Are Time Series Foundation Models Ready for Zero-Shot Forecasting?

Anonymous Authors[1]

Abstract

Recent successes of foundation models in various domains have spurred interest in time series foundation models (TSFMs), especially for zero-shot forecasting. We challenge the necessity of zero-shot forecasting, and demonstrate that a simpler model, PCA+Linear, can effectively serve as a TSFM. PCA+Linear uses Principal Component Analysis (PCA) as a universal feature extractor and a linear head trained specifically on each downstream dataset. Experiments show PCA+Linear achieves results competitive with state-of-the-art TSFMs. We further evaluate the robustness of transformer-encoder-based TSFMs on out-of-distribution data, highlighting the importance of the final linear layer in addition to the attention mechanism. Our findings also emphasize the effectiveness of diverse pretraining data over extensive datasets from limited sources.

1. Introduction

The success of foundation models in Natural Language Processing and Computer Vision has inspired a growing body of work on developing *time series foundation models* (TSFMs) (Das et al., 2024; Ansari et al., 2024; Ekambaram et al., 2024; Liu et al.; Woo et al., 2024; Darlow et al., 2024). Unlike traditional approaches that train separate models for each dataset or task, a time series foundation model (TSFM) is trained on a large-scale corpora of time series datasets, often from multiple domains. By learning patterns transferable across domains, a single TSFM can be applied to different downstream datasets to solve various tasks. Among these tasks, most work focuses on benchmarking and improving zero-shot forecasting ability of TSFMs. Zero-shot forecasting means that, given past time steps in the form of a context window, we do not alter the trained model weights, and directly apply inference to generate predictions.

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

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

However, we argue that the practical necessity for zero-shot forecasting in real-world applications is often overstated. Large Language Models (LLMs) focus on zero-shot learning since they directly face the end users, and thus require a low level of latency. Conversely, most time series forecasting applications either (i) have plenty of downstream data available if they require a low latency due to the high frequency of the dataset (e.g., stock trading (Cao et al., 2024) or wearable sensors (Zhang et al., 2023)), or (ii) have limited downstream data available, but it is often due to the low frequency of data collection, which inherently allows for higher latency as decisions are less time-critical.

Motivated by this observation, we propose an alternative: pretrain a PCA as a universal feature extractor, and repurpose the context window itself to train a lightweight linear head based on the features extracted by PCA. By incorporating key design components from TSFMs, we can achieve performance on par with state-of-the-art TSFMs. The only notable overhead is that the linear layer could take up to a minute to train and tune its hyperparameters, but it should be affordable in most practical applications where data are limited. Moreover, we conduct a thorough robustness analysis of two transformer-encoder based TSFMs variants on out-of-distribution (OOD) downstream data.

Contributions. We introduce PCA+Linear as a simple, lightweight baseline for zero-shot forecasting and demonstrate its competitive performance relative to contemporary TSFMs. Additionally, we analyze the OOD performance of two transformer-encoder based TSFMs, and show that the final linear layer, which was previously shown to be critical for transformer-based time series models (Li et al., 2024; Tan et al., 2024), might still be critical for generalizability in the age of time series foundation models. Our findings also suggest that incorporating diverse data sources during pretraining is more beneficial than merely expanding data volume from limited sources.

2. Can PCA+Linear also be a TSFM?

In real-world settings, the number of time steps available in the history often varies widely, while the number of time steps that we need to predict into the future also varies due to different business needs. For example, hourly energy

consumption (Lai et al., 2018; Trindade, 2015) can easily accumulate tens of thousands of time steps, and the goal is often to predict several hundred time steps into the future. On the other hand, for daily strawberry yield, we may hardly gather more than several hundred time steps, and the desired forecast horizon is also much shorter. To enable zero-shot forecasting in various downstream scenarios, time series foundation models must be capable of handling a flexible context window length, as well as a flexible forecast horizon.

We argue that a simple PCA model coupled with a domain-specific linear layer head can also satisfy the two requirements above. Specifically, PCA is used as the universal encoder to extract features, while a linear layer is used to transform the features into the predictions.

