DYNLMC: DYNAMIC LINEAR COREGIONALIZATION FOR REALISTIC SYNTHETIC MULTIVARIATE TIME SERIES

Anonymous authors
Paper under double-blind review

ABSTRACT

Synthetic data is essential for training foundation models for time series (FMTS), but most generators assume static correlations, missing realistic inter-channel dependencies. We introduce **DynLMC**, a Dynamic Linear Model of Coregionalization that incorporates time-varying, regime-switching correlations and cross-channel lag structures. This approach produces synthetic multivariate time series with correlation dynamics that closely resemble real data. Fine-tuning three foundational models on DynLMC-generated data yields consistent zero-shot forecasting improvements across nine benchmarks. Our results demonstrate that modeling dynamic correlations enhances FMTS transferability, highlighting the importance of data-centric pretraining.

Track: Research

1 INTRODUCTION

Foundation models for time series (FMTS) aim for broad generalization using large-scale data and flexible architectures. Unlike text or vision, time series data is highly domain-specific, and rarely available at large scale, particularly for realistic multivariate dynamics, making synthetic data pretraining a scalable alternative. Models like Chronos Ansari et al. (2024) and TimePFN Schlagenhauf et al. (2024) show that synthetic priors enable strong zero- and few-shot generalization, with the generator's quality and diversity driving transferability.

However, current synthetic generators rely on static correlation assumptions. Chronos uses univariate Gaussian process (GP) sampling, while TimePFN employs the Linear Model of Coregionalization (LMC) for multivariate data, assuming fixed, instantaneous correlations. This produces diverse samples but fails to capture evolving correlations, lagged dependencies, and regime shifts seen in real systems. Our analyses reveal that real datasets exhibit greater temporal variability in cross-channel relationships than LMC-generated data (Figure 1). Pretraining on static-correlation data can lead to rigid relational priors, limiting adaptability to real-world signals.

To address this, we introduce DynLMC, a Dynamic Linear Model of Coregionalization (DynLMC) that generalizes TimePFN's generator. DynLMC extends LMC by: (1) modeling smooth correlation drift via autoregressive updates, (2) introducing regime-switching correlations with a Hidden Markov Model, and (3) incorporating lagged cross-channel dependencies. These mechanisms generate realistic, nonstationary multivariate time series, better reflecting real-world domains like energy, finance, and climate (Figure 2).

Our main contributions are:

- We present DynLMC, a synthetic dataset generator for multivariate time series that captures evolving and lagged inter-channel dependencies, enabling realistic, nonstationary correlation structures.
- We empirically show that fine-tuning pretrained forecasting models on DynLMC improves robustness and generalization, yielding consistent gains on real-world test datasets.

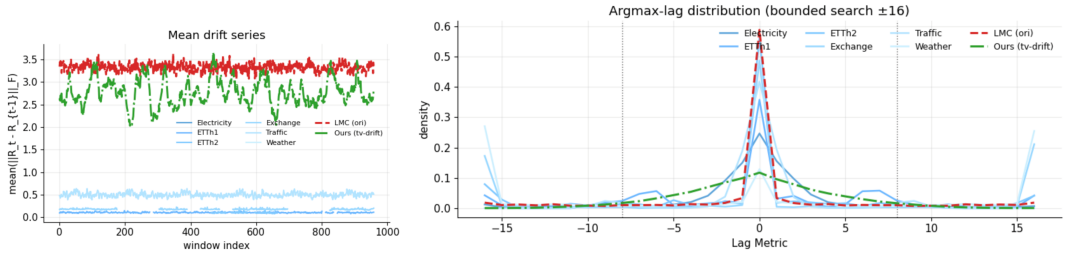

Figure 1: **Left:** Correlation drift comparison across real and synthetic datasets. **Right:** Argmax-lag distribution comparison across real-world and synthetic datasets.

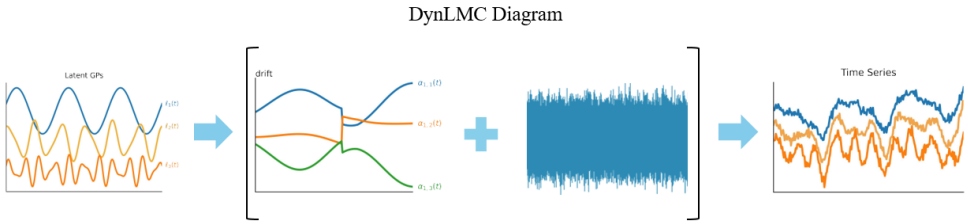

Figure 2: An illustration of DynLMC with evolving inter-channel structures (drift).

2 Background

Notation and Preliminaries. Let $\mathcal{D} := \{t, X_t\}_{t=1}^{T}$ denote an N-channel multivariate time series of length T, where each observation at time t is given by

$$X_t = [x_{t,1}, x_{t,2}, \ldots, x_{t,N}]^\top \in \mathbb{R}^N,$$

representing the values of all N channels at that time step. Each channel $x_{t,i}$ can depend on its own past values as well as those of other channels, so the entire sequence $X_{1:T} \in \mathbb{R}^{T \times N}$ jointly encodes both temporal and inter-channel relationships.

