MORPHEUS : A Foundation Model for Multivariate Time Series Forecasting

Anonymous Authors[1]

Abstract

Multivariate time series are vital for capturing complex interactions among variables in a number of domains including finance, e-commerce, and climate science. Research in this space is predominantly focused on developing bespoke models tailored to specific tasks, with relatively little exploration into foundation models that offer universal forecasting capabilities. We address this gap with two contributions: a novel framework that adapts traditional tokenization techniques to multivariate time series, integrating multiple target and feature series into a unified model allowing us to leverage existing foundation models for language; and an innovative synthetic data generation process to overcome data scarcity, enabling robust model training. Our approach handles diverse covariates and is validated through extensive experiments, demonstrating superior performance over current state-of-the-art methods.

1. Introduction

Time series forecasting is a fundamental task in various domains, including finance, economics, meteorology, and supply-chain management (Chatfield, 2000; De Gooijer & Hyndman, 2006; Lim & Zohren, 2021) where accurate predictions can drive critical decision-making processes.

The literature on time-series forecasting can broadly be divided in two: *univariate*, and *multivariate* forecasting. The former aims to predict future values of a single sequence based solely on its historical observations. More precisely, given a time series $\mathbf{y}_{1:c} = (y_1, \ldots, y_c)$ of real-valued, uniformly-spaced observations $y_t,\ t \in [c]$ of a quantity of interest, the objective is to forecast its future values $\mathbf{y}_{c+1:c+h} = (y_{c+1}, \ldots, y_{c+h})$ over a desired forecasting horizon h. For this univariate forecasting task, a variety of approaches have been proposed, ranging from statistical methods such as ARIMA (Box et al., 2015), ETS (Hyndman et al., 2002; 2008b;a), and Theta (Assimakopoulos & Nikolopoulos, 2000), to more modern deep-learning based approaches based on RNNs (Zhao et al., 2017; Lai et al., 2018), CNNs (Bai et al., 2018; Hewage et al., 2020; Wang et al., 2023), and MLPs (Oreshkin et al., 2019; Zeng et al., 2023). While successful, these models are of limited use in scenarios requiring the simultaneous consideration of multiple interdependent variables. The latter *multivariate* forecasting case involves multiple sequences of data points over time, providing a richer representation by capturing interactions among various quantities of interest. Formally, given a time series $\mathbf{Y}_{1:c} = (\mathbf{y}_1, \ldots, \mathbf{y}_c)$ of real-valued, uniformly-spaced observations $\mathbf{y}_t = (y_{1,t}, \ldots, y_{n,t}),\ t \in [c]$ of n quantities of interest (endogenous variables), as well as an auxiliary time series $\mathbf{X}_{1:c+h} = (\mathbf{x}_1, \ldots, \mathbf{x}_{c+h})$ corresponding to observations of m *exogenous features* $\mathbf{x}_t = (x_{1,t}, \ldots, x_{m,t}),\ t \in [c+h]$ with *known future values* over the desired forecasting horizon h, the objective is to forecast the future values of the endogenous variables $\mathbf{Y}_{c+1:c+h} = (\mathbf{y}_{c+1}, \ldots, \mathbf{y}_{c+h})$ over the forecasting horizon. While there is a growing body of work aimed at addressing this more general problem, (Zhou et al., 2021; 2022; Wu et al., 2021; Liu et al., 2023; Lim et al., 2021; Salinas et al., 2020; Das et al., 2023; Wang et al., 2024b;a; Zhang & Yan, 2023; Woo et al., 2022), it is largely focused on developing bespoke, application-specific models that need to be trained from scratch for the task at hand. This limits their applicability as such custom models are prone to overfitting, particularly in domains such as finance where the prevalent data are small and noisy.

The development of foundation models for universal forecasting—large models pretrained on diverse data for various tasks, adaptable with minimal fine-tuning—has transformed fields like natural language processing and computer vision. However, their application to multivariate forecasting remains underexplored. Notable efforts include LLMTime (Gruver et al., 2024) and Time-LLM (Jin et al., 2024) which utilize pretrained LLMs with adapters or novel tokenization techniques to leverage the learned knowledge of LLMs and adapt their output to time series modeling. More recent approaches such as TimesFM (Das et al., 2024) and Chronos (Ansari et al., 2024), employ stacked transformer architectures specifically for time series data but remain limited

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

to univariate forecasting due to their reliance on existing language model architectures in a blackbox manner.

A key hurdle in this area is that most foundation models are based on language models, which inherently operate on single sequences. This creates a challenge in adapting them for multivariate time series, as it is unclear how to effectively flatten multiple interdependent variables into a single sequence while maintaining the integrity of the transformer's attention mechanism. MOIRAI (Woo et al., 2024) attempts to address this by concatenating multiple variates into a single series and using a permutation invariant attention mechanism, but our experiments indicate limitations in this approach for multivariate forecasting, suggesting the need for alternative methodologies.

