the backbone part of the architecture. We consider two pre-trained TTM models, the 512-96 (512 context window length, 96 prediction length) and the 1024-96 variant for finetuning. The head decoder for the 512-96 and 1024-96 model variants have approximately $290k$ and $390k$ parameters respectively. For the finetuning we used 80% of the data and 20% for testing. To obtain the best performance for the finetuning we use a Sequential Model-Based Optimization (SMBO) framework where we optimize the few-shot percentage, dropout and head dropout to minimize the evaluation loss. All other hyperparamaters are kept fixed and we use the AdamW optimizer for all tests which were run on a single NVIDIA V100 GPU. The hyperparameter definitions and value ranges are presented in Table 1 and the optimized results for the global finetuning in Table 3 with the corresponding Mean Squared Error (MSE) for both the fine-tuned and standard pre-trained models. For further details on the data, training methods, hyperparameter selection, and model performance results, see Appendix A.

3 Results

Figures 2 and 3 show the best and worst fitting forecasts, based on the forecasted MSE, for each city using our fine-tuned 1024-96 and 512-96 TTM models, respectively. To facilitate visualization, we plot the time series using a context length that is twice the size of the prediction length, thus the vertical red line is at a Time of 192 which is twice the prediction length of 96. Additionally, to accommodate the different temperature scales across cities, the temperature variable θ is normalized as per the method in the first step of the TTM architecture (see Figure 1). A notable characteristic observed from both the 512-96 and 1024-96 model forecasts is the significant variation in prediction accuracy across different cases. For the best-fitting forecasts of the 1024-96 model shown in the top panel of Figure 2, both the magnitude and periodicity are well predicted by our TTM model, particularly

Table 1: Hyperparameter definition and values used in our hyperparameter optimization framework.

Hyperparameter	Description	Values
`fewshot`	Fraction of data used for few-shot fine-tuning	5% - 20%
`head_dropout`	Dropout rate applied to the model's head	0.2 - 0.9
`dropout`	General dropout rate applied throughout the model	0.2 - 0.9
`learning_rate`	Learning rate for the AdamW optimizer	0.001
`batch_size`	Batch size for training and evaluation	64
`num_epochs`	Maximum number of epochs for training	100
`freeze_backbone`	Whether to freeze the backbone during fine-tuning	`True`
`context_length`	Length of input sequence	512, 1024
`forecast_length`	Length of forecasted output sequence	96
`n_trials`	Number of trials for the hypersearch	100

for Miami, New York, and Seattle. Even when the amplitude and periodicity characteristics are not obviously clear in the context window, as in the case for Los Angeles, the model still yields an accurate forecast compared to the real values. Conversely if we consider the worst fitting forecasts for each city, bottom sub-panels, we observe even when there is a clear periodicity in the dynamics present in the context window as in the case of Miami, Seattle and Los Angeles the model appears to somewhat capture this periodicity but fails to appropriately capture the magnitude resulting in substantial deviations from the actual values. A similar characteristic is observed for the predicted forecasts of the 512-96 model in Figure 3 which yields very accurate best-fitting forecasts but the worst-fitting forecasts, while seem to pick up some level of periodicity, completely fail at capturing the real amplitude. The reason for this discrepancy is not immediately apparent, but we hypothesize that the data used in the model's pre-training is likely a contributing factor. Additionally, given the use of hourly climate data, the data distribution is primarily influenced by periodic dynamics with a 24-hour frequency, as reflected in our observations. We hypothesize that this inherent daily pattern influences the forecasts, leading the model to produce predictions that retain this periodicity even when it is not present in the actual data. This discrepancy can lead to substantially divergent forecasts, especially when the true data lacks the expected periodic behavior, see the bottom sub-panel of the upper right panel in Figure 3 for an example. Additional forecasts showing this characteristic can be seen in Appendix C. Although there are discrepancies between the results, it is interesting to see that the fine-tuned model is able to capture non-obvious patterns, as those seen for the best results for Seattle and Los Angeles, in which regions with irregular behavior are followed by periodic or nearly periodic intervals. For further details on the behavior of the TTM models, see Appendix B.