Static LMC for Synthetic Data. In the LMC model used by Schlagenhauf et al. (2024), an N-variate series $C(t) \in \mathbb{R}^N$ is generated as:

$$C_i(t) = \sum_{j=1}^{L} \alpha_{i,j}\, l_j(t), \tag{1}$$

where $l_j(t)$ are independent latent Gaussian processes (GPs) drawn from a diverse kernel bank, and $\alpha_{i,j}$ are fixed coregionalization weights. This induces a constant cross-channel covariance:

$$\mathrm{Cov}(C_i, C_k) = \sum_j \alpha_{i,j}\alpha_{k,j}\mathrm{Cov}(l_j, l_j),$$

yielding stationary, instantaneous correlations.

While this framework captures realistic intra-channel variation (trend, seasonality), it cannot model evolving or lagged inter-channel dependencies. In the next section, we extend LMC to dynamically incorporate these structures.

3 DynLMC: Dynamic Linear Model of Coregionalization

To generate synthetic multivariate time series with evolving and lagged inter-channel dependencies, we introduce the DynLMC process, which extends the static LMC framework by allowing dynamic mixing weights and random lags between latent and observed channels. This produces datasets with realistic, non-stationary correlation structures, such as correlation drift, regime shifts, and lead-lag effects. The full procedure is detailed in Algorithm 1 in the Appendix.

3.1 Generative Process

Given L latent GP functions $\{l_j(t)\}_{j=1}^{L}$ sampled via KernelSynth, the DynLMC process generates each observed channel as:

$$C_i(t) = \sum_{j=1}^{L} \alpha_{i,j}(t)\, l_j(t - \tau_{i,j}), \tag{2}$$

where, $\alpha_{i,j}(t)$ are **time-varying mixing weights**, modeled either as smooth autoregressive drifts or discrete regime changes, and $\tau_{i,j}$ are integer lags sampled from $[-\tau_{\max}, \tau_{\max}]$, inducing causal offsets between latent and observed channels.

3.2 Dynamic Correlation Mechanisms

(a) Smooth Drift. Weights evolve as an autoregressive process:

$$\tilde{\alpha}_{i,j}(t) = \rho\, \tilde{\alpha}_{i,j}(t-1) + \epsilon_{i,j}(t), \quad \epsilon_{i,j}(t) \sim \mathcal{N}(0, \sigma^2), \tag{3}$$

with $\rho \in (0.9, 0.99)$ controlling correlation persistence and are normalized by a softmax at each t:

$$\alpha_{i,j}(t) = \frac{\exp(\tilde{\alpha}_{i,j}(t))}{\sum_k \exp(\tilde{\alpha}_{i,k}(t))}.$$

Larger ρ yields slower, smoother drift (stronger temporal persistence), while smaller ρ produces faster-changing weights.

(b) Regime Switching (HMM). We define K latent regimes $\{s_t\}$ with transition probabilities $p(s_t|s_{t-1})$. Each regime k has a fixed mean matrix $\bar{\alpha}^{(k)}$. At each time step:

$$\alpha_{i,j}(t) = (1-\eta)\bar{\alpha}_{i,j}^{(s_t)} + \eta\epsilon_{i,j}(t),$$

where η controls intra-regime variability, introducing structured, interpretable correlation regimes.

(c) Lagged Dependencies. To simulate lead–lag effects, we sample delays $\tau_{i,j} \sim \mathcal{U}[-\tau_{\max}, \tau_{\max}]$. Positive $\tau_{i,j}$ means latent j leads observed i. A larger $\tau_{\max}$ gives a larger temporal offset between the leading and lagging signals.

4 Experiments

We assess the effectiveness of the synthetic data by fine-tuning a pretrained multivariate time series forecasting model on DynLMC-generated data and evaluating robustness and generalization on real data. Details of the fine-tuning strategy are provided in section A.4 of the Appendix.

Models. We evaluate three multivariate time series architectures, TimePFN (Schlagenhauf et al., 2024), iTransformer (Liu et al., 2024), and Chronos-2 (Ansari et al., 2025), representing distinct modeling paradigms to assess the impact of our generated data.

Datasets. Our dataset suite comprises three primary datasets (Section 3.2): (i) *DynLMC - Drift* (ρ varies from 0.92 to 0.99), (ii) *DynLMC - Lag* ($\tau_{\max}$ varies from 0 to 8), and (iii) *DynLMC - Regime Shift* (K varies from 2 to 6). We also include two combined variants: (iv) *DynLMC - Combined*, which mixes equal weights of each primary dataset, and (v) *DynLMC - Combined BO*, which uses Bayesian optimization to determine optimal mixing weights (see Section A.3). Each synthetic dataset contains 1,500 samples, 160 channels, and 1,024 time steps. For the Bayesian optimized variant, the mixing algorithm determines each primary dataset contribution.