Another significant challenge is the scarcity of large-scale multivariate time series datasets, essential for training robust foundation models. Unlike the abundance of language data, multivariate time series data is often limited, fragmented, and domain-specific, posing a barrier to developing and fine-tuning large foundation models (Woo et al., 2024; Das et al., 2024; Ansari et al., 2024).

This work aims at addressing these two challenges through intuitive yet performant ideas, which we describe below.

2. Methodology

We address these aforementioned challenges by proposing a simple yet effective scheme to flatten multidimensional time series data through interleaving of covariates with special separator tokens, enabling the use of existing language architectures in a near blackbox manner. Additionally, we introduce a novel synthetic data generation process to augment real multivariate time series data, facilitating the pretraining of our foundation model. We detail our methodology below.

2.1. Modeling

We introduce a versatile "interleaving" method to effectively manage high-dimensional multivariate data that is compatible with any sequence-to-sequence architecture. Given a target series $\mathbf{Y}_{1:c}$ of past observations for n endogenous covariates and an auxiliary series $\mathbf{X}_{1:c+h}$ of past and future observations for m exogenous covariates, we construct an interleaved series comprising all $((n+m) \times c + m \times h)$ variables to predict $\mathbf{Y}_{c+1:c+h}$. As shown in Figure 1, the interleaved series (I) starts with a Feature Separator (FS) token, followed by the earliest observation $\mathbf{x}_1$ of the exogenous features, a Target Separator (TS), and then the earliest observation $\mathbf{y}_1$ of the target endogenous series. The order of exogenous and endogenous variables is arbitrary but consistent. This pattern is repeated for all time points in the historical context up to the first point in the prediction window ($c+1$), where interleaving ends after the Target Separator, as the endogenous values beyond this point are unknown and need forecasting.

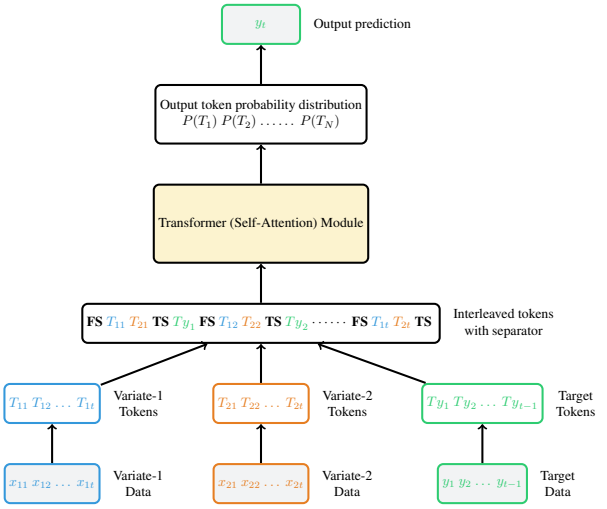

Figure 1. Architecture for MORPHEUS illustrating a case with 2 exogenous features and 1 endogenous target. The feature and target series are discretized to map real-valued observations into model tokens, and then interleaved using Feature and Target Separators (FS, TS). The interleaved series is passed through a stack of T5 transformer blocks to produce the next token, which is then mapped back to a real value which is our forecast.

This interleaving method ensures that more recent observations are positioned closer to the end of the context, which is particularly important for transformers, known to focus more on the end of the context. Unlike stacking covariate embeddings, altering the context length from c to $c+l$ during interleaving does not affect the relative position of tokens in the existing context with respect to the target token. This allows us to train models to be more robust against variations in context length. This setup simplifies the task to next-token prediction, allowing the interleaved sequence to be input into any sequence-to-sequence model to forecast endogenous values, one time point at a time, until the entire prediction window is covered.

While this approach can be integrated with any sequence-to-sequence architecture, we apply our method to the Chronos T5 model (Ansari et al., 2024), a model originally designed for univariate forecasting to extend its capabilities to the multivariate case. Our choice of Chronos is motivated by the following factors: (a) It has demonstrated state-of-the-art performance across numerous open-source and proprietary benchmarks; (b) it employs a transformer-based architecture, which is extensively studied and well-understood in the literature; and (c) it operates without assumptions regarding the distribution of output tokens and does not alter its pipeline based on data characteristics such as frequency (e.g., hourly, daily).

Our choice of this underlying "base" model necessitates another key step - discretizing and tokenizing our underlying real valued multivariate time series, since T5 is fundamentally a language model architecture for predicting tokens from a finite vocabulary. We do so by quantizing our interleaved real-valued series into 4,094 bins ranging from -15 to 15, and tokenizing the bins. Special tokens are assigned to the Feature Separator and Target Separator, with token numbers 0 and 1, respectively. The remaining 4,092 tokens are sequentially allocated to the bins within the specified range to generate token numbers. This tokenized series is fed into the T5 transformer, which is trained to predict a probability distribution over potential output tokens. During inference, we sample a desired number of output tokens from the predicted distribution and use their median as our final point forecast. While we use the T5 family, this method can be easily adapted to other next-token prediction models with minimal adjustments to the tokenization strategy.