4 Conclusion

In this study, we investigated the performance of the recently developed 'small' time series-based MLP-Mixer FM TTM, in accurately forecasting real-world dynamical systems within practical resource and cost constraints, evaluating their ability to generalize to new problems in both zero-shot and fine-tuned scenarios. We found that the compact size of the pre-trained MLP-Mixer models used in this study enabled us to implement a novel hyperparameter optimized fine-tuning and inference pipeline with modest hardware and computational resources. We find a substantial variation in the predicted forecasts from the fine-tuned TTM models: in some cases, the models capture the amplitude and periodicity within the context window extremely well, while in others, they appear less sensitive to the periodicity and fail to accurately predict the amplitude, resulting in significant forecasting errors. The origin of this 'hallucination' like behaviour is not clear and to our knowledge has not been reported on. We hypothesize that there could be at least two reasons for this behaviour. Firstly, the data used for pre-training the TTM. Secondly, as we are using hourly climate data, the data distribution is predominantly characterized by periodic dynamics with a 24-hour frequency. This inherent periodic pattern contributes to the periodic nature of the predicted forecasts, causing the model to retain this characteristic even when it is absent in the actual data. This results in significantly divergent forecasts, particularly when true data does not exhibit the expected periodicity as observed in many of our predicted forecasts. The origin of this behavior is further discussed in Appendix B and will be explored in an upcoming study. This study highlights that while TTMs can capture dominant periodic dynamics in the data, they also inherit these patterns even when absent in true observations, leading to notable forecast deviations suggesting the need for further model refinement to better adapt

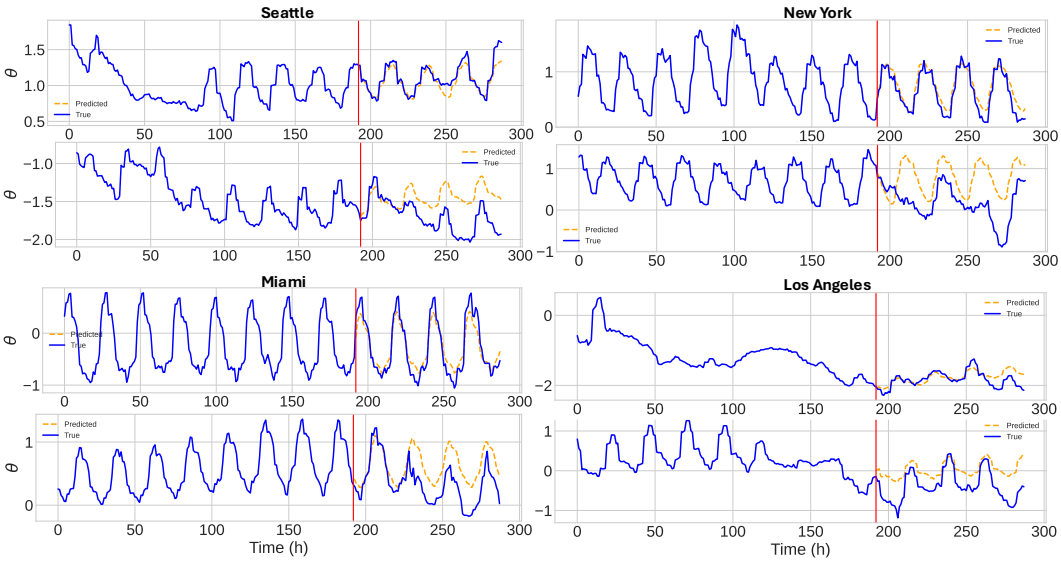


Figure 2: Predicted forecasts by our finetuned 1024-96 TTM model for the four cities: Seattle (upper left), New York (upper right), Miami (lower left), Los Angeles (lower right). The best- and worst-fitting forecasts are given by the upper and lower sub panels respectively. The vertical red line is when the predicted forecasts begin at Time of 192.