Training protocol. For TimePFN and iTransformer, we pretrain each model using the original LMCSynth synthetic data, dataset from TimePFN, then fine-tune the pretrained checkpoints on each of the four DynLMC datasets. For Chronos-2, we use the official pretrained checkpoint due to unavailable pretraining data and similarly fine-tune on each synthetic dataset. Further details on fine-tuning and training infrastructure are in section A.4 of the Appendix.

Testing protocol. We evaluate both pretrained and fine-tuned models on nine real benchmark datasets: ECL, ETTh1/2, ETTm1/2, Exchange, Solar, Traffic, and Weather. Dataset descriptions are in Appendix section A.6.

Table 1: Mean Absolute Error (MAE) performance and number of wins over pretrained baseline, across benchmarks (averaged over 5 seeds). KS = KernelSynth, LMC = static LMCSynth generator, and DynLMC = our dynamic extension with time-varying correlations and lag-structures. The best result is highlighted in bold and the second-best is underlined.

Model	ECL	ETTh1	ETTh2	ETTm1	ETTm2	Exchange	Solar	Traffic	Weather	Wins
TimePFN Backbone										
Baseline pretrained on KS + LMC	0.369	0.440	0.369	0.483	0.298	0.226	1.063	0.596	0.297	–
Fine-tuned (KS + LMC + DynLMC - Drift)	0.371	**0.436**	0.374	0.496	0.299	**0.225**	1.077	0.596	0.297	2
Fine-tuned (KS + LMC + DynLMC - Lag)	0.456	0.459	0.382	0.529	0.321	0.229	**0.967**	0.666	0.300	1
Fine-tuned (KS + LMC + DynLMC - Regime Shift)	0.360	0.438	0.369	0.492	0.301	0.229	1.009	**0.593**	0.301	4
Fine-tuned (KS + LMC + DynLMC - Combined)	**0.356**	0.437	**0.367**	**0.476**	**0.297**	0.228	1.021	0.596	**0.294**	7
iTransformer Backbone										
Baseline pretrained on KS + LMC	0.472	0.462	0.367	0.563	0.307	0.226	0.844	0.676	0.279	–
Fine-tuned (KS + LMC + DynLMC - Drift)	0.456	0.468	0.369	0.518	0.295	0.233	0.763	0.638	**0.278**	7
Fine-tuned (KS + LMC + DynLMC - Lag)	0.436	0.458	0.368	0.524	0.303	0.232	**0.735**	0.672	0.279	7
Fine-tuned (KS + LMC + DynLMC - Regime Shift)	0.376	0.448	**0.358**	0.513	0.294	0.226	0.781	0.610	**0.278**	9
Fine-tuned (KS + LMC + DynLMC - Combined)	**0.368**	**0.447**	0.364	**0.458**	**0.288**	**0.225**	0.762	**0.603**	0.286	8
Chronos-2 Backbone										
Baseline (pretrained checkpoint)	**0.273**	**0.397**	0.344	0.452	0.273	**0.209**	**0.343**	**0.309**	**0.230**	–
Fine-tuned (KS + LMC + DynLMC - Drift)	0.286	0.399	0.345	**0.400**	**0.264**	0.217	0.370	0.337	0.250	2
Fine-tuned (KS + LMC + DynLMC - Lag)	0.290	0.401	0.348	0.406	0.267	0.216	0.348	0.333	0.243	2
Fine-tuned (KS + LMC + DynLMC - Regime Shift)	0.286	0.406	**0.336**	0.418	0.268	0.228	0.368	0.330	0.251	3
Fine-tuned (KS + LMC + DynLMC - Combined)	0.285	0.406	0.343	0.422	0.274	0.231	0.417	0.354	0.248	2
Naïve Predictions										
Naïve Mean	0.794	0.558	0.387	0.548	0.307	0.269	0.734	0.805	0.271	–
Naïve Last	0.996	0.713	0.422	0.665	0.328	**0.196**	0.815	1.077	**0.254**	–
Seasonal Naïve	1.017	0.727	0.431	0.674	0.334	0.205	0.844	1.098	0.263	–

4.1 Zero-Shot Forecasting Results

Table 1 shows the mean absolute error (MAE) for pretrained TimePFN, iTransformer, and Chronos-2 models, as well as their performance after fine-tuning on each DynLMC variant. Results for the DynLMC - Combined BO variant (TimePFN only) are in Table 2 in the Appendix. TimePFN and iTransformer are pretrained solely on synthetic data, while the public Chronos-2 model was also pretrained on real data.

Performance on TimePFN and iTransformer. Fine-tuning on DynLMC consistently improves TimePFN and iTransformer performance, especially on highly multivariate datasets like ECL, Traffic, and Weather. For TimePFN, the Regime Shift and Combined variants achieve 4 and 7 wins, respectively, over the baseline. For iTransformer, Regime Shift and Combined yield 9 and 8 wins, showing that modeling regime shifts and mixed dependencies during synthetic pretraining significantly enhances generalization and robustness.