2.2. Synthetic Data Augmentation

To address data scarcity, we propose a novel multivariate synthetic data generation procedure, which we use to augment real timeseries data, the combination which is then used to pretrain our foundation model. Similar ideas have been proposed before in (Ansari et al., 2024; Dooley et al., 2024; Das et al., 2024). This is important since multivariate time series are particularly scarce: even within the relatively large time series data corpus LoTsa (Woo et al., 2024), less than 2% contain covariates.

Our synthetic data generation approach is based on the hypothesis that time series are realizations of an underlying Hidden Markov Model with causal links to observed variables. We generate synthetic data by sampling this hidden time series for each feature-target pair, ensuring all feature and target series are partial or total realizations of this hidden series.

Building upon the synthetic data generation algorithm in (Ansari et al., 2024), we sample up to M kernels from an underlying kernel bank, per feature-target tuple. This serves as the hidden Markov state for the tuple. Next, each feature and target series are generated by sampling up to N kernels from the above-sampled M kernels, independently with replacement. Depending on choice of (M, N, K), we can produce targets with lot of common kernels with features or almost independent of features.This is directly comparable to real-world data where target and feature series can rely on the same 'hidden' variable, like electricity usage across homes (known) can rely on the outside weather (unknown), or cases where targets are only partially reliant on the associated features, like people with a cold (known) could rely on outside temperature (unknown) or be reliant on other factors like a flu outbreak. We call this the MySTiC A dataset.

We augment our MySTiC A dataset, with a complimentary approach, MySTic B, where features are made by sampling up to 3 kernels from the original kernel bank, and the target is produced as a weighted sum of polynomial functions of these generated features. This produces synthetic features with higher correlation than MySTiC A.

3. Training, Evaluation and Results

3.1. Training

Our model training consists of two steps: pre-training on a synthetic dataset via the MySTiC algorithm and fine-tuning on real-world and synthetic datasets. The MORPHEUS model uses a T5-tiny transformer with 8 million parameters.

Pre-training uses MySTiC A and B datasets at a 2:1 ratio to acclimate the model to interleaved multivariate data and diverse input types. Fine-tuning involves 10% synthetic and 90% real-world data, with real-world sampling based on size, ensuring a minimum 2% for small datasets to prevent extreme sub-sampling. Additional details on datasets used for model training and evaluation, training hyperparameters are in Table 3, Table 4 and Table 5.

3.2. Evaluation

We extensively evaluate MORPHEUS and compare it against other open-source time series foundation models (Chronos, MOIRAI), and several bespoke multivariate time series models from recent literature. We focus on the forecasting problem i.e. predicting the endogenous target y_{t+1} for the next step using its historical values $\mathbf{y}_{t-c:t}$, and known (both historical and future) exogenous covariate values $\mathbf{X}_{t-c:t+1}$, for time series from a wide variety of application domains.

Using a fixed context length of 256 time periods, we report the Mean Absolute Scaled Error (MASE), calculated as the ratio of the Mean Absolute Error (MAE) of the model to the MAE of a naive model predicting the last observed target value. Except for foundational models (Chronos, MOIRAI, MORPHEUS), models are trained specifically for each dataset and table combination. For instance, to report MASE for the 2-feature US Treasury dataset using a Vanilla NN Model, which contains 50 tables, we train 50 individual models, one per table, and report the mean MASE for over all tables. Additional details on data splits and "in-domain" vs. "zero-shot" datasets are in Table 3 and Table 4. Additional details regarding training these non-foundation models is specified in Appendix E

3.3. Results

The results in Table 1 indicate that MORPHEUS consistently achieves the lowest average MASE across diverse

