CauKer: Classification Time Series Foundation Models Can Be Pretrained on Synthetic Data only

Shifeng Xie [1] **Vasilii Feofanov** [1] **Marius Alonso** [1] **Ambroise Odonnat** [1] **Jianfeng Zhang** [1] **Ievgen Redko** [1]

Abstract

Time series foundation models (TSFMs) have recently gained significant attention due to their strong zero-shot capabilities and widespread real-world applications. Such models typically require a computationally costly pretraining on large-scale, carefully curated collections of real-world sequences. To allow for a sample-efficient pretraining of TSFMs, we propose CauKer, a novel algorithm designed to generate diverse, causally coherent synthetic time series with realistic trends, seasonality, and nonlinear interactions. CauKer combines Gaussian Process (GP) kernel composition with Structural Causal Models (SCM) to produce data for sample-efficient pretraining of state-of-the-art classification TSFMs having different architectures and following different pretraining approaches.

1. Introduction

Time series data are ubiquitous in applications ranging from healthcare (Gnassounou et al., 2025) and human activity recognition (Chen et al., 2025) to industrial monitoring (Susto et al., 2018). Recently, the time series community has devoted significant effort to developing large-scale pretrained time series foundation models (TSFMs). Inspired by advances in natural language processing and computer vision, these models aim to achieve strong zero-shot performance in out-of-distribution (OOD) settings. TSFMs have been proposed for both forecasting (Ansari et al., 2024; Woo et al., 2024; Bhethanabhotla et al., 2024) and classification tasks (Goswami et al., 2024; Lin et al., 2024; Feofanov et al., 2025), showing promising results. TSFMs are usually trained on large-scale pretraining dataset collections gathered from different application domains. Recent works used as many as 1.13 billion timepoints of 13M unique time series for model pretraining (Goswami et al., 2024).

Despite the prevalence of large-scale pretraining in the development of TSFMs, several works (Hoo et al., 2024; Dooley et al., 2023; Taga et al., 2025) showed that comparable performance can be achieved by training them purely on synthetic data. The latter approach has several important advantages. First, it removes the need for time-consuming data collection and curation. This is especially important in time series classification that lacks diverse and rich pretraining corpora. Second, it allows for generating arbitrarily large datasets for model scaling. Finally, it makes the OOD evaluation more meaningful, mitigating the risk of data leakage. Inspired by the recent success of foundation models in tabular classification (Hollmann et al., 2023), our paper proposes a novel sample-efficient pretraining framework for TSFMs in classification based purely on synthetic data. Contrary to tabular and forecasting synthetic data generation pipelines, our proposal seeks to generate sequences with meaningful correlations between samples and realistic temporal dependencies within them. We provide an in-depth, large-scale study of its benefits compared to pretraining on commonly used time series classification corpora. [1,2].

Findings Overall, our findings can be summarized as follows:

1. A carefully designed synthetic data generation pipeline can be efficiently used in training classification TSFMs. We propose such a pipeline and show that it requires rethinking synthetic data generators proposed previously for tabular data and time series forecasting.

2. Pretraining on synthetic data reveals clear scaling laws both in terms of dataset size and model size. We illustrate this finding by showing that such scaling laws are broken when using common classification benchmarks for pretraining, likely due to the lack of diversity in existing classification datasets.

[*]Equal contribution [1]Paris Noah's Ark Lab, Huawei. Correspondence to: Shifeng Xie <shifeng.xie@telecom-paris.fr>, Ievgen Redko <ievgen.redko@huawei.com>.

Proceedings of the 1^{st} *ICML Workshop on Foundation Models for Structured Data*, Vancouver, Canada. 2025.

[1]A complete review of the related work can be found in Appendix A.

[2]Code implementation is available at `https://github.com/ShifengXIE/CauKer.git`.

3. Distinct from forecasting (Yao et al., 2025), where the leaderboard (with the exception of (Hollmann et al., 2023)) is still dominated by models pretrained on large-scale real-world datasets, we show that pretraining on solely synthetic data can lead to state-of-the-art performance in classification.

2. Our contributions

2.1. Problem setup

Zero-shot classification As done in prior work on unsupervised representation learning (Franceschi et al., 2019; Yue et al., 2022), we see a TSFM as an encoder $F : \mathbb{R}^t \to \mathbb{R}^q$ that is kept frozen during the evaluation. For a downstream classification dataset $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^n$ with labels $y_i \in \{1, \dots, C\}$, we use a TSFM to obtain embeddings $z_i = F(x_i)$ and train a lightweight classifier $h : \mathbb{R}^q \to \{1, \dots, C\}$ solely on $\{(z_i, y_i)\}$. At test time, an unseen series x^* is classified by $\hat{y} = h\big(F(x^*)\big)$. As F is kept frozen, the resulting accuracy measures the quality of its learned representations.

To quantify OOD generalization ability, we follow (Yao et al., 2025) and evaluate the studied TSFMs only on samples not seen during their pretraining. In practice, if we evaluate a given TSFM on a test set from a UCR (Dau et al., 2019) dataset, we ensure that the TSFM was not pretrained on it, but we allow for the train set of this same dataset to be used for pretraining. We note that (Feofanov et al., 2025; Goswami et al., 2024; Lin et al., 2024) all used train sets from the datasets on which they've reported the zero-shot OOD generalization. Next, a lightweight classifier h is fitted on the UCR train set embeddings and evaluated on the disjoint UCR test set embeddings as explained above.

2.2. CAUKER: synthetic data generation for time series classification

We now present our proposed synthetic data generation pipeline, termed CAUKER for **Cau**sal-**Ker**nel generation. To develop our intuition about it, we note that the synthetic data for the time series classification task needs to combine two key ingredients. On the one hand, the generated sequences should exhibit common time series patterns such as seasonality, periodicity, and trend. On the other hand, successful classification assumes that individual time series have a meaningful clustering structure that allows the trained model to successfully learn how to disentangle the underlying clusters during training. Below, we present a generation pipeline that satisfies these desiderata.

Proposed approach To proceed, we now define three banks of functions, namely: kernel, mean and activation banks denoted as $\mathcal{K} = \{\kappa_i(t,t')\}_{i=1}^{n_\mathcal{K}}$, $\mathcal{M} = \{\mu_i(t)\}_{i=1}^{n_\mathcal{M}}$ and $\mathcal{A} = \{\sigma(t)_i\}_{i=1}^{n_\mathcal{A}}$, respectively. For the kernel bank, we use the same kernel functions as (Ansari et al., 2024). For mean functions, we consider a linear function $ax + b$, exponential function ae^{bx}, and anomaly mean function that inserts random values from $\mathcal{U}(-5, 5)$ at random indexes. Finally, the activation functions we use for $\mathcal{A}$ are a linear function $ax + b$ with $a \sim \mathcal{U}(0.5, 2)$, $b \sim \mathcal{U}(-1, 1)]$, ReLU activation, sigmoid, sine function, element-wise modulo operation $x \bmod c$ for $c \sim \mathcal{U}[1, 5]$, and Leaky ReLU with a random negative slope from $\mathcal{U}(0.01, 0.3)$. For simplicity, in what follows we let $\{s_i\}_{i=1}^n \sim \mathcal{S}$ denote an i.i.d. sampling (without replacement) of n elements from a set $\mathcal{S}$.