Performance on Chronos-2. For Chronos-2, improvements are more modest. The Regime Shift variant achieves 3 wins over the official checkpoint, while other variants have smaller or mixed effects. This aligns with the Chronos-2 training, which already includes synthetic data with evolving inter-channel dependencies, reducing the marginal benefit of additional synthetic fine-tuning with DynLMC. Additionally, the Chronos-2 pretraining on both synthetic and real data likely contributes to its stronger baseline and limits further gains from synthetic fine-tuning.

5 Conclusion

We presented **DynLMC**, a dynamic extension of the Linear Model of Coregionalization for realistic synthetic multivariate time series. By incorporating smooth drift, regime-switching, and lagged dependencies, DynLMC bridges the realism gap between synthetic and real data. Fine-tuning TimePFN with DynLMC improves zero-shot forecasting without modifying architecture or training objective. This work highlights the role of realistic inter-channel dynamics as a key ingredient in synthetic pretraining for time-series foundation models.

REFERENCES

Abdul Fatir Ansari, Lorenzo Stella, Ali Caner Turkmen, Xiyuan Zhang, Pedro Mercado, Huibin Shen, Oleksandr Shchur, Syama Sundar Rangapuram, Sebastian Pineda Arango, Shubham Kapoor, Jasper Zschiegner, Danielle C. Maddix, Hao Wang, Michael W. Mahoney, Kari Torkkola, Andrew Gordon Wilson, Michael Bohlke-Schneider, and Bernie Wang. Chronos: Learning the language of time series. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856. URL https://openreview.net/forum?id=gerNCVqqtR. Expert Certification.

Abdul Fatir Ansari, Oleksandr Shchur, Jaris Küken, Andreas Auer, Boran Han, Pedro Mercado, Syama Sundar Rangapuram, Huibin Shen, Lorenzo Stella, Xiyuan Zhang, Mononito Goswami, Shubham Kapoor, Danielle C. Maddix, Pablo Guerron, Tony Hu, Junming Yin, Nick Erickson, Prateek Mutalik Desai, Hao Wang, Huzefa Rangwala, George Karypis, Yuyang Wang, and Michael Bohlke-Schneider. Chronos-2: From univariate to universal forecasting, 2025. URL https://arxiv.org/abs/2510.15821.

James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for hyper-parameter optimization. *Advances in neural information processing systems*, 24, 2011.

Ben Cohen, Emaad Khwaja, Kan Wang, Charles Masson, Elise Ramé, Youssef Doubli, and Othmane Abou-Amal. Toto: Time series optimized transformer for observability, 2024. URL https://arxiv.org/abs/2407.07874.

Abhimanyu Das, Weihao Kong, Rajat Sen, and Yichen Zhou. A decoder-only foundation model for time-series forecasting, 2024. URL https://arxiv.org/abs/2310.10688.

Liang Du, Ruobin Gao, Ponnuthurai Nagaratnam Suganthan, and David ZW Wang. Bayesian optimization based dynamic ensemble for time series forecasting. *Information Sciences*, 591:155–175, 2022.

Vijay Ekambaram, Arindam Jati, Pankaj Dayama, Sumanta Mukherjee, Nam H. Nguyen, Wesley M. Gifford, Chandra Reddy, and Jayant Kalagnanam. Tiny time mixers (ttms): Fast pre-trained models for enhanced zero/few-shot forecasting of multivariate time series. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 74147–74181. Curran Associates, Inc., 2024. doi: 10.52202/079017-2359. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/874a4d89f2d04b4bcf9a2c19545cf040-Paper-Conference.pdf.

Fadi Hamad, Shinpei Nakamura-Sakai, Saheed Obitayo, and Vamsi Potluru. A supervised generative optimization approach for tabular data. In *Proceedings of the Fourth ACM International Conference on AI in Finance*, pp. 10–18, 2023.

Guokun Lai, Wei-Cheng Chang, Yiming Yang, and Hanxiao Liu. Modeling long- and short-term temporal patterns with deep neural networks. *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, 2017. URL https://api.semanticscholar.org/CorpusID:4922476.

Chenghao Liu, Taha Aksu, Juncheng Liu, Xu Liu, Hanshu Yan, Quang Pham, Silvio Savarese, Doyen Sahoo, Caiming Xiong, and Junnan Li. Moirai 2.0: When less is more for time series forecasting, 2026. URL https://arxiv.org/abs/2511.11698.

Yong Liu, Tengge Hu, Haoran Zhang, Haixu Wu, Shiyu Wang, Lintao Ma, and Mingsheng Long. itransformer: Inverted transformers are effective for time series forecasting. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun (eds.), *International Conference on Learning Representations*, volume 2024, pp. 11116–11140, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/2ea18fdc667e0ef2ad82b2b4d65147ad-Paper-Conference.pdf.

Nikolas Schlagenhauf, Theo Galy-Fajou, Matej Balog, Vitus Musil, Johannes von Oswald, Vincent Fortuin, et al. Timepfn: Effective multivariate time series forecasting with synthetic data. *arXiv preprint arXiv:2502.16294*, 2024. Accepted at AAAI 2025.

Artur Trindade. ElectricityLoadDiagrams20112014. UCI Machine Learning Repository, 2015. DOI: https://doi.org/10.24432/C58C86.