2.1. Patch-Level Pretraining

Inspired by the recent success of grouping adjacent time steps into patches in deep learning methods (Nie et al., 2023), we divide the context window of arbitrary length into patches and apply PCA to each individual patch. In the pretraining stage, analogous to time series foundation models, we fit the PCA model on the patches extracted from various pretraining datasets. For multivariate time series, we process each variable independently, following DLinear (Zeng et al., 2023). Specifically, given a multivariate time series $\mathbf{X} \in \mathbb{R}^{T \times d}$, we divide it into $n = \left\lfloor \frac{T}{p} \right\rfloor$ non-overlapping patches $\mathbf{X}_i \in \mathbb{R}^{p \times d}$. For each patch $\mathbf{X}_i$, and each variable $j = 1, \ldots, d$, we extract the univariate segment $\mathbf{x}_i^{(j)} = \mathbf{X}_i[:, j] \in \mathbb{R}^p$, and apply PCA:

$$\mathbf{z}_i^{(j)} = \mathbf{W}^{(j)\top} \left(\mathbf{x}_i^{(j)} - \boldsymbol{\mu}^{(j)} \right), \quad \mathbf{W}^{(j)} \in \mathbb{R}^{p \times k}$$

where $\mathbf{W}^{(j)}$ and $\boldsymbol{\mu}^{(j)}$ are the PCA projection matrix and mean vector respectively, learned during pretraining. We then concatenate the reduced embeddings $\mathbf{z}_i^{(j)}$ across the context window to form the final embedding:

$$\mathbf{z}^{(j)} = \text{concat} \left(\mathbf{z}_1^{(j)}, \mathbf{z}_2^{(j)}, \ldots, \mathbf{z}_n^{(j)} \right) \in \mathbb{R}^{k \cdot n}$$

This patch-level PCA embedding serves as the input representation for downstream tasks.

2.2. Any-Horizon Decoding

To generate predictions on each downstream dataset, instead of using the entire context window for zero-shot inference, we use it as both the training set and the validation set for a linear head specific to the dataset. To train the linear head, we use a shorter context window than the original context window for zero-shot inference. Additionally, we recognize several key design components from recent time series foundation models are also applicable to linear models, and thus incorporate them into our framework. To allow for flexible forecast horizon, we generate the predictions autoregressively in a decoder fashion. Since autoregressive predictions are prone to error accumulation (Zeng et al., 2023), we follow DecOnly (Das et al., 2024) and let the linear head predict more time steps than the input patch size at once.

2.3. Pretraining Details

For fair comparisons with state-of-the-art TSFMs, we follow their procedures by first fitting the PCA model on selected datasets[1] from LOTSA (Woo et al., 2024), DAM (Darlow et al., 2024), and Monash (Godahewa et al., 2021). Since the datasets and even the different variables in the same dataset can differ widely in terms of magnitude, we follow Moirai's approach to assign a sampling weight to each variable in the dataset depending on its number of available time steps. Additionally, instead of loading the entire LOTSA corpus into memory, we first determine the dataset and the variable to sample from based on the sampling weights. However, loading the entire sequence can still be overly time-consuming. Instead, we use Numpy to access it as a memory-mapped file, which is useful for accessing small segments of large files on disk, without reading the entire file into memory. We then determine the appropriate window to select, before finally loading the selected window into memory. This procedure drastically speeds up the dataloader, enabling the use of larger pretraining datasets within the memory constraints[2]. This is particularly useful when we train large TSFMs in Section 3. However, for PCA, we need to load all the patches into the memory at once in order to perform necessary matrix manipulations, so we presample 10,000 windows of 2016 steps each. In practice, we found the gain from additional pretraining windows to be marginal for PCAs.

2.4. Experiment Results

We first compare PCA+Linear with Moirai (Woo et al., 2024) on the ETT, Electricity, and Weather datasets. In Figure 1, instead of using a context window of $100-5000$ steps for zero-shot forecasting as in MOIRAI, we use it as the training and validation set to train a linear layer from scratch, as described in Section 2.2. On the original validation set of both Electricity and Weather, we see that PCA+Linear is able to consistently outperform Moirai across the majority of context window sizes. Notably, with