Our generative pipeline, illustrated in Figure 1, then proceeds in five steps as follows:

Step 1. **Kernel bank sampling** We start by sampling candidate kernels from the kernel bank, *ie*, $\{\kappa_i(t,t')\}_{i=1}^K \overset{\text{i.i.d.}}{\sim} \mathcal{K}$ for some random number of candidate kernels $K \sim \mathcal{U}(1, n_\mathcal{K})$.

Step 2. **Kernel composition** We define a composite kernel based on $K-1$ randomly sampled binary operations ($+$ and $\times$). More formally, for a random sequence $\{\star_i\}_{i=1}^{K-1} \sim \{+, \times\}$, we let $\kappa^* = \kappa_1(t,t') \star_i \cdots \star_{K-1} \kappa_K(t,t')$.

Step 3. **Root nodes generation** We draw M mean functions $\{\mu_i(t)\}_{i=1}^M \overset{\text{i.i.d.}}{\sim} \mathcal{M}$, $M \sim \mathcal{U}(1, n_\mathcal{M})$ and repeat Step 1 and Step 2 M times to obtain composite kernels $\{\kappa_i^*\}_{i=1}^M$. We further define M GP priors to sample from $\{\mathcal{GP}(\mu_i, \kappa_i^*)\}_{i=1}^M$.

Step 4. **Activation bank sampling** We sample a set of E activation functions from the activation bank, *ie*, $\{\sigma_i\}_{i=1}^E \sim \mathcal{A}$, $E \sim \mathcal{U}(1, n_\mathcal{A})$.

Step 5. **Causal graph propagation** We randomly generate a directed acyclic graph (DAG) $(\mathcal{V}, \mathcal{E})$ with $|\mathcal{E}| = E$, $|\mathcal{V}| = V$, and $M < V$ *root nodes*, i.e., nodes with in-degree zero. We then define a bijection $\phi : \mathcal{E} \to \{\sigma_1, \sigma_2, \dots, \sigma_E\}$ such that each directed edge $e_{ij} = (u_i, v_j)$ is uniquely associated with a function σ_l, i.e., $\phi(e_{ij}) = \sigma_l$. We then associate a time series $t_i \in \mathbb{R}^L$ sampled from $\mathcal{GP}(\mu_i, \kappa_i^*)\}$ to each of the M root nodes. The value t_{v_j} associated with a given non-root vertex v_j is then calculated as follows. First, for each incoming edge e_{ij}, we apply an activation function $\phi(e_{ij})$ to t_{u_i}. Then, we aggregate all $\phi(e_{\cdot j})(t_{u_\cdot})$ using a randomly initialized linear layer with weights and biases $W, b \sim \mathcal{N}(0, 1)$, *ie*, $t_{v_j} = W \times [\phi(e_{\cdot j})(t_{u_\cdot})] + b$, with $[\cdot]$ denoting the concatenation operation.

The composition of the kernel, mean, and activation banks, are provided in Appendix D.

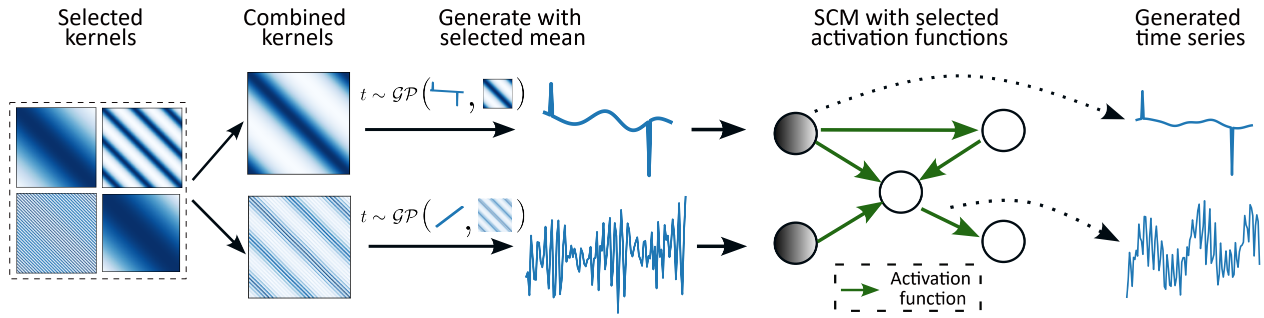

Figure 1: An illustration of the proposed CAUKER pipeline. Kernels sampled from the kernel bank $\mathcal{K}$ are randomly combined and used together with sampled mean functions to form GP priors. Time series sampled from these GP priors act as root nodes in a directed acyclic graph that encodes causal dependencies between nodes. Each edge of this graph applies an activation function from a predefined activation function bank and aggregates over incoming edges using a random linear transformation to propagate transformed time series through the graph. Intermediate node outputs are optionally interpolated to fixed length, forming the final synthetic dataset. This procedure yields rich, diverse, and causally consistent time series for self-supervised pretraining.

3. Experimental results

In all our experiments, we consider two recent TSFMs, namely Mantis and MOMENT. Mantis is an 8M encoder-only model pretrained using contrastive learning. We use the 77M version of the MOMENT model. The latter is an encoder-decoder model pretrained based on masked reconstruction. Considering these two models allows us to compare two different pretraining paradigms as previously done in (Yao et al., 2025) for forecasting. Finally, we follow (Feofanov et al., 2025) and evaluate Mantis in a zero-shot regime by learning a Random Forest classifier on the embeddings of training examples of a given dataset. For MOMENT, (Goswami et al., 2024) evaluated their model using an Support Vector Machine classifier. For both models, we report the test accuracy averaged over 128 UCR datasets, where each dataset has train and test sets following (Dau et al., 2019). Detailed formulations of the loss functions and architecture specifics for these models are provided in the Appendix C.

3.1. CAUKER against alternative synthetic generators

Experimental setup To better understand the exact contribution of the proposed CAUKER, we first start by establishing the virtues of our synthetic data generation pipeline compared to prior work. For this, we generate four different synthetic corpora, namely: 1) FPFN (Taga et al., 2025) that uses a linear model of coregionalization to sample multivariate time series, 2) Kernel (Ansari et al., 2024) that randomly composes covariance kernels to define a Gaussian process with zero mean; 3) Mean+Kernel: our re-implementation of the Kernel baseline in which we additionally add non-zero mean functions in the GP; 4) SCM, a reconstruction of the structural-causal model proposed by Hollmann et al. (2023) for tabular classification [3]. We generate univariate time series with length $T = 512$ as both Mantis and MOMENT were trained on time series of this length. For a fair comparison, we fix the number of synthetic samples to 100K.

Table 1: Average zero-shot accuracy (%) on the UCR benchmark after pretraining on synthetic corpora generated by different methods.

Model	SCM	FPFN	Kernel	Mean-Kernel	CAUKER
Mantis	73.49	77.52	77.70	78.20	**78.31**
MOMENT	59.23	70.85	69.31	72.56	**74.24**

Results Table 1 shows a relative comparison of our proposal compared to other methods. Our first observation is that classification-tailored tabular data generation pipeline SCM underperforms significantly compared to all other methods. This suggests that temporal dependencies are important for time series classification, differently from the forecasting setup, where TabPFN trained using SCM-generated data is among the strongest foundation models. We further note that forecasting-tailored FPFN and KernelSynth also provide suboptimal results, even more so for MOMENT. In the case of Mantis, the results of pretraining on these two datasets are closer to the reported performance of the Mantis model. This can be likely explained by the architecture of Mantis that incorporates strong time series classification priors into it (mean, standard deviation, and difference encoding in the token generator unit). On the contrary, MOMENT is a generic encoder-decoder model.