		MySTiC-B	MySTiC-A	US Treasury	GB Treasury	Etth1	Etth2	Ettm1	Ettm2	Electricity	Traffic	Weather	Illness	All
2 Features	Chronos	0.43	0.30	1.09	1.21	1.01	0.60	1.08	0.73	**0.74**	0.51	1.04	1.06	0.82
	MOIRAI	0.83	0.77	1.23	1.67	**0.96**	0.95	1.14	1.13	0.80	0.57	1.19	1.22	1.04
	MORPHEUS	**0.25**	**0.27**	0.58	**0.68**	1.02	0.54	1.06	0.68	0.77	0.51	**1.00**	1.19	**0.71**
	DLinear	2.33	5.46	1.08	14.01	1.00	0.69	**1.00**	0.64	0.85	0.75	1.19	1.93	2.58
	LightTS	6.08	0.97	1.67	4.48	1.10	0.84	1.06	0.90	0.97	0.77	1.41	1.87	1.84
	Non-stationary Transformer	0.63	1.24	0.66	1.25	0.98	0.54	1.03	0.64	1.03	0.56	1.06	1.31	0.91
	TFT	1.42	1.57	0.77	1.46	0.97	**0.45**	1.02	0.57	0.90	0.50	1.02	2.00	1.05
	TiDE	1.54	1.57	1.10	3.62	1.03	0.81	**1.00**	0.67	0.91	0.89	1.26	1.86	1.35
	TimeMixer	0.82	0.74	1.02	1.48	0.99	0.52	1.01	0.60	0.91	0.61	1.09	1.70	0.96
	TimeXer	0.34	0.49	0.59	0.51	**0.96**	0.48	**1.00**	0.63	0.89	0.52	1.05	**1.01**	**0.71**
	Vanilla NN	0.39	0.57	**0.55**	0.96	**0.96**	0.51	1.01	0.61	0.92	0.52	1.05	1.06	0.76
	iTransformer	0.32	0.49	0.56	0.84	0.97	**0.45**	1.01	**0.59**	0.88	**0.49**	1.09	**1.01**	0.72
5 Features	Chronos	0.60	**0.24**	1.09	-	1.01	0.60	1.07	0.73	**0.74**	0.51	1.04	1.07	0.79
	MOIRAI	0.75	0.75	1.22	-	**0.92**	0.98	1.11	1.10	0.76	0.59	1.19	1.24	0.96
	MORPHEUS	**0.22**	0.30	0.50	-	1.04	0.57	1.09	0.70	0.77	0.45	**0.99**	1.39	0.73
	DLinear	2.07	2.43	1.07	-	1.00	0.69	**1.00**	0.64	0.85	0.74	1.19	1.93	1.24
	LightTS	1.04	1.28	1.70	-	1.12	0.87	1.07	0.79	0.98	0.76	1.29	1.88	1.16
	Non-stationary Transformer	0.70	1.32	0.59	-	0.99	0.57	1.03	0.66	1.07	0.59	1.07	-	0.86
	TFT	0.81	1.35	0.63	-	0.97	0.46	1.01	**0.58**	0.91	0.50	1.02	1.68	0.90
	TiDE	1.40	1.26	1.09	-	1.03	0.80	**1.00**	0.66	0.91	0.88	1.25	1.98	1.11
	TimeMixer	0.76	0.72	1.03	-	0.99	**0.52**	1.01	0.60	0.91	0.62	1.10	1.27	0.87
	TimeXer	0.39	0.40	0.53	-	0.96	0.48	1.01	0.63	0.88	0.52	1.05	1.01	0.71
	Vanilla NN	0.39	0.47	**0.45**	-	0.95	0.52	1.01	0.61	0.90	0.50	1.09	**0.99**	0.72
	iTransformer	0.34	0.41	0.49	-	0.97	0.45	1.01	0.59	0.88	**0.48**	1.07	1.00	**0.70**

Table 1. MASE of different models across feature sets (lower is better; **bold** for best, underlined for second-best). MASE is ratio of the model MAE (Mean Absolute Error) to a naive model MAE. Naive model is a model which repeats the last available target value as a forecast. Note: GB Treasury dataset only has 2 features, hence the 5 feature metrics cannot be computed

datasets, either outperforming or matching the best benchmark models on both 2 and 5 feature datasets. TimeXer and iTransformer follow closely but these are trained specifically for each dataset and table combination.

On synthetic datasets generated with MySTiC algorithms, MORPHEUS excels, likely due to pre-training on similar data. Chronos ranks second, performing well on MySTiC-A due to kernel similarities with its training data, but struggles on MySTiC-B where targets have explicit linear and polynomial dependencies on features. For real-world datasets, two categories emerge:

1) Etth1, Ettm1, Weather, Illness: Most models fail to surpass the naive model, which repeats the last value, due to insufficient feature information or complex dependencies. MORPHEUS maintains a desirable MASE around 1 (except for Illness dataset). iTransformer and TimeXer are also doing very well defaulting to naive estimate as opposed to other models like LightTS, DLinear. Chronos handles these datasets effectively, maintaining a MASE around 1.

2) US & GB Treasury, Etth2, Ettm2, Electricity, Traffic: Benchmark models generally outperform the naive model, with MASE scores below 1. MORPHEUS is either the best or among the top performers across these datasets. TimeXer and iTransformer are also strong performers. Chronos performs well except on US and GB Treasury datasets, where feature information is crucial for predicting target variables. These datasets contain historical yield curve data, where predicting yields at each maturity in isolation is challenging, as evidenced by the performance of all models in the univariate prediction scenario in Table 6. The multivariate prediction problem becomes feasible when additional points on the curve are considered. MOIRAI is unable to outperform other models even though it is able to support covariates.