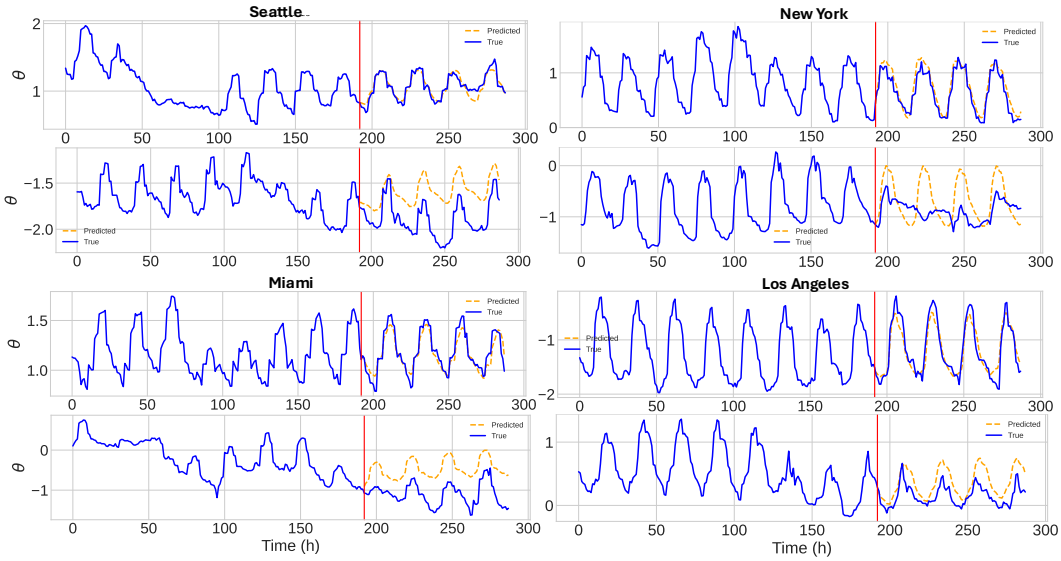


Figure 3: Predicted forecasts by our finetuned 512-96 TTM model for the four cities: Seattle (upper left), New York (upper right), Miami (lower left), Los Angeles (lower right). The best- and worst-fitting forecasts are given by the upper and lower sub panels respectively. The vertical red line is when the predicted forecasts begin at Time of 192.

to varying data characteristics. **Limitations:** One of the limitations with the current framework is the fixed context length and prediction window required by both the model and fine-tuning, which would need to be adapted to be applied to domain specific SciML scenarios which typically have variable datasize and forecast length requirements. Another limitation not addressed in this study is the lack of inclusion of exogenous variables that could influence target predictions. These variables, with known or estimated values throughout the forecast horizon, can provide valuable context to the model. For instance, in the climate forecasting case presented, variables from the ERA5 dataset, such as surface pressure, boundary layer height, and wind speed, could be incorporated. Including these additional features would likely enhance the model's forecasting capabilities post-finetuning.

References

[1] 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.

[2] Rishi Bommasani, Drew A Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, et al. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*, 2021.

[3] Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. *Advances in neural information processing systems*, 31, 2018.

[4] Si-An Chen, Chun-Liang Li, Nate Yoder, Sercan O. Arik, and Tomas Pfister. TSMixer: An All-MLP architecture for time series forecasting, 2023.

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

[6] Vijay Ekambaram, Arindam Jati, Nam H Nguyen, Pankaj Dayama, Chandra Reddy, Wesley M Gifford, and Jayant Kalagnanam. TTMs: Fast multi-level tiny time mixers for improved zero-shot and few-shot forecasting of multivariate time series. *arXiv preprint arXiv:2401.03955*, 2024.

[7] Mononito Goswami, Konrad Szafer, Arjun Choudhry, Yifu Cai, Shuo Li, and Artur Dubrawski. Moment: A family of open time-series foundation models. *arXiv preprint arXiv:2402.03885*, 2024.

[8] Lu Lu, Pengzhan Jin, Guofei Pang, Zhongqiang Zhang, and George Em Karniadakis. Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators. *Nature machine intelligence*, 3(3):218–229, 2021.

[9] J. Muñoz Sabater. ERA5-land hourly data from 1950 to present, 2019. Accessed on 26-Jul-2024.

[10] Imran Nasim and Joaõ Lucas de Sousa Almeida. Towards foundation models for the industrial forecasting of chemical kinetics. *arXiv preprint arXiv:2408.10720*, 2024.

[11] Maziar Raissi, Paris Perdikaris, and George E Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational physics*, 378:686–707, 2019.

[12] Kashif Rasul, Arjun Ashok, Andrew Robert Williams, Arian Khorasani, George Adamopoulos, Rishika Bhagwatkar, Marin Biloš, Hena Ghonia, Nadhir Vincent Hassen, Anderson Schneider, et al. Lag-llama: Towards foundation models for time series forecasting. *arXiv preprint arXiv:2310.08278*, 2023.