[3]As the original generator of (Hollmann et al., 2023) is not open–sourced, we followed the algorithmic description in the paper and validated the implementation on the illustrative examples provided therein.

We further note a distinct positive effect of including non-mean functions in the GP used to generate time series in our pipeline. Finally, CAUKER improves upon this stronger baseline in both cases, highlighting the additional benefit of causal structure. The last two observations are particularly valid for MOMENT, indicating that they compensate for the lack of useful inductive biases for the task of time series classification.

3.2. Sample-efficient pretraining of TSFMs using CAUKER synthetic data

Experimental setup We want to study the performance and the sample efficiency of pretraining Mantis and MOMENT foundation models on different datasets. Our main goal is to show that the performance of both models pretrained on a total of 1.89M (Mantis) and 13M (MOMENT) unique time series can be almost matched by a pretraining on a smaller synthetic dataset generated using CAUKER. For the latter, we generate as few as 100k samples for Mantis and 10M for MOMENT to account for the model size difference (8M vs. 77M). As before, we include in our study a baseline given by pretraining Mantis and MOMENT on 100k samples of the real-world UEA time series classification collection. Additionally, we also experiment with a subset of 100k time series randomly drawn from standard forecasting datasets (ETTh1, ETTh2, ETTm1, ETTm2, Electricity, ExchangeRate, Illness, Traffic, Weather) (Zhou et al., 2021; Li et al., 2020; Lai et al., 2018; Matsubara et al., 2014; Li et al., 2018; Rasp et al., 2020). Although no prior work trained a classification model on such data, we include it to verify whether the forecasting benchmarks can be a good alternative for classification TSFM pretraining.

Results From the results presented in Table 5, we note that the performances of Mantis and MOMENT can be almost matched by pretraining them on synthetic datasets that are $\sim 20\times$ and $\sim 1.3\times$ smaller than the original pretraining datasets used by each of the papers. The accuracy drop in the case of Mantis is less than 0.1%, while for MOMENT it barely exceeds 1%. This suggests that the synthetic data generated by CAUKER makes model pretraining more sample-efficient. We also note that the training loss and test accuracy of Mantis pretrained on 100k and 1.89M time series exhibit a very different behavior. For the synthetic dataset the training loss (shown in E.3) remains higher indicating that it is harder to learn, likely due to the high diversity of the generated time series. Yet, the test accuracy in this case steadily improves and surpasses the accuracy of the original model which quickly learns the real-world pretraining dataset. This is reminiscent of the MOMENT pretraining which only required 2 epochs (Goswami et al., 2024) (even for the largest 783M) to converge.

In addition to this, the reported UCR classification accuracies of the original Mantis and MOMENT models represent *in-distribution* performance, since their respective training corpora include UCR train samples. In this sense, these scores may serve as a practical upper bound for zero-shot accuracy, beyond which out-of-distribution generalization is unlikely without direct exposure to test distributions. Finally, we note that the comparison with two other pretraining dataset candidates leads to strictly worse results, despite their comparable size.

Additionally, our experiments reveal that CAUKER-generated datasets exhibit clear scaling laws for both dataset size (10K to 10M samples) and model capacity (1M to 783M parameters), unlike real-world datasets, which display irregular scaling behavior. A thorough discussion of the observed data–, model– and compute–scaling laws is deferred to Appendix E.

Table 2: Performance comparison of Mantis and MOMENT models on different pretraining datasets.

Model	Pretrain. set	Size	UCR Included?	UCR acc. (%)
Mantis	CAUKER	100K	No	78.55
	Mantis dataset	1.89M	Yes	78.66
	UEA	100K	No	76.73
	Forecasting	100K	No	75.81
MOMENT	CAUKER	10M	No	77.49
	Time Series Pile	13M	Yes	78.85
	CAUKER	100K	No	74.24
	UEA	100K	No	73.55
	Forecasting	100K	No	73.93

4. Conclusion

In this work, we introduced CAUKER, a novel synthetic data generation framework tailored for time series classification. By integrating Gaussian Process kernel composition with Structural Causal Models, CAUKER generates synthetic datasets that are both temporally realistic and causally coherent. We demonstrated that TSFMs pretrained solely on CAUKER-generated data can match the performance of models trained on larger real-world datasets.

Our findings underscore a key insight already known in vision and natural language processing: the quality and structure of pretraining data have a profound impact on the generalization performance of TSFMs. While much recent progress in time series community has focused on architectural innovations, our results suggest that equivalent gains can be achieved through principled design of synthetic training data. We hope this work encourages the community to direct greater attention to the design, analysis, and benchmarking of time series training datasets, as a complementary path toward building scalable, general-purpose time series foundation models.

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.

Bagnall, A., Dau, H. A., Lines, J., Flynn, M., Large, J., Bostrom, A., Southam, P., and Keogh, E. The uea multivariate time series classification archive, 2018, 2018. URL https://arxiv.org/abs/1811.00075.

Bhethanabhotla, S. K., Swelam, O., Siems, J., Salinas, D., and Hutter, F. Mamba4cast: Efficient zero-shot time series forecasting with state space models. *arXiv preprint arXiv:2410.09385*, 2024.

Cao, D., Jia, F., Arik, S. O., Pfister, T., Zheng, Y., Ye, W., and Liu, Y. Tempo: Prompt-based generative pre-trained transformer for time series forecasting. *arXiv preprint arXiv:2310.04948*, 2023.

Chang, C., Peng, W.-C., and Chen, T.-F. Llm4ts: Two-stage fine-tuning for time-series forecasting with pre-trained llms. *arXiv preprint arXiv:2308.08469*, 2023.

Chang, C., Wang, W.-Y., Peng, W.-C., and Chen, T.-F. Llm4ts: Aligning pre-trained llms as data-efficient time-series forecasters. *ACM Trans. Intell. Syst. Technol.*, 16 (3), April 2025. ISSN 2157-6904. doi: 10.1145/3719207. URL https://doi.org/10.1145/3719207.

Chen, B., Wongso, W., Li, Z., Khaokaew, Y., Xue, H., and Salim, F. Comodo: Cross-modal video-to-imu distillation for efficient egocentric human activity recognition. *arXiv preprint arXiv:2503.07259*, 2025.

Chung, H. W., Hou, L., Longpre, S., Zoph, B., Tay, Y., Fedus, W., Li, E., Wang, X., Dehghani, M., Brahma, S., Webson, A., Gu, S. S., Dai, Z., Suzgun, M., Chen, X., Chowdhery, A., Narang, S., Mishra, G., Yu, A., Zhao, V., Huang, Y., Dai, A., Yu, H., Petrov, S., Chi, E. H., Dean, J., Devlin, J., Roberts, A., Zhou, D., Le, Q. V., and Wei, J. Scaling instruction-finetuned language models, 2022. URL https://arxiv.org/abs/2210.11416.

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.

Dau, H. A., Bagnall, A., Kamgar, K., Yeh, C.-C. M., Zhu, Y., Gharghabi, S., Ratanamahatana, C. A., and Keogh, E. The ucr time series archive, 2019. URL https://arxiv.org/abs/1810.07758.

Dooley, S., Khurana, G. S., Mohapatra, C., Naidu, S., and White, C. Forecastpfn: Synthetically-trained zero-shot forecasting, 2023. URL https://arxiv.org/abs/2311.01933.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., and Houlsby, N. An image is worth 16x16 words: Transformers for image recognition at scale, 2021. URL https://arxiv.org/abs/2010.11929.