Gerald Woo, Chenghao Liu, Akshat Kumar, Caiming Xiong, Silvio Savarese, and Doyen Sahoo. Unified training of universal time series forecasting transformers. In *International Conference on Machine Learning*, 2024. URL https://api.semanticscholar.org/CorpusID:267411817.

Haixu Wu, Jiehui Xu, Jianmin Wang, and Mingsheng Long. Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting, 2022. URL https://arxiv.org/abs/2106.13008.

Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang. Informer: Beyond efficient transformer for long sequence time-series forecasting, 2021. URL https://arxiv.org/abs/2012.07436.

Muhammad Zulfiqar, Kelum AA Gamage, Muhammad Kamran, and Muhammad Babar Rasheed. Hyperparameter optimization of bayesian neural network using bayesian optimization and intelligent feature engineering for load forecasting. *Sensors*, 22(12):4446, 2022.

A Appendix

A.1 Related Work

Time Series Foundation Models. Recent work has investigated foundation models for time series, aiming to pretrain large neural architectures on diverse temporal data and transfer them across tasks and domains. TimesFM Das et al. (2024) and Chronos Ansari et al. (2024) pioneered this direction by framing forecasting as a sequence modeling problem and training transformers on large-scale synthetic univariate time series. The Chronos Ansari et al. (2024) synthetic generation pipeline, KernelSynth, captures common temporal structures such as trends, seasonality, and stochastic noise, enabling strong zero-shot and few-shot performance across heterogeneous benchmarks without reliance on curated real-world datasets. These approaches predominantly focus on univariate series or treat multiple variables independently, limiting their capacity to model cross-variable dependencies inherent in multivariate data.

Multivariate Time Series Modeling. Modeling dependencies across variables is central to multivariate time series forecasting. Classical approaches rely on vector autoregressive or state-space models, while modern deep learning methods employ attention mechanisms and graph-based inductive biases. iTransformer Liu et al. (2024) introduced an inverted attention mechanism that attends across variables rather than time steps, explicitly targeting multivariate dependency modeling and achieving strong empirical performance.

In parallel, models such as Toto Cohen et al. (2024), a general-purpose time series forecasting model specifically tuned for observability metrics, and TTMs Ekambaram et al. (2024), a compact model based on the TSMixer architecture, explored alternative pretraining objectives, tokenization strategies, and transformer architectures for time series foundation models.

Models such as Moirai Woo et al. (2024) and Moirai-2 Liu et al. (2026) extend foundation-model-style training to multivariate settings, emphasizing scalability and robustness across datasets with varying dimensionality. However, these methods primarily rely on real-world multivariate datasets or implicit modeling of inter-variable structure, offering limited control over the statistical properties of cross-series relationships during pretraining.

More recent work in Chronos-2 Ansari et al. (2025) refined architectural choices and scaling behavior, including support for univariate, multivariate, and covariate-informed forecasting and introducing a new group attention mechanism sharing information across multiple time series within a group.

Synthetic Data for Multivariate Pretraining. Synthetic data generation has proven critical for scaling time series foundation models. Chronos Ansari et al. (2024) demonstrated that synthetic

pretraining can induce useful inductive biases and broad generalization. However, its generation pipeline assumes independent univariate series and does not model interactions between variables. TimePFN bridges this gap by extending the Chronos synthetic generation framework to multivariate time series through correlated covariates, enabling the model to learn cross-variable dependencies during pretraining. This approach shows improved performance on multivariate downstream tasks, particularly in data-scarce regimes.

Nevertheless, the generated dependencies are largely static, assuming fixed correlation structures across time, and often across all variates. In contrast, real-world multivariate time series frequently exhibit non-stationary dependency patterns, including correlation drift, lagged interactions, and regime-dependent relationships between variables. Such dynamics are common in financial, environmental, and sensor data but remain underrepresented in existing synthetic pretraining pipelines. Concurrent work to ours in Chronos-2 Ansari et al. (2025) introduced the concept of multivariatizers, sampling multiple time series from base univariate generators and introducing correlations between variates in a series and dependencies across time.

Our work extends synthetic multivariate pretraining by explicitly modeling dynamic inter-variable structure. Building on Chronos Ansari et al. (2024) and TimePFN Schlagenhauf et al. (2024), we introduce synthetic generators that incorporate correlation drift, variable-specific lags, and regime shifts between covariates. By enriching the synthetic training distribution with these realistic dependency dynamics, we enable multivariate foundation models to better generalize to non-stationary and structurally complex real-world datasets. While our contributions in introducing inter-dependencies between variates overlap with Chronos-2 Ansari et al. (2025), we are making our multivariate time series generation pipeline available upon request and show that fine-tuning the open-source Chronos-2 Ansari et al. (2025) model with the DynLMC data improves its performance in several real datasets.