4. Conclusion

In this paper, we make progress towards developing a foundation model for multivariate forecasting by introducing a simple, yet effective interleaving strategy that integrates multiple target and feature series into a single unified model without any loss of information, allowing us to leverage existing foundation model frameworks designed for natural language. Additionally, we address the time series data scarcity problem through an innovative synthetic data generation process, ensuring effective and robust model training. These advancements enable the development of a new multivariate time series foundation model that offers predictive performance that is competitive with state-of-the-art bespoke models across diverse datasets. The techniques are adaptable to any sequence-to-sequence architecture, paving the way for a universal model capable of general multivariate forecasting.

Our work leaves open several avenues for future research, the most compelling ones being developing more sophisticated synthetic data generation processes that more closely mimic real world patterns, and applying our interleaving scheme to base models other than the T5 used in our work.

References

Ansari, A. F., Stella, L., Turkmen, C., Zhang, X., Mercado, P., Shen, H., Shchur, O., Rangapuram, S. S., Arango, S. P., Kapoor, S., Zschiegner, J., Maddix, D. C., Wang, H., Mahoney, M. W., Torkkola, K., Wilson, A. G., Bohlke-Schneider, M., and Wang, Y. Chronos: Learning the language of time series, 2024. URL https://arxiv.org/abs/2403.07815.

Assimakopoulos, V. and Nikolopoulos, K. The theta model: a decomposition approach to forecasting. *International journal of forecasting*, 16(4):521–530, 2000.

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

Box, G. E., Jenkins, G. M., Reinsel, G. C., and Ljung, G. M. *Time series analysis: forecasting and control*. John Wiley & Sons, 2015.

Chatfield, C. *Time-series forecasting*. Chapman and Hall/CRC, 2000.

Das, A., Kong, W., Leach, A., Mathur, S., Sen, R., and Yu, R. Long-term forecasting with tide: Time-series dense encoder. *arXiv preprint arXiv:2304.08424*, 2023.

Das, A., Kong, W., Sen, R., and Zhou, Y. A decoder-only foundation model for time-series forecasting, 2024. URL https://arxiv.org/abs/2310.10688.

De Gooijer, J. G. and Hyndman, R. J. 25 years of time series forecasting. *International journal of forecasting*, 22(3): 443–473, 2006.

Dooley, S., Khurana, G. S., Mohapatra, C., Naidu, S. V., and White, C. Forecastpfn: Synthetically-trained zero-shot forecasting. *Advances in Neural Information Processing Systems*, 36, 2024.

Goswami, M., Szafer, K., Choudhry, A., Cai, Y., Li, S., and Dubrawski, A. Moment: A family of open time-series foundation models, 2024. URL https://arxiv.org/abs/2402.03885.

Gruver, N., Finzi, M., Qiu, S., and Wilson, A. G. Large language models are zero-shot time series forecasters. *Advances in Neural Information Processing Systems*, 36, 2024.

Hewage, P., Behera, A., Trovati, M., Pereira, E., Ghahremani, M., Palmieri, F., and Liu, Y. Temporal convolutional neural (tcn) network for an effective weather forecasting using time-series data from the local weather station. *Soft Computing*, 24:16453–16482, 2020.

Hyndman, R., Koehler, A. B., Ord, J. K., and Snyder, R. D. *Forecasting with exponential smoothing: the state space approach*. Springer Science & Business Media, 2008a.

Hyndman, R. J., Koehler, A. B., Snyder, R. D., and Grose, S. A state space framework for automatic forecasting using exponential smoothing methods. *International Journal of forecasting*, 18(3):439–454, 2002.

Hyndman, R. J., Akram, M., and Archibald, B. C. The admissible parameter space for exponential smoothing models. *Annals of the Institute of Statistical Mathematics*, 60:407–426, 2008b.

Jin, M., Wang, S., Ma, L., Chu, Z., Zhang, J. Y., Shi, X., Chen, P.-Y., Liang, Y., Li, Y.-F., Pan, S., and Wen, Q. Time-llm: Time series forecasting by reprogramming large language models, 2024. URL https://arxiv.org/abs/2310.01728.

Lai, G., Chang, W.-C., Yang, Y., and Liu, H. Modeling long-and short-term temporal patterns with deep neural networks. In *The 41st international ACM SIGIR conference on research & development in information retrieval*, pp. 95–104, 2018.

Lim, B. and Zohren, S. Time-series forecasting with deep learning: a survey. *Philosophical Transactions of the Royal Society A*, 379(2194):20200209, 2021.

Lim, B., Arık, S. Ö., Loeff, N., and Pfister, T. Temporal fusion transformers for interpretable multi-horizon time series forecasting. *International Journal of Forecasting*, 37(4):1748–1764, 2021.