Edwards, T. D. P., Alvey, J., Alsing, J., Nguyen, N. H., and Wandelt, B. D. Scaling-laws for large time-series models, 2025. URL https://arxiv.org/abs/2405.13867.

Feofanov, V., Wen, S., Alonso, M., Ilbert, R., Guo, H., Tiomoko, M., Pan, L., Zhang, J., and Redko, I. Mantis: Lightweight calibrated foundation model for user-friendly time series classification. *arXiv preprint arXiv:2502.15637*, 2025. URL https://arxiv.org/abs/2502.15637.

Franceschi, J.-Y., Dieuleveut, A., and Jaggi, M. Unsupervised scalable representation learning for multivariate time series. *Advances in neural information processing systems*, 32, 2019.

Gao, S., Koker, T., Queen, O., Hartvigsen, T., Tsiligkaridis, T., and Zitnik, M. Units: A unified multi-task time series model, 2024. URL https://arxiv.org/abs/2403.00131.

Gnassounou, T., Collas, A., Flamary, R., and Gramfort, A. Psdnorm: Test-time temporal normalization for deep learning on eeg signals. *arXiv preprint arXiv:2503.04582*, 2025.

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.

Hollmann, N., Müller, S., Eggensperger, K., and Hutter, F. Tabpfn: A transformer that solves small tabular classification problems in a second, 2023. URL https://arxiv.org/abs/2207.01848.

Hoo, S. B., Müller, S., Salinas, D., and Hutter, F. The tabular foundation model tabPFN outperforms specialized time series forecasting models based on simple features. In *NeurIPS 2024 Third Table Representation Learn-*

ing Workshop, 2024. URL https://openreview.net/forum?id=H02X7RO3OC.

Jaiswal, A., Babu, A. R., Zadeh, M. Z., Banerjee, D., and Makedon, F. A survey on contrastive self-supervised learning. *Technologies*, 9(1):2, 2020.

Jin, M., Wang, S., Ma, L., Chu, Z., Zhang, J. Y., Shi, X., Chen, P.-Y., Liang, Y., Li, Y.-F., Pan, S., et al. Time-llm: Time series forecasting by reprogramming large language models. *arXiv preprint arXiv:2310.01728*, 2023.

Lai, G., Chang, W.-C., Yang, Y., and Liu, H. Modeling long- and short-term temporal patterns with deep neural networks, 2018. URL https://arxiv.org/abs/1703.07015.

Langley, P. Crafting papers on machine learning. In Langley, P. (ed.), *Proceedings of the 17th International Conference on Machine Learning (ICML 2000)*, pp. 1207–1216, Stanford, CA, 2000. Morgan Kaufmann.

Li, S., Jin, X., Xuan, Y., Zhou, X., Chen, W., Wang, Y.-X., and Yan, X. Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting, 2020. URL https://arxiv.org/abs/1907.00235.

Li, Y., Yu, R., Shahabi, C., and Liu, Y. Diffusion convolutional recurrent neural network: Data-driven traffic forecasting, 2018. URL https://arxiv.org/abs/1707.01926.

Lin, C., Wen, X., Cao, W., Huang, C., Bian, J., Lin, S., and Wu, Z. Nutime: Numerically multi-scaled embedding for large-scale time-series pretraining, 2024. URL https://arxiv.org/abs/2310.07402.

Liu, X., Zhang, F., Hou, Z., Mian, L., Wang, Z., Zhang, J., and Tang, J. Self-supervised learning: Generative or contrastive. *IEEE Transactions on Knowledge and Data Engineering*, 35(1):857–876, 2023. doi: 10.1109/TKDE.2021.3090866.

Liu, Y., Zhang, H., Li, C., Huang, X., Wang, J., and Long, M. Timer: Generative pre-trained transformers are large time series models, 2024. URL https://arxiv.org/abs/2402.02368.

Matsubara, Y., Sakurai, Y., Van Panhuis, W. G., and Faloutsos, C. Funnel: automatic mining of spatially coevolving epidemics. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 105–114, 2014.

Quan, P., Mulayim, O. B., Han, L., Hong, D., Berges, M., and Srivastava, M. Reimagining time series foundation models: Metadata and state-space model perspectives. In *NeurIPS Workshop on Time Series in the Age of Large Models*, 2024.

Rasp, S., Dueben, P. D., Scher, S., Weyn, J. A., Mouatadid, S., and Thuerey, N. Weatherbench: A benchmark data set for data-driven weather forecasting. *Journal of Advances in Modeling Earth Systems*, 12(11), November 2020. ISSN 1942-2466. doi: 10.1029/2020ms002203. URL http://dx.doi.org/10.1029/2020MS002203.

Rasul, K., Ashok, A., Williams, A. R., Ghonia, H., Bhagwatkar, R., Khorasani, A., Bayazi, M. J. D., Adamopoulos, G., Riachi, R., Hassen, N., Biloš, M., Garg, S., Schneider, A., Chapados, N., Drouin, A., Zantedeschi, V., Nevmyvaka, Y., and Rish, I. Lag-llama: Towards foundation models for probabilistic time series forecasting, 2024.

Shi, J., Ma, Q., Ma, H., and Li, L. Scaling law for time series forecasting, 2024. URL https://arxiv.org/abs/2405.15124.

Susto, G. A., Cenedese, A., and Terzi, M. Time-series classification methods: Review and applications to power systems data. *Big data application in power systems*, pp. 179–220, 2018.

Taga, E. O., Ildiz, M. E., and Oymak, S. Timepfn: Effective multivariate time series forecasting with synthetic data, 2025. URL https://arxiv.org/abs/2502.16294.

Wang, Y., Qiu, Y., Chen, P., Zhao, K., Shu, Y., Rao, Z., Pan, L., Yang, B., and Guo, C. Rose: Register assisted general time series forecasting with decomposed frequency learning. *arXiv preprint arXiv:2405.17478*, 2024.

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.

Xue, H. and Salim, F. D. Promptcast: A new prompt-based learning paradigm for time series forecasting. *IEEE Transactions on Knowledge and Data Engineering*, 2023.

Yao, Q., Yang, C.-H. H., Jiang, R., Liang, Y., Jin, M., and Pan, S. Towards neural scaling laws for time series foundation models. In *The Thirteenth International Conference on Learning Representations*, 2025.

Yue, Z., Wang, Y., Duan, J., Yang, T., Huang, C., Tong, Y., and Xu, B. Ts2vec: Towards universal representation of time series. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36(8):8980–8987, Jun. 2022. doi: 10.1609/aaai.v36i8.20881. URL https://ojs.aaai.org/index.php/AAAI/article/view/20881.

Zhang, C., Zhang, C., Song, J., Yi, J. S. K., Zhang, K., and Kweon, I. S. A survey on masked autoencoder for self-supervised learning in vision and beyond. *arXiv preprint arXiv:2208.00173*, 2022.

Zhang, H., Liu, Y., Qiu, Y., Liu, H., Pei, Z., Wang, J., and Long, M. Timesbert: A bert-style foundation model for time series understanding, 2025. URL https://arxiv.org/abs/2502.21245.

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, 2021. URL https://arxiv.org/abs/2012.07436.

Zhou, T., Niu, P., Sun, L., Jin, R., et al. One fits all: Power general time series analysis by pretrained llm. *Advances in neural information processing systems*, 36: 43322–43355, 2023.