A.2 DynLMC Dataset Generation Algorithm

Algorithm 1 DynLMC Generation Algorithm

Input: Number of variates N, time-series length T, Weibull shape β, Weibull scale λ, minimum number of latent functions m, maximum number of kernel compositions in KernelSynth J, drift parameters (ρ, σ), regime type $r \in \{\texttt{none}, \texttt{hmm}\}$, number of HMM regimes K, HMM stickiness κ, lag bound $\tau_{\max}$

Output: Synthetic MTS C with N variates and length T

$L \sim \max(\min(\text{Weibull}(\beta, \lambda), N), m)$
for $j \in \{1 \dots L\}$ **do**
    $l_j(t) \leftarrow \text{KernelSynth}(J, T)$
end for
$\tau_{i,j} \sim \text{Unif}\{-\tau_{\max}, \dots, \tau_{\max}\}$ for all (i, j)
if $r = \texttt{none}$ **then**
    For all (i, j): define logits $A_{i,j}(t) \leftarrow \text{AR1Path}(T; \rho, \sigma)$
else
    Sample HMM states $s_{1:T}$ with stickiness κ and regime means $\bar{\alpha}^{(k)}$
    For all (i, j): $A_{i,j}(t) \leftarrow \bar{\alpha}_{i,j}^{(s_t)} + \text{AR1Path}(T; \rho, \sigma)$
end if
For each t: $\big[\alpha_{i,1}(t), \dots, \alpha_{i,L}(t)\big] \leftarrow \text{softmax}\big(A_{i,1:L}(t)\big)$ for all i
for $i \in \{1 \dots N\}$ **do**
    $C_i(t) \leftarrow \sum_{j=1}^{L} \alpha_{i,j}(t)\, l_j\big(t - \tau_{i,j}\big)$ for $t = 1 \dots T$
end for
return $\{C_i(t)\}_{i=1}^{N}$

A.3 Bayesian Optimization for Synthetic Dataset Mixing

Our dataset generation model DynLMC provides methods for generating the three distinct types of multivariate time series data, as described in Section 3.2: (i) DynLMC Drift, where inter-variable correlations evolve smoothly over time; (ii) DynLMC Lag, where dependencies manifest with temporal offsets between variates; and (iii) DynLMC Regime Shift, characterized by abrupt changes in the underlying correlation structure. Finally, we combine the three datasets with equal weights to create a fourth variant (iv) DynLMC - Combined, allowing us to assess whether incorporating all three datasets into the training process yields any additional performance improvements.

Beyond this naive mixing we also investigate the potential of a data centric generative optimization procedure, (v) DynLMC - Combined BO, that learns an optimal mixture over multiple synthetic time series sources with the objective to optimize the training data distribution for downstream forecasting accuracy. This is a time series extension of the optimization framework introduced for tabular data Hamad et al. (2023). Let M denote the index set of all available synthetic time series sources where each source corresponds to a distinct temporal generative process (e.g drift/lag, regime shift generators) and let $\{S_m\}$ denote the dataset corresponding to the source $m \in M$. At each iteration, we construct a synthetic training mixture by sampling sliding window sequences (X, Y) where $X \in \mathbb{R}^{T \times N}$ and $Y \in \mathbb{R}^{T_{\text{pred}} \times N}$ from each source according to mixture weights $\boldsymbol{\gamma}^{(0)} \in \Delta^{|M|}$. In some occasions where synthetic sources could have different channel dimensionalities, we can align these windows to a common channel dimensionality $\boldsymbol{N}^\star$ which can be used to train a forecasting model f_ϕ(in our implementation, a chronos/TimePFN-style transformer encoder. The trained model is then evaluated on a real validation dataset $\mathcal{D}_{\text{val}}$ using a downstream forecasting metric such as mean absolute error (MAE). The resulting validation loss define a black-box objective over a mixture of weights. Concretely, a single evaluation of this objective consist of training the forecasting model on a candidate synthetic mixture and computing its MAE on the validation set:

$$\ell(\gamma) = \mathcal{L}(f_\phi(D_{\text{syn}}(\gamma)), D_{\text{val}}). \tag{4}$$

where $\mathcal{D}_{\text{syn}}$ denotes the synthetic dataset sampled from the mixture defined by $\boldsymbol{\gamma}$. We optimize this objective using sequential based optimization via the Tree-structured Parzen Estimator (TPE) Bergstra et al. (2011), which iteratively proposes mixture weights, retrains the forecasting model on the resulting synthetic dataset, and updates its surrogate model based on observed validation loss. After K trials, we select $\boldsymbol{\gamma}^\star = \arg\min_k \ell^{(k)}$ and get the mixture dataset (X, Y) sampled according to $\boldsymbol{\gamma}^\star$. Unlike prior work that applies Bayesian optimization primarily for hyperparameter tuning of time series models Zulfiqar et al. (2022) or ensemble weighting Du et al. (2022), our approach treats the training data as the optimization variable. This yields a data centric optimization framework in which the optimal synthetic mixture is defined by its downstream predictive utility on real time series forecasting tasks. Algorithm 2, details this method.

Bayesian Optimization Mixing - Results. Table 2 compares the mean absolute error (MAE) of the fine-tuned model on the naively combined dataset variant and the fine-tuned model on the mixed dataset variant generated using the Bayesian optimization algorithm from Section A.3.