Liu, Y., Hu, T., Zhang, H., Wu, H., Wang, S., Ma, L., and Long, M. itransformer: Inverted transformers are effective for time series forecasting. *arXiv preprint arXiv:2310.06625*, 2023.

Oreshkin, B. N., Carpov, D., Chapados, N., and Bengio, Y. N-beats: Neural basis expansion analysis for interpretable time series forecasting. *arXiv preprint arXiv:1905.10437*, 2019.

Salinas, D., Flunkert, V., Gasthaus, J., and Januschowski, T. Deepar: Probabilistic forecasting with autoregressive recurrent networks. *International journal of forecasting*, 36(3):1181–1191, 2020.

Wang, H., Peng, J., Huang, F., Wang, J., Chen, J., and Xiao, Y. Micn: Multi-scale local and global context modeling for long-term series forecasting. In *The eleventh international conference on learning representations*, 2023.

Wang, S., Wu, H., Shi, X., Hu, T., Luo, H., Ma, L., Zhang, J. Y., and Zhou, J. Timemixer: Decomposable multiscale mixing for time series forecasting. *arXiv preprint arXiv:2405.14616*, 2024a.

Wang, Y., Wu, H., Dong, J., Qin, G., Zhang, H., Liu, Y., Qiu, Y., Wang, J., and Long, M. Timexer: Empowering transformers for time series forecasting with exogenous variables. *arXiv preprint arXiv:2402.19072*, 2024b.

Woo, G., Liu, C., Sahoo, D., Kumar, A., and Hoi, S. Etsformer: Exponential smoothing transformers for time-series forecasting. *arXiv preprint arXiv:2202.01381*, 2022.

Woo, G., Liu, C., Kumar, A., Xiong, C., Savarese, S., and Sahoo, D. Unified training of universal time series forecasting transformers, 2024. URL https://arxiv.org/abs/2402.02592.

Wu, H., Xu, J., Wang, J., and Long, M. Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. *Advances in neural information processing systems*, 34:22419–22430, 2021.

Zeng, A., Chen, M., Zhang, L., and Xu, Q. Are transformers effective for time series forecasting? In *Proceedings of the AAAI conference on artificial intelligence*, volume 37, pp. 11121–11128, 2023.

Zhang, Y. and Yan, J. Crossformer: Transformer utilizing cross-dimension dependency for multivariate time series forecasting. In *The eleventh international conference on learning representations*, 2023.

Zhao, Z., Chen, W., Wu, X., Chen, P. C., and Liu, J. Lstm network: a deep learning approach for short-term traffic forecast. *IET intelligent transport systems*, 11:68–75, 2017.

Zhou, H., Zhang, S., Peng, J., Zhang, S., Li, J., Xiong, H., and Zhang, W. Informer: Beyond efficient transformer for long sequence time-series forecasting. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pp. 11106–11115, 2021.

Zhou, T., Ma, Z., Wen, Q., Wang, X., Sun, L., and Jin, R. Fedformer: Frequency enhanced decomposed transformer for long-term series forecasting. In *International conference on machine learning*, pp. 27268–27286. PMLR, 2022.

A. Synthetic Data Generation Details

In this section, we provide formal algorithms that detail our synthetic data generation process described in Section 2.2, with algorithm 1 describing the generation of dataset MySTiC A, and algorithm 2 describing the generation of dataset MySTiC B. In this paper, the synthetic dataset MySTiC refers to the combination of MySTiC A and MySTiC B datasets. Additional details regarding these synthetic datasets are in Table 2

Algorithm 1 MySTiC A Series Generation

Input: KernelBank (size K), M, N, S
Output: Dataset
Dataset $\leftarrow \{\}$
for each i from 1 to S **do**
  $F \leftarrow \text{UniformRandom}(1, 11)$
  Tuple $\leftarrow \{\}$
  HiddenState $\leftarrow$ Sample(KernelBank, M)
  for each Feature in 1 to F **do**
    SampledKernels $\leftarrow$ Sample(HiddenState, N, with replacement)
    FeatureSeries $\leftarrow$ Combine(SampledKernels)
    Add FeatureSeries to Tuple
  end for
  SampledKernels $\leftarrow$ Sample(HiddenState, N, with replacement)
  TargetSeries $\leftarrow$ Combine(SampledKernels)
  Add TargetSeries to Tuple
  Add Tuple to Dataset
end for
Return Dataset