A. Related work

Time series foundation models Recent advances in TSFM have followed two primary directions: (1) training models from scratch on large-scale, diverse time series datasets (Ansari et al., 2024; Goswami et al., 2024; Das et al., 2024; Gao et al., 2024; Rasul et al., 2024; Wang et al., 2024; Woo et al., 2024; Bhethanabhotla et al., 2024; Feofanov et al., 2025; Gao et al., 2024; Lin et al., 2024; Liu et al., 2024), and (2) leveraging large language models (LLMs) as backbones for time series tasks (Chang et al., 2023; Gruver et al., 2024; Zhou et al., 2023; Xue & Salim, 2023; Cao et al., 2023; Jin et al., 2023). The first approach focuses on developing architectures specifically tailored for time series, while the second approach explores encoding time series data into textual formats or extending the model's input mechanisms to natively handle sequential numeric data. Among the TSFMs mentioned above, a vast majority were proposed for time series forecasting, with only (Feofanov et al., 2025; Gao et al., 2024; Goswami et al., 2024; Chang et al., 2025; Lin et al., 2024; Zhang et al., 2025) natively supporting time series classification. In particular, (Feofanov et al., 2025; Lin et al., 2024) specifically target classification by contrastively pretraining encoder-only models over time series gathered from popular classification benchmarks. They achieve state-of-the-art results in this task. (Goswami et al., 2024) is an encoder-decoder model used for classification and other popular time series tasks, such as forecasting, imputation, and anomaly detection. (Gao et al., 2024) relies on a custom architecture and is used in generative and prediction tasks by leveraging task-specific tokens. Finally, (Chang et al., 2025) fine-tunes an LLM by adding an appropriate encoder for input data and a classification head to generate predictions.

Pretraining datasets The training data for TSFM generally fall into three categories: real-world, synthetic, or hybrid datasets combining the two. Models trained (or fine-tuned in case of LLM-based TSFMs) exclusively on real data (Das et al., 2024; Gao et al., 2024; Rasul et al., 2024; Wang et al., 2024; Feofanov et al., 2025; Gao et al., 2024; Lin et al., 2024; Chang et al., 2023; Gruver et al., 2024; Zhou et al., 2023; Xue & Salim, 2023; Cao et al., 2023; Jin et al., 2023) typically leverage extensive collections (ranging from 300k to 50M distinct time series) drawn from diverse domains such as traffic, finance and environmental monitoring. Training on these datasets, however, may be suboptimal scaling-wise as (Quan et al., 2024) obtained comparable performance using $< 1\%$ of the original 27B pretraining dataset from (Woo et al., 2024), while (Yao et al., 2025) showed that famous forecasting TSFMs have very flat scaling laws in the multivariate setting. Meanwhile, forecasting models such as Chronos (Ansari et al., 2024) and TimesFM (Das et al., 2024) enhance their training corpus by incorporating synthetic time series data alongside real-world data. Finally, such methods as TimePFN (Taga et al., 2025) and ForecastPFN (Dooley et al., 2023) are pre-trained solely on synthetic data. In all these forecasting models, synthetic data is commonly generated through structured statistical procedures, including Gaussian process (kernel-based) methods or piecewise linear and seasonal pattern constructions with additive noise (for more details, we refer the interested reader to Appendix B.) To the best of our knowledge, no prior work has proposed classification-oriented synthetic data generation methods for training time series foundation models.

B. Overview of pretraining datasets for time series foundation models

Table 3 summarizes the pretraining datasets used by representative Time Series Foundation Models. For each model, we report whether synthetic data was used, the total number of time points and time series samples, whether the datasets are publicly available. The table is organized alphabetically by model name.

Model	Synthetic	Real	Time Points	Series Count	Open
Chronos (Ansari et al., 2024)	Yes	Yes	84B	890K	Yes
ForecastPFN (Dooley et al., 2023)	Yes	No	60M	300K	Yes
Mantis (Feofanov et al., 2025)	No	Yes	N/A	$\sim$1.89M [1]	Yes
MOMENT(Goswami et al., 2024)	No	Yes	1.23B	13M	Yes
NuTime (Lin et al., 2024)	No	Yes	60M	1.89M	Yes
TabPFN (Hollmann et al., 2023)	Yes	No	N/A	9.216M	No
TimePFN (Taga et al., 2025)	Yes	No	$\sim$ 200M	$\sim$3M	Yes
UniTS (Gao et al., 2024)	No	Yes	35M	6K	Yes

Table 3: Overview of pretraining datasets for Time Series Foundation Models (TSFMs).

C. Loss and architecture of Mantis and MOMENT

Self-supervised pretraining Self-supervised learning (SSL) has emerged as a powerful training paradigm for foundation models, allowing them to effectively learn discriminative representations from large-scale unlabeled datasets, significantly reducing dependency on costly data labeling (Jaiswal et al., 2020). SSL methods are categorized into two principal types: contrastive learning and masked (reconstruction) learning (Liu et al., 2023). Contrastive learning focuses on distinguishing between similar (positive) and dissimilar (negative) data pairs to learn meaningful representations. Conversely, masked learning leverages reconstruction objectives by training models to predict masked parts of the input, thereby gaining robust contextual understanding (Zhang et al., 2022).

In our work, we cover both pretraining regimes. To this end, we consider Mantis (Feofanov et al., 2025), an open-source FM pretrained contrastively, and MOMENT (Goswami et al., 2024), which is a masked-based pretrained model.

Contrastive learning loss of Mantis. Given an encoder $F : \mathbb{R}^t \to \mathbb{R}^q$, we consider random augmentations $\phi, \psi \sim \mathcal{U}(\mathcal{T})$. The similarity between two augmented samples is measured after projecting their embeddings to a new dimension q' via $g : \mathbb{R}^q \to \mathbb{R}^{q'}$. Specifically, the cosine similarity is defined as:

$$s_{\cos}(\mathbf{a}, \mathbf{b}) = \frac{\mathbf{a}^\top \mathbf{b}}{\|\mathbf{a}\|\|\mathbf{b}\|}, \quad \forall (\mathbf{a}, \mathbf{b}) \in \mathbb{R}^{2q'}.$$

Given a batch $B = \{\mathbf{x}_i\}_{i=1}^b$, we compute pairwise similarities:

$$\mathbf{s}_i(\phi, \psi) = [s_{\cos}(g \circ F \circ \phi(\mathbf{x}_i), g \circ F \circ \psi(\mathbf{x}_j))]_{j=1}^b \in \mathbb{R}^b.$$

The Mantis encoder F and projector g are optimized by minimizing the contrastive loss:

$$\mathcal{L}_{\text{contrastive}} = \sum_{i=1}^{b} l_{\text{ce}} \left(\frac{\mathbf{s}_i(\phi, \psi)}{T}, i \right),$$

where l_{ce} is the cross-entropy loss and T is a temperature parameter set to 0.1.

Masked learning loss of MOMENT. Given a univariate time series $\mathcal{T} \in \mathbb{R}^{1 \times T}$, it is segmented into N disjoint patches of length P. Each patch is mapped into a D-dimensional embedding, replaced with a learnable mask embedding $[\text{MASK}] \in \mathbb{R}^{1 \times D}$ for masked patches. The resulting embeddings are fed into a transformer encoder, producing transformed embeddings that are then decoded by a lightweight reconstruction head h_{rec}. The masked loss for reconstruction is defined as the mean squared error (MSE):

$$\mathcal{L}_{\text{masked}} = \frac{1}{|\Omega|} \sum_{n \in \Omega} \|\mathcal{T}_n - h_{\text{rec}}(F([\text{MASK}]))_n\|^2,$$

where Ω denotes the set of indices corresponding to masked patches.