Table 2: Mean Absolute Error (MAE) performance across datasets combining drift, lag and regime shift characteristics, naively and using a Bayesian mixing method. Results averaged over 5 seeds. The best result is highlighted in bold.

Model	ECL	ETTh1	ETTh2	ETTm1	ETTm2	Exchange	Solar	Traffic	Weather	Wins
TimePFN Backbone										
Fine-tuned (KernelSynth + LMCSynth + DynLMC - Combined)	**0.356**	**0.437**	0.367	**0.476**	0.297	0.228	1.021	0.596	0.294	-
Fine-tuned (KernelSynth + LMCSynth + DynLMC - Combined BO)	0.378	0.440	**0.363**	0.502	**0.294**	**0.226**	**0.882**	**0.592**	**0.270**	6

Algorithm 2 Bayesian Optimization for Time Series Dataset Mixing

Input: Time-series source generators $\{S_m\}_{m \in M}$; validation dataset $\mathcal{D}_{\text{val}}$; Forecasting model f_ϕ; Window parameters $(L_{\text{seq}}, L_{\text{label}}, L_{\text{pred}})$; time split $(\text{holdout}_{frac}, \text{holdout}_{tail})$; Number of variates N; Bayesian optimizer $\mathcal{B}$

Output: Optimal mixture weights $\boldsymbol{\gamma}^\star$; Mixture dataset $(X^\star, Y^\star)$

Initialize mixture weights $\boldsymbol{\gamma}^{(0)} \in \Delta^M$
Set channel target $C^\star = \max_m C_m$
for $k \in \{1 \dots K\}$ **do**
  Compute allocation $n_m^{(k)} = \lfloor \gamma_m^{(k)} N \rfloor$ with $\sum_m n_m^{(k)} = N$
  for $m \in \{1 \dots M\}$ **do**
    Sample windows $(X_m^{(k)}, Y_m^{(k)}) \leftarrow \textsc{SampleWindows}(S_m, n_m^{(k)}, L_{\text{seq}}, L_{\text{pred}})$
    Apply channel alignment $X_m^{(k)} \leftarrow \textsc{Align}(X_m^{(k)}, C^\star)$ and $Y_m^{(k)} \leftarrow \textsc{Align}(Y_m^{(k)}, C^\star)$
  end for
  Compose mixture dataset $(X^{(k)}, Y^{(k)}) = \bigcup_{m \in M} (X_m^{(k)}, Y_m^{(k)})$
  Train model with AMP + early stopping:

$$\phi^{(k)} \leftarrow \text{Train}(f_\phi, (X^{(k)}, Y^{(k)}), \phi_0)$$

  Subsample validation windows $(X_{\text{val}}, Y_{\text{val}}) \leftarrow \textsc{Subsample}(D_{\text{val}}, N_{\text{val}})$ and align channels
  Predict $\hat{Y}_{\text{val}}^{(k)} = f_{\phi^{(k)}}(X_{\text{val}})$
  Compute loss $\ell^{(k)} = \mathcal{L}(\hat{Y}_{\text{val}}^{(k)}, Y_{\text{val}})$
  Update mixture using BO:

$$\boldsymbol{\gamma}^{(k+1)} \leftarrow \mathcal{B}\big(\{(\boldsymbol{\gamma}^{(i)}, \ell^{(i)})\}_{i=1}^{k}\big)$$

end for
Select $k^\star = \arg\min_k \ell^{(k)}$ and set $\boldsymbol{\gamma}^\star = \boldsymbol{\gamma}^{(k^\star)}$
return $\boldsymbol{\gamma}^\star, (X^\star, Y^\star)$

A.4 Fine-Tuning Strategy

Our objective is to enrich the pretrained model with additional temporal dependency patterns, while avoiding overfitting to any single synthetic distribution. The original TimePFN work adopted a curriculum learning paradigm for multivariate adaptation. Concretely, they warm-started training by fully fine-tuning on KernelSynth, a dataset derived from the Chronos Ansari et al. (2024) univariate generation pipeline and adapted to the multivariate setting by concatenating independent time series into tensors of shape $(N \times T)$, where N denotes the number of channels and T the sequence length. Subsequently, they introduced LMCSynth, a dataset of dependent multivariate time series generated via a linear model of coregionalization (LMC), thereby switching from a regime of no inter-variable dependencies to one including them.

In our setting, however, curriculum training did not yield optimal results. Instead, we observed improved generalization by jointly mixing the data used during pretraining with newly generated synthetic data. Specifically, we interleave samples from KernelSynth, LMCSynth, and our proposed dataset DynLMC, which explicitly models dynamic correlation structures. Importantly, the contribution of DynLMC increases over the course of training. This progressive reweighting encourages the model to first stabilize on simpler or previously seen distributions before gradually incorporating more complex and dynamically evolving dependency patterns.