Algorithm 2 MySTiC B Series Generation

Input: KernelBank (size K), MaxFeatures (N), NumTuples (S)
Output: Dataset
Dataset $\leftarrow \{\}$
for each Tuple in 1 to S **do**
  $F \leftarrow \text{UniformRandom}(1, 11)$
  Features $\leftarrow \{\}$
  Weights $\leftarrow \{\}$
  for each FeatureIndex in 1 to 11 **do**
    Feature $\leftarrow$ Sample(KernelBank, N)
    Weight $\leftarrow$ SampleWeight()
    Features.append(Feature)
    Weights.append(Weight)
  end for
  $t \leftarrow 0$
  for each Feature in Features **do**
    Polynomial $\leftarrow$ MapToRandomPolynomial(Feature, MaxDegree d)
    $t \leftarrow t +$ Polynomial $\times$ Weight
  end for
  ConvolutionArray $\leftarrow$ Sample([-1, 1], 3)
  $T \leftarrow \text{Convolve}(t, \text{ConvolutionArray})$
  Dataset.append(T)
end for
Return Dataset

	MySTiC A	MySTiC B
Number of Tuples	250000	100000
Maximum Features	11	11
Length of Tuples	1000	1000
Weight in Final Dataset	2	1

Table 2. Comparison of MySTiC A and MySTiC B

B. Dataset Statistics

Dataset	Split	# Tables			# Observations		
		0 Features	2 Features	5 Features	0 Features	2 Features	5 Features
ETTh1	train	1	15	6	13,936	627,120	501,696
	test	1	10	6	3,484	104,520	125,424
ETTh2	train	1	15	6	13,936	627,120	501,696
	test	1	10	6	3,484	104,520	125,424
ETTm1	train	1	15	6	55,744	2,508,480	2,006,784
	test	1	10	6	13,936	418,080	501,696
ETTm2	train	1	15	6	55,744	2,508,480	2,006,784
	test	1	10	6	13,936	418,080	501,696
Electricity	train	1	1,500	750	21,043	94,693,500	94,693,500
	test	1	50	50	5,261	789,150	1,578,300
GB Treasury	train	1	1	-	261	783	-
	test	1	1	-	59	177	-
Illness	train	1	15	6	772	34,740	27,792
	test	1	15	6	194	8,730	6,984
Traffic	train	1	1,000	500	14,035	42,105,000	42,105,000
	test	1	50	50	3,509	526,350	1,052,700
US Treasury	train	6	168	336	45,012	3,781,008	15,124,032
	test	6	50	50	7,344	183,600	367,200
Weather	train	1	190	350	42,156	24,028,920	88,527,600
	test	1	50	50	10,540	1,581,000	3,162,000

Table 3. Statistics on open source datasets used for training and testing.

We have integrated several open-source datasets (Zhou et al., 2021; Goswami et al., 2024) into our training and testing pipeline. These datasets are primarily sampled daily across various domains, except for ETTh and ETTm, which are sampled at 1-hour and 15-minute intervals respectively.

To prepare these datasets, we processed the raw data to create train/test splits with varying numbers of features. For the US and GB Treasury datasets, we used data from `"2020-01-01"` onwards to form the test split. For all other datasets, the test split comprises the last 20% of the data. Within each dataset split, we further generate sub-datasets with different numbers of features by randomly sampling the feature and target columns.

To illustrate this process, consider the US Treasury dataset, which includes daily yields for US Treasuries from 1990 to 2024. We create train and test splits by splitting the data at `"2020-01-01"`. Each split contains daily yields for eight maturities: `3 M`, `6 Mo`, `1 Yr`, `2 Yr`, `3 Yr`, `5 Yr`, `7 Yr`, and `10 Yr`. To form a sub-dataset with two features, we randomly select two columns from the 8 available columns and then choose a target from the remaining six columns. This process is repeated to create multiple sub-datasets (referred to as "# Tables" in Table 3) for each dataset. The number of sub-datasets that can be created depends on the number of columns in the original dataset. For instance, the GB Treasury dataset has only three columns, allowing for only one sub-dataset with two features and none with five features.

Table 3 and Table 4 provide more information on the datasets used for model training and evaluation. Datasets are split into 2 and 5 feature sets by sampling features randomly. We divide datasets by time periods to create splits for "unseen" periods. For the US Treasury dataset, training data is up to December 31, 2019, and test data from January 1, 2020. A 2-feature test table is constructed by sampling 3 columns from the test split, using 2 as features and 1 as the target, repeated to create 50 test tables. An optional train dataset is created for fine-tuning, categorized as "in-domain" if used for fine-tuning, or "zero-shot" otherwise.

In our model testing, we exclusively use the target variable to evaluate performance. In the univariate scenario (i.e., when "# Features" = 0), there is typically only one target variable available for most datasets, resulting in "# Tables" = 1. For the US Treasury datasets, we considered all yields as potential targets, except for `3 M` and `10 Yr`. This choice reflects the common industry practice of interpolating yields based on surrounding points along the yield curve.

The last 25% of the "train" split is used as validation set in the MORPHEUS training process.

C. Datasets used for training & evaluation

The datasets used for traning and evaluation are classified into in-domain and zero-shot categories as shown in Table 4 below. In-domain datasets are used in the second stage of training process (as described in 3.1). Zero-shot datasets are only used for evaluation and to report metrics.