Model architectures. For the masked learning approach, MOMENTs leverages a Transformer-based architecture derived from the T5 family (Chung et al., 2022)model. Specifically, MOMENT employs a 8, 12, 24-layer Transformer encoder with hidden dimensions $D = 512, 768, 1024$, and 8, 12, 16 attention heads for "Small", "Base", "Large" model. The model processes input time series by segmenting them into $N = 64$ patches of length $P = 8$, applying positional embeddings, and then reconstructing masked patches.

Conversely, Mantis utilizes a Vision Transformer (ViT)(Dosovitskiy et al., 2021) architecture. Initially, the input time series is divided into tokens, to which a learnable class token is appended. Positional embeddings are added to encode temporal information explicitly. The ViT unit consists of 6 transformer layers, each comprising multi-head attention with 8 heads. The final output is derived from the class token's embedding after aggregation by the transformer layers. It is worth noting that Mantis employs a customized tokenizer. For detailed information, please refer to the original Paper[4].

[1] The updated number of training samples (∼1.38M) is confirmed in the official repository: https://github.com/vfeofanov/mantis/issues/2. The arXiv version initially reported ∼7M.

[4] https://github.com/vfeofanov/Mantis

D. Details of CAUKER

D.1. Details of banks

Specifically, the kernel banks include:

- **ExpSineSquared** — captures periodic patterns with a fixed wavelength; produces strongly oscillatory samples with global smoothness.
- **DotProduct** — induces linear trend behavior; sample paths grow or decay steadily over time.
- **RBF (Radial Basis Function)** — generates smooth, localized fluctuations around zero with short-range correlations.
- **RationalQuadratic** — a scale mixture of RBF kernels, allowing for multiscale smooth variations in the signal.
- **WhiteKernel** — models uncorrelated noise; sample paths resemble pure Gaussian noise with no temporal structure.
- **ConstantKernel** — generates flat constant signals; serves as a component for additive models with nonzero mean.

These six kernels represent only a small subset of our full kernel bank. In practice, we construct a much larger kernel bank comprising 36 distinct kernels. This is achieved by varying the hyperparameters of each kernel (e.g., length-scale, periodicity, noise level, amplitude) across a range of scales to capture diverse temporal dynamics. For instance, we use multiple versions of the ExpSineSquared kernel with different periodicities to simulate both high- and low-frequency periodic patterns. Similarly, we vary the length-scale of RBF and RationalQuadratic kernels to control smoothness and correlation range.

During synthetic data generation, kernels are sampled from the full kernel bank, which offers significantly richer diversity than what is shown here. These base kernels are subsequently composed using random additive and multiplicative operations to define flexible Gaussian process priors for root node generation in the CAUKER pipeline.

Figure 2 presents the four representative mean functions used in our synthetic data generation pipeline. Each subplot illustrates a randomly sampled instance from the corresponding function class. These functions can be combined multiplicatively or additively during Gaussian process sampling to enrich the diversity of generated signals.

- Zero Mean: A baseline function returning a constant zero across the time axis, corresponding to the standard GP assumption with zero-centered priors.
- Linear Mean: A simple affine transformation $a \cdot t + b$, enabling trends such as monotonic increases or decreases over time.
- Exponential Mean: A parametric form $a \cdot \exp(bt)$ that introduces strong, nonlinear growth or decay patterns into the signal.
- Sparse Anomalies: A piecewise-constant mean vector with a few randomly placed spikes, simulating rare disruptive events (e.g., faults, attacks, regime shifts).

These mean functions serve as building blocks for composing realistic non-stationary temporal structures in synthetic time series. In the generation process, two functions are randomly selected and combined (either by summation or elementwise multiplication), forming the final mean vector used in GP sampling. The images shown in Figure 2 are illustrative samples; in practice, stochastic variation over parameters (slopes, amplitudes, etc.) ensures that each generated series presents unique mean behavior.

Activation function bank. In addition to kernel and mean banks, CAUKER employs a diverse *activation function bank* $\mathcal{A}$ to propagate nonlinear transformations through the structural causal graph. Each edge in the DAG is randomly assigned an activation from this bank, which governs how parent node values influence their children. The activation bank comprises both classical and domain-specific transformations:

- **Linear:** Identity or affine mappings $ax + b$, preserving proportional signal propagation.

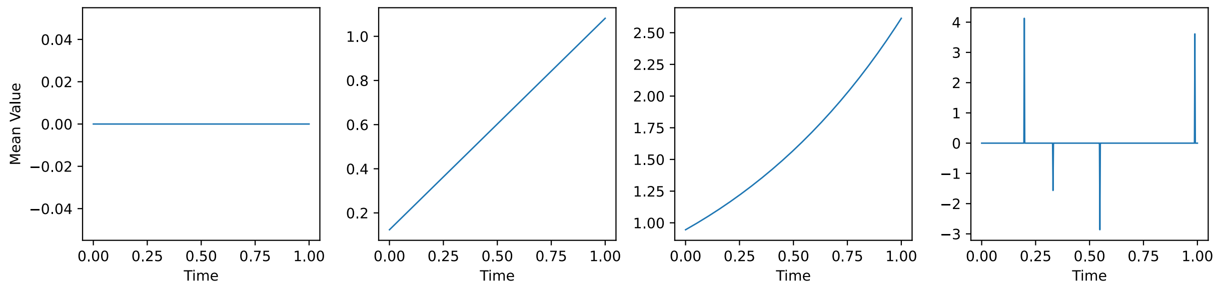

Figure 2: Examples of four mean function types used in the synthetic data pipeline. Each function introduces distinct temporal structure, contributing to the diversity and realism of generated sequences.

- **ReLU:** Rectified linear units $\max(0, x)$, introducing sparsity and piecewise linearity.
- **Sigmoid:** Smooth squashing function $\sigma(x) = 1/(1 + e^{-x})$, modeling saturation effects.
- **Sinusoidal:** Periodic modulations $\sin(x)$, inducing wave-like behaviors.
- **Modulo:** Modular transformations $x \bmod c$, yielding abrupt nonlinearities or periodic clipping.
- **Leaky ReLU:** Slope-preserving variant of ReLU, ensuring non-zero gradients for negative inputs.

These nonlinearities enhance the diversity of functional relationships within the generated synthetic time series and allow the resulting signals to exhibit complex, structured dependencies. As illustrated in the SCM pipeline, these functions are applied edge-wise to linear combinations of parent signals before assigning values to child nodes.

Design choices The synthetic datasets generated using our CAUKER approach effectively encode diverse, realistic patterns and causal dynamics characteristic of real-world classification problems. Unlike the kernel-only generator of Ansari et al. (2024) (Steps 1,2), which was designed for forecasting and therefore draws zero-mean Gaussian-process samples that emphasize smooth trend extrapolation, our task calls for retaining the mean level itself (Step 3) as a discriminative cue – a choice that is empirically confirmed in Section 3.2. Conversely, the structural causal model (SCM) generator (Steps 4,5) originally proposed for tabular classification (Hollmann et al., 2023) produces rich non-linear dependencies but lacks hallmark time series motifs such as seasonality or linear trends. By unifying kernel composition with an SCM backbone, CAUKER inherits the local smoothness and periodic structure of Gaussian processes while simultaneously injecting causal semantics through directed edges, yielding synthetic series that are explicitly classification-oriented and more faithful to real-world temporal dynamics.