Progressive Dataset Mixing. Let $\mathcal{D}_1$, $\mathcal{D}_2$, and $\mathcal{D}_3$ denote KernelSynth, LMCSynth, and DynLMC, respectively. At training index i, we define a mixing probability

$$p_{\text{mix}}(i) = \log\left(1 + \frac{2i}{M}\right), \tag{5}$$

where M is the total dataset size (minimum length across datasets). Since $p_{\text{mix}}(i)$ is monotonically increasing in i, the probability of constructing mixed samples grows over time. Mixed samples are formed by concatenating variable subsets from each dataset along the channel dimension and

Algorithm 3 Progressive Interleaved Mixing of Three Datasets

Require: Datasets $\mathcal{D}_1$ (KernelSynth), $\mathcal{D}_2$ (LMCSynth), $\mathcal{D}_3$ (DynLMC)

$N \leftarrow \min(|\mathcal{D}_1|, |\mathcal{D}_2|, |\mathcal{D}_3|)$
for index $i = 0, \ldots, N-1$ **do**
  $p_{\text{mix}} \leftarrow \log\left(1 + \frac{2i}{M}\right)$
  Sample $u \sim \text{Uniform}(0, 1)$
  if $u < p_{\text{mix}}$ **then**
    Sample $(x_1, y_1) \sim \mathcal{D}_1[i]$
    Sample $(x_2, y_2) \sim \mathcal{D}_2[i]$
    Sample $(x_3, y_3) \sim \mathcal{D}_3[i]$
    Randomly choose channel split sizes
    Concatenate channel subsets:

$$x \leftarrow \text{concat}(x_1, x_2, x_3), \quad y \leftarrow \text{concat}(y_1, y_2, y_3)$$

    Randomly permute channel order
  else
    Randomly choose $k \in \{1, 2, 3\}$
    Sample $(x, y) \sim \mathcal{D}_k[i]$
  end if
  return (x, y)
end for

randomly permuting channels to prevent positional bias. Otherwise, with probability $1 - p_{\text{mix}}(i)$, a single dataset is sampled uniformly at random. This design ensures that dynamically correlated samples from DynLMC are incorporated more frequently in later training iterations, effectively implementing a soft curriculum through probabilistic mixing rather than staged fine-tuning. Algorithm 3, which details this method, is included in the appendix.

Retention–Adaptation Tradeoff in Chronos-2. Unlike TimePFN and iTransformer, where we control the full synthetic pretraining pipeline, we do not have access to the original Chronos-2 training corpus. Consequently, during DynLMC fine-tuning we cannot explicitly preserve the data distribution learned during pretraining. To mitigate potential overfitting to DynLMC, we freeze all but the final three layers and adopt a reduced learning rate. Nevertheless, the absence of the original pretraining distribution may limit our ability to balance knowledge retention and adaptation. This could explain why DynLMC fine-tuning occasionally degrades performance on certain real-world datasets for Chronos-2, despite producing gains elsewhere. Importantly, because we compare the official pretrained checkpoint to our DynLMC fine-tuned version under identical evaluation conditions, the reported differences isolate the contribution of DynLMC fine-tuning itself.

A.5 Training Infrastructure

All experiments were run on an AWS g6e.2xlarge instance with a single NVIDIA L40S GPU (48GB VRAM), 8 vCPUs, and 64GB RAM for both training and evaluation. Fine-tuning samples were created by slicing synthetic data with a 96-step context window and 96-step forecasting horizon. Early stopping with a patience of 200 iterations was used and results are averaged over five random seeds for both pretraining and fine-tuning.

A.6 Testing Datasets

We evaluated all pretrained and fine-tuned models on nine real-world datasets: *ECL*, *ETTh1*, *ETTh2*, *ETTm1*, *ETTm2*, *Exchange*, *Solar*, *Traffic*, and *Weather*. These datasets encompass diverse real-world distributions and capture a range of temporal patterns. Brief descriptions of each dataset are provided below:

- ECL: The ECL (Electricity Load Diagrams; Trindade (2015)) contains hourly electricity consumption data for 321 clients. The dataset includes 18317/2633/5261 samples for train/validation/test splits.
- ETTh Family: The ETTH (Electricity Transformer Temperature; Zhou et al. (2021)) datasets each contain seven variates. ETTh1 and ETTh2 are sampled hourly, while ETTm1 and ETTm2 are sampled every 15 minutes. ETTh datasets include 8545/2881/2881 samples for train/validation/test splits, and ETTm datasets contain 34465/11521/11521 samples, respectively.
- Exchange: The Exchange Wu et al. (2022) dataset offers daily exchange rates for eight countries, with eight variables in total. The dataset includes 5120/665/1422 samples for train/validation/test splits.
- Solar: The Solar Lai et al. (2017) dataset contains power production data from 137 solar power plants, sampled every 10 minutes, yielding 137 variables. The dataset includes 36601/5161/10417 samples for train/validation/test splits.
- Traffic: The Traffic Wu et al. (2022) dataset comprises hourly road occupancy rates from 862 locations, yielding 862 variables. The dataset includes 12185/1757/3509 samples for train/validation/test splits.
- Weather: The Weather Wu et al. (2022) dataset consists of 21 meteorological variables collected every 10 minutes and contains 36792/5271/10540 samples for train/validation/test splits.