Dataset	**MORPHEUS**	**MOIRAI**	**Chronos**
US Treasury	In-domain	-	-
GB Treasury	Zero-shot	-	-
ETTh1	Zero-shot	Zero-shot	Zero-shot
ETTh2	Zero-shot	Zero-shot	Zero-shot
ETTm1	Zero-shot	Zero-shot	Zero-shot
ETTm2	Zero-shot	Zero-shot	Zero-shot
Traffic	Zero-shot	In-domain	Zero-shot
Electricity	In-domain	Zero-shot	In-domain
Weather	In-domain	In-domain	Zero-shot
Illness	In-domain	-	-

Table 4. Evaluation of datasets across different models.

D. MORPHEUS Training & evaluation parameters

The model training and evaluation described in 3.1 uses hyperparameters from the Table 5. For evaluation, models are limited to a fixed context length of 256 periods. Depending on their architecture, they may use varying amounts of data for inference. MORPHEUS for instance, is trained to handle context lengths up to 512 tokens. When incorporating feature data into the interleaving framework, the context size increases to 2560 tokens for a dataset with 2 features and 1 target, as each time period requires 5 tokens (2 features, 1 target, 1 TS, 1 FS). Truncating to the most recent 512 tokens reduces the effective context length to approximately 103 time periods, potentially impacting performance. Despite this limitation, we maintain a 256 context length for other models in our evaluations.

Parameter	Pre-training	Fine-tuning	Evaluation
Learning Rate	0.001	0.001	-
Number of Steps	100,000	20,000	-
Context Length	512	512	512
Prediction Length	4	4	1
Batch Size	256	256	256
Number of Samples	-	-	50

Table 5. Training and Evaluation Parameters

E. Benchmark model settings

In this section, we make a few clarifying remarks regarding model parameters chosen for training the bespoke multivariate forecasting models (i.e. non-foundation models) specified in Table 1. Fundamentally, these models, although they model cross variate dependencies, they do not support exogenous variables (i.e. variates whose value is known in the prediction window). In order to overcome this, we allow these models to "cheat" by peeking into the future for the exogenous variates. We do so by intentionally shifting all timeseries corresponding to exogenous variates back by duration equal to the length of the prediction window, effectively pushing the future information for these variates into the past context window. The second key modification we make is the length of the context window available to these models. While we technically allow a lookback window of length 256 time periods, in our experimental evaluation, we observed a substantial degradation in forecasting performance (in terms of MASE) for all these bespoke models. We therefore shrunk the lookback window to a much shorter length through preliminary hyperparameter tuning on a small random sample of datasets from our experimental setup. The ideal lookback windows were found to be pretty small - ranging from 12-20 in length. The lookback window size was then fixed for every model for the rest of the experiments. No additional hyperparameter tuning was performed for optimization or model architectures, and default parameters were used from their open-source implementations, which we obtained from https://github.com/thuml/Time-Series-Library.git.

F. Univariate Results

We compare the performance of various models in univariate prediction scenario, where the target is forecasted using only the past target data. The setup for each model is described below:

- **Linear Model**: For a given context, we fit an Ordinary Least Squares (OLS) model with an autoregressive (AR) lag of order 1 for all features and target variables. We perform a rolling forecast, fitting one model per context (past 256 time steps) and making a 1-step ahead prediction.
- **MOIRAI**: Using the pretrained model, we conduct zero-shot predictions with a context length of 256. We select the "small" variant of the MOIRAI model because its size, approximately 14 million parameters, is the closest to that of our pretrained MORPHEUS model.
- **Chronos**: For comparison, we include results from the "tiny" model variant. Since it only supports univariate predictions, we use the target variable and perform zero-shot predictions with a context length of 256. The "tiny" variant was selected to match the size of our pretrained MORPHEUS model.

	US Treasury	GB Treasury	ETTh1	ETTh2	ETTm1	ETTm2	Traffic	Electricity	Weather	Illness	All
Linear	**1.01**	**0.94**	**1.00**	1.01	**1.01**	1.06	1.02	1.02	<u>1.05</u>	1.11	1.02
Chronos	<u>1.07</u>	<u>1.21</u>	<u>1.01</u>	<u>0.61</u>	<u>1.07</u>	<u>0.73</u>	0.50	**0.74**	**1.03**	<u>1.06</u>	0.90
MORPHEUS	1.27	1.34	1.09	**0.55**	1.17	**0.69**	0.51	<u>0.78</u>	1.06	**1.05**	0.95
MOIRAI	1.21	1.39	1.19	0.80	1.31	1.21	**0.45**	0.83	1.24	1.18	1.08

Table 6. MASE scores of different models on univariate datasets (lower is better; **bold** for best, <u>underlined</u> for second-best)