E. Scaling laws for zero-shot classification with TSFMs

Scaling laws are fundamental to improving foundation models, underpinning their ability to generalize and demonstrate emergent capabilities with increased data and model scale. While scaling laws are widely studied in language and vision, their systematic exploration in the context of zero-shot time series classification remains is currently absent. To the best of our knowledge, our work is the first to thoroughly investigate scaling laws specifically in the setup of zero-shot time series classification which is of independent interest.

E.1. Data scaling laws

Experimental setup To investigate data scaling laws, we systematically vary the pretraining dataset sizes from two distinct sources: (1) randomly selected subsets of the real-world UEA benchmark (Bagnall et al., 2018) at increments of 0.1%, 1% ... 100%, and (2) synthetic data generated by our proposed CAUKER method, at varying scales from 10K up to 10M samples. We recall that both Mantis and MOMENT take as input univariate time series. This means that each channel of multivariate UEA datasets becomes a training sample, with a total of 12M channels (train set and test set combined) from 30 different datasets. Additional details are provided in Appendix F.

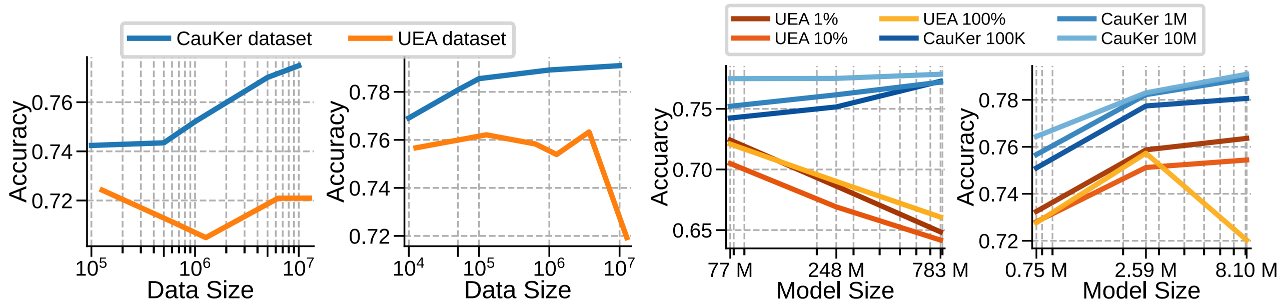

Figure 3: Scaling law of MOMENT and Mantis depending on the dataset size (**left**, **middle left**, respectively) model trained on different subsets of UEA and CauK datasets. Scaling law for the same models depending on the model size (**middle right**, **right**, respectively)

Results As illustrated in Figures 3, our experiments indicate that the classification accuracy on the UCR datasets does not monotonically increase with the size of training data when trained on subsets of the UEA dataset (left for Mantis, middle left for MOMENT). We hypothesize that this behavior may be a result of a domain mismatch between UEA and UCR, further exacerbated by the lack of diversity within the real-world time series of UEA.

In contrast, CauKer-generated datasets exhibit clear and consistent scaling laws. The accuracy steadily improves with increasing data size, demonstrating the CauKer-generated data's effectiveness in capturing diverse patterns essential for generalizing to the UCR target set. Additionally, these results also suggest an interesting contrast between model capacities: the lightweight Mantis model achieves competitive performance even with smaller training sets, likely due to the strong time series classification priors incorporated in its architecture that we have mentioned above. In contrast, the larger and more generic MOMENT model exhibits more significant accuracy gains as the training data increases, highlighting its greater capacity to leverage large-scale data for improved representation learning. This distinction underscores the importance of jointly considering model capacity and data availability when designing scalable TSFMs.

E.2. Model scaling laws

Experimental setup We further assessed model scaling laws by varying the size of the MOMENT model (Small, Base, Large versions of sizes 77M, 248M, and 783M, respectively), and Mantis model (with number of parameters 0.75M, 2.59M, 8.10M) using both UEA and CauKer-generated datasets. More details on the experiments can be found in Appendix G.

Results Results, as shown in Figure 3 (middle right for Mantis, right for MOMENT), indicate that models trained on real-world UEA data do not exhibit consistent performance gains with increasing model size, reinforcing the notion of limited data diversity or domain mismatch. Conversely, models trained on CauKer-generated datasets consistently demonstrate increased accuracy as model size grows, clearly validating the presence of model scaling laws enabled by the synthetic CauKer-generated pretraining data. We further notice that, apart from the single outlier of MOMENT trained on the 10M samples CauKer corpus, every model pretrained on CauKer exhibits a strictly increasing UCR accuracy as its capacity grows. The small increase for MOMENT at 10M indicates that this particular encoder has reached (or is close to) saturation; a similar saturation point can be observed for Mantis once the parameter count exceeds approximately 28M (see Appendix G for a more large-scale experiment). Conversely, the unstable – or even degrading – trend on models pretrained with larger UEA subsets is most plausibly explained by two factors: (i) the UEA collection lacks a clean, easily learnable generative structure, and (ii) its underlying distribution is mismatched with that of UCR, making additional capacity harder to exploit.

E.3. Training time scaling laws

We now study the training time scaling law that aims at identifying the gains in terms of test accuracy that more compute given by longer optimization of the model can bring.

Experimental setup We track the evolution of zero-shot accuracy with training epochs for Mantis and MOMENT pretrained on two corpora, namely a 10% subset of the real-world UEA benchmark and a synthetic set of 1M series generated by CauKer.

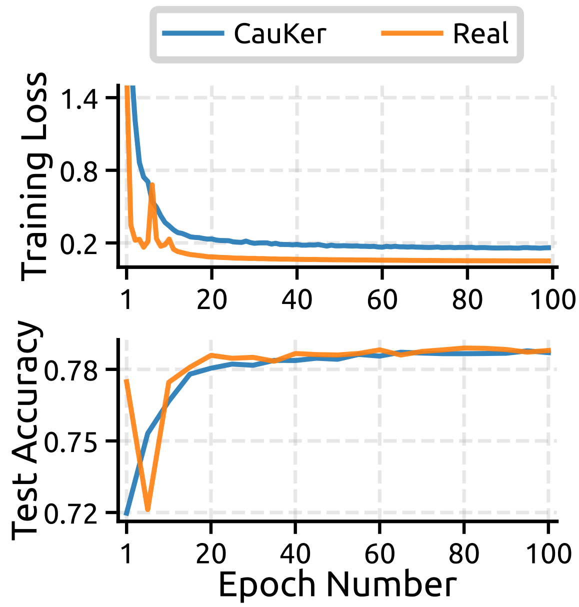

Figure 5: The figure show that synthetic data is harder to train on, but leads to a smoother increase of the test accuracy across epochs.

Results As illustrated in Figure 4, accuracy rises steadily when the models are trained on CAUKER; additional epochs translate into consistent gains for both architectures. When pretrained on UEA, however, accuracy curves remain flat or fluctuate, especially for MOMENT, indicating that prolonged optimisation yields little benefit on this dataset. These findings echo the data- and model-scaling observations reported earlier: causally structured, diverse CAUKER data sustains learning over long horizons.