[13] Vincent Sitzmann, Julien Martel, Alexander Bergman, David Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. *Advances in neural information processing systems*, 33:7462–7473, 2020.

[14] Shashank Subramanian, Peter Harrington, Kurt Keutzer, Wahid Bhimji, Dmitriy Morozov, Michael W Mahoney, and Amir Gholami. Towards foundation models for scientific machine learning: Characterizing scaling and transfer behavior. *Advances in Neural Information Processing Systems*, 36, 2024.

[15] Matthew Tancik, Pratul Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains. *Advances in neural information processing systems*, 33:7537–7547, 2020.

[16] Ce Zhou, Qian Li, Chen Li, Jun Yu, Yixin Liu, Guangjing Wang, Kai Zhang, Cheng Ji, Qiben Yan, Lifang He, et al. A comprehensive survey on pretrained foundation models: A history from bert to chatgpt. *arXiv preprint arXiv:2302.09419*, 2023.

A Appendix: Data, Training, Hyperparameter Selection, and Model Performance results

Climate data details: The longitude and latitude coordinates used to acquire the temperature data is presented in Table 2.

Table 2: Coordinates of Selected Cities

City	Latitude	Longitude
New York, NY	40.7128° N	74.0060° W
Seattle, WA	47.6062° N	122.3321° W
Miami, FL	25.7617° N	80.1918° W
Los Angeles, CA	34.0522° N	118.2437° W

Training and Hyperparameter optimization details:

Table 3: Best metric and hyperparameter values obtained from our optimization framework used in the 512-96 and 1024-96 model finetuning.

Context Length	MSE	Fewshot %	Head Dropout	Global Dropout	MSE (Zero shot)
512	0.0921	5	0.815	0.482	0.0935
1024	0.1002	6	0.773	0.492	0.1008

We employed OPTUNA for hyperparameter optimization, leveraging its Sequential Model-Based Optimization (SMBO) approach, specifically the Tree-structured Parzen Estimator (TPE), to efficiently explore the hyperparameter space. The objective was to minimize the evaluation loss on the time series forecasting task. We used the optimizer AdamW, combined with an early-stopping mechanism with patience of 10 epochs without improvement higher than 0. The learning rate is updated by an OneCycleLR scheme with updating step equal to the number of batches in the train dataset. All the tests were performed using a single NVIDIA V100 GPU and took less than 20 s to be finished. An additional hyperparamter not included here is the possibility of updating or not the backbone. It was not included in our experiments as we observed no significant effect over the results.
To further investigate the difference in forecast accuracy between the base and fine-tuned models, we computed the mean and standard deviation values from the test batches. A total of 96 batches were used to derive the statistics presented in Table 4. For the fine-tuned models, we applied the best-fitting hyperparameter values obtained through our hyperparameter optimization framework. We employed the relative Mean Squared Error ($\mathrm{MSE_{rel}}$) as the evaluation metric for this analysis, which is computed as follows:

$$\mathrm{MSE_{rel}} = \frac{\left(\sum_{\mathrm{i=1}}^{\mathrm{n(t)}} \left(\mathrm{Y_i} - \hat{\mathrm{Y}}_\mathrm{i}\right)^2\right)^{1/2}}{\left(\sum_{\mathrm{i=1}}^{\mathrm{n(t)}} \left(\mathrm{Y_i}\right)^2\right)^{1/2}} \tag{1}$$

where Y_i represents the true value and $\hat{Y_i}$ represents the predicted value for each data point i within the test set. The metric normalizes the error by the magnitude of the true values, thus providing a relative measure of prediction accuracy.

Table 4: Performance Metrics for Zero-shot and Fine-tuned Models

Model variant	Training Type	Mean	Standard Deviation
512-96	Zeroshot	0.4087	0.2756
	Fewshot (best)	0.3873	0.2353
1024-96	Zeroshot	0.4235	0.2769
	Fewshot (best)	0.4106	0.2581

B Appendix: Additional Analysis of TTM Model Observations

B.1 Periodic 'Hallucination' in Forecasting Behavior

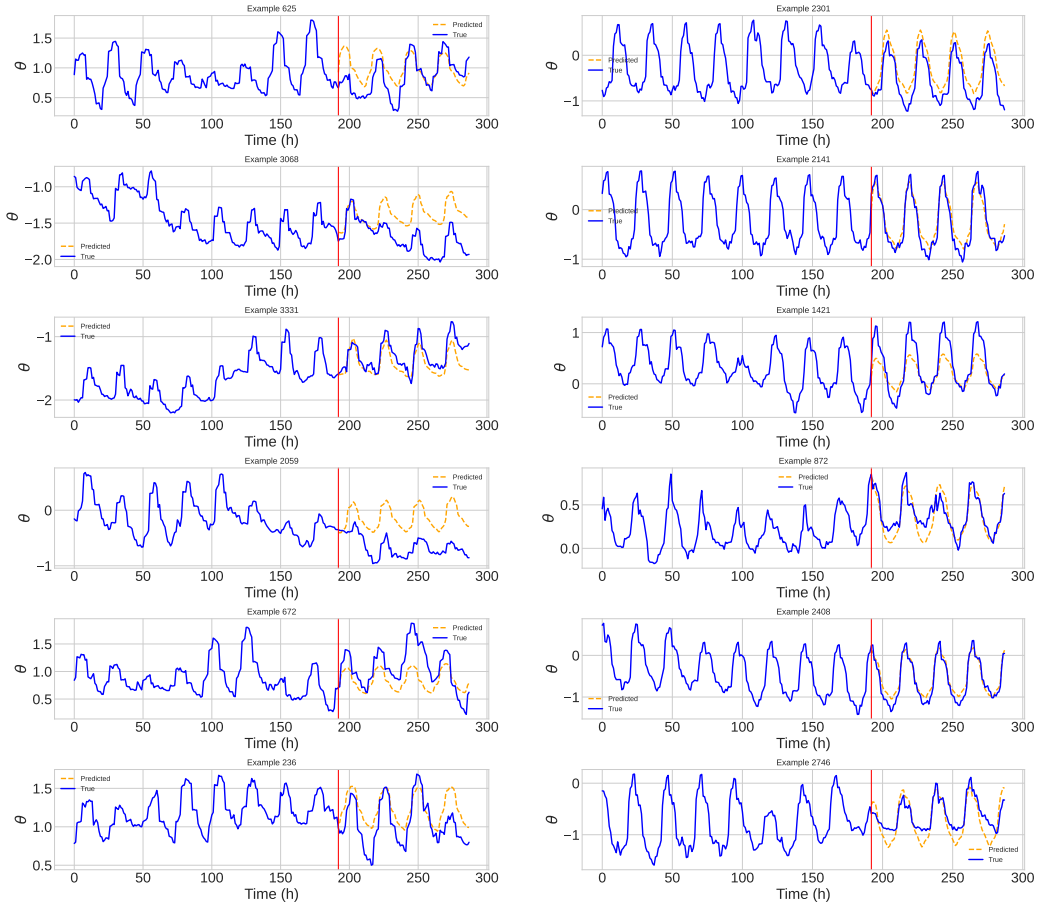
We note in the main text that the TTM model exhibits a tendency to "hallucinate" periodic behavior in some instances. We attribute this phenomenon to two primary factors: (1) the nature of the pre-training data, which may implicitly encode periodic patterns, and (2) the periodic characteristics present in the fine-tuning datasets. Regarding (1), the datasets used to pre-train the TTM model are described in detail in both the original paper [6] and on the dataset repository at `https://huggingface.co/ibm-granite/granite-timeseries-ttm-v1#training-data`. While it is difficult to definitively state that all pre-training datasets exhibit periodic trends, we hypothesize that datasets such as 'Australian Weather,' 'Australian Electricity Demand,' 'Sunspots,' and 'Saugeen River Flow' show significant periodic behavior. This assumption is reasonable, given that these datasets represent natural phenomena, which frequently exhibit cyclical trends. Regarding (2), the fine-tuning data consists of hourly temperature data, which clearly contains periodic characteristics due to daily and seasonal cycles. This likely reinforces the model's inclination to generate periodic patterns.

B.2 Limitations of TTM in Handling High-Frequency Data and Potential Improvements

Another characteristic we observe is that the temperature curves learned by TTM tend to be 'smoothened' compared to the actual curves. We note that this could be a possible limitation of the TTM framework on high-frequency data which suggests the potential to add Fourier features to this architecture on this problem.