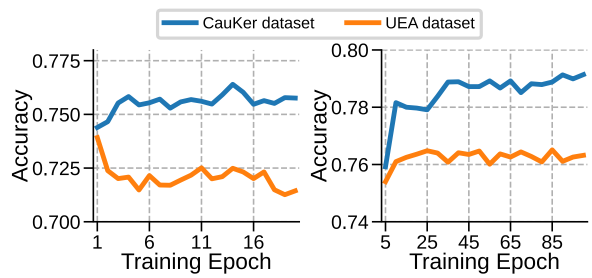

Figure 4: Test accuracy across epochs for MOMENT (**left**) and Mantis (**right**).

E.4. Comparison with forecasting scaling laws

We conclude this section by relating our obtained results to those provided for the time series forecasting task. To this end, we note that our empirical insights differ from prior work (Edwards et al., 2025; Yao et al., 2025; Shi et al., 2024) in several ways. First, while we observed clear data- and model-scaling trends when pretraining on CAUKER data, we also found signs of saturation at high data volumes or model capacities. Although (Edwards et al., 2025; Yao et al., 2025) reported a rather flat scaling law for real-world multivariate TSFMs, they were still monotonically decreasing. Second, our observed accuracy improvements follow sub-exponential rather than clean exponential growth. This suggests that the scaling dynamics in time series classification may follow different patterns compared to other modalities like language or vision, and that a more systematic, theory-driven study of such behavior is needed to fully understand its implications.

F. Experimental details of Section E.1

In our scaling law experiments, we systematically evaluated the performance of two distinct models, Mantis and MOMENT, across varying dataset sizes from both real-world and synthetic sources. We adopted the official 8M parameters configuration of Mantis as released in its open-source repository, which includes a 6-layer ViT encoder with 8 attention heads and a hidden dimension of 256. The classification head used was a Random Forest classifier trained on frozen embeddings.

For MOMENT, we used the officially supported “google/flan-t5-small” variant containing 77M parameters as the encoder backbone. This model structure is one of the pretrained configurations endorsed in the original MOMENT framework. During training, we froze the encoder and trained only the classification head, which was implemented as a Support Vector

Machine (SVM). This setup mirrors the zero-shot classification evaluation protocol used in prior TSFM literature.

For both models, we varied the training data sizes as follows: for the real-world UEA dataset, subsets ranging from 0.1% to 100% (12.7K to 12.67M samples) were randomly sampled. For synthetic data, we generated samples using our CauKer method at 10K, 50K, 100K, 500K, 1M, 5M, and 10M scales. All series were univariate with length 512. The full list of data sizes and corresponding classification accuracy values on the UCR benchmark are reported in Table 4.

Model	**Train Set**	**Data Size**	**UCR Accuracy (%)**
MOMENT (77M)	UEA	127K	72.42
	UEA	1.27M	70.49
	UEA	633K	71.09
	UEA	6.33M	72.09
	UEA	12.67M	72.10
	CauKer	100K	74.24
	CauKer	500K	74.35
	CauKer	1M	75.21
	CauKer	5M	77.01
	CauKer	10M	77.49
Mantis (8M)	UEA	12.7K	75.67
	UEA	127K	76.21
	UEA	633K	75.83
	UEA	1.27M	75.39
	UEA	3.68M	76.33
	UEA	12.67M	71.93
	CauKer	10K	76.91
	CauKer	50K	78.08
	CauKer	100K	78.55
	CauKer	1M	78.91
	CauKer	10M	79.09

Table 4: Exact accuracy values used in the scaling law plots (Figure 3).

G. Experimental details of Section E.2

To investigate model scaling laws, we evaluated a range of model capacities for both MOMENT and Mantis using synthetic datasets generated by CauKer. For MOMENT, we adopted the official series of models given by:

- **flan-t5-small** (77M parameters),
- **flan-t5-base** (248M parameters),
- **flan-t5-large** (783M parameters).

For the Mantis encoder, we varied the transformer depth and width while keeping the sequence length fixed at 512 and using the same patching configuration. The model variants are as follows:

- **0.75M**: `hidden_dim=256`, `transf_depth=1`, `transf_num_heads=2`, `transf_mlp_dim=512`, `transf_dim_head=128`.
- **2.59M**: same as above, with `transf_depth=3`, `transf_num_heads=4`.
- **8.10M**: same as above, with `transf_depth=6`, `transf_num_heads=8`.

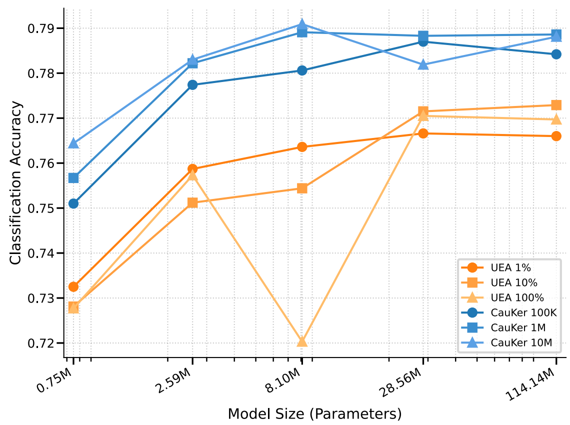

Figure 6: Accuracy on UCR dataset with varying model sizes for the Mantis model trained on UEA subsets and synthetic CAUKER data.

- **28.56M**: same as above, with `transf_depth=12`, `transf_num_heads=16`.
- **114.14M**: `hidden_dim=512`, `transf_depth=12`, `transf_num_heads=16`, `transf_mlp_dim=1024`, `transf_dim_head=256`.

All Mantis variants used the following fixed parameters: `seq_len=512`, `num_patches=32`, `scalar_scales=None`, `hidden_dim_scalar_enc=32`, and `epsilon_scalar_enc=1.1`. The model output embeddings were classified using a Random Forest classifier trained on frozen features.

This design allows us to jointly assess the impact of model depth, width, and hidden dimensionality on zero-shot classification performance under a consistent synthetic data regime.

Table 5 reports the exact accuracy values corresponding to the model scaling plots shown in Figure 6. For both MOMENT and Mantis, we list results under varying model sizes and dataset configurations.

Model Size	UEA 1%	UEA 10%	UEA 100%	CAUKER 100K	CAUKER 1M	CAUKER 10M
77M (MOMENT)	72.42	70.49	72.10	74.24	75.21	77.49
248M (MOMENT)	68.62	66.91	69.01	75.16	76.16	77.51
783M (MOMENT)	64.85	64.18	66.07	77.28	77.20	77.85
0.75M (Mantis)	73.25	72.81	72.77	75.10	75.67	76.44
2.59M (Mantis)	75.87	75.12	75.73	77.74	78.22	78.30
8.10M (Mantis)	76.36	75.44	72.03	78.06	78.91	79.09
28.56M (Mantis)	76.66	77.15	77.05	78.70	78.83	78.19
114.14M (Mantis)	76.60	77.29	76.97	78.42	78.86	78.81

Table 5: Exact zero-shot accuracy (%) on the UCR benchmark under different model sizes and pretraining dataset configurations.

H. Experimental details of Section 3.2

For all compared models, we adopted the best training loss epoch as the checkpoint for final evaluation. Specifically, the official setting for Mantis involves training for 100 epochs, while MOMENT is typically trained for 2 epochs. However, for our experiments, we trained Mantis for 100 epochs and MOMENT for 10 epochs to allow sufficient convergence, consistent with our goal of achieving the best performance on the CAUKER and UEA datasets. For the MOMENT model, we utilized the base model ”google/flan-t5-small” with 77M parameters, trained on both the CauKer and UEA datasets. The official MOMENT checkpoint used in our experiments (Time Series Pile), ”google-t5/t5-small,” has 60M parameters.