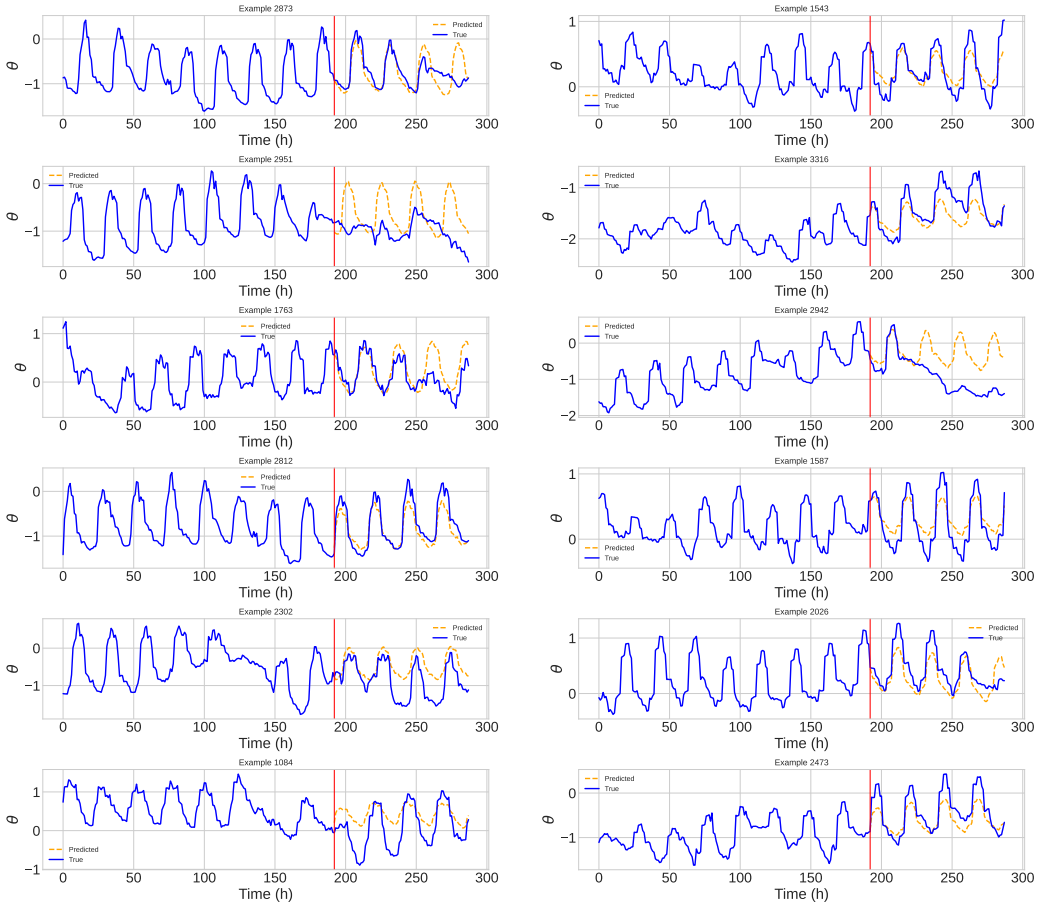
The original foundation model was trained using conventional activation functions, specifically combinations of Linear and Softmax layers. During fine-tuning, we are somewhat constrained by the model architecture, as we retain the pre-trained encoder as a backbone and only train a simple linear decoder. As a result, this setup may not be optimal for capturing multiscale or high-frequency patterns in the data. Techniques such as Fourier Features [15] or SIREN [13] could potentially enhance the model's ability to handle such data. To address this, two possible approaches could be explored: (1) training a new foundation model from scratch that incorporates Fourier-based features, which would significantly enhance the current architecture's ability to represent high-frequency components, or (2) designing a new type of decoder that utilizes these features. While the second approach may provide some improvements, it may still be limiting in terms of the overall representation learning capacity when compared to a full architectural overhaul.

C Appendix: Additional forecasting plots



(a) (b)

Figure 4: More examples for the 512 context length model. a) Seattle, b) Miami



(a) (b)

Figure 5: More examples for the 1024 context length model. a) New York, b) Los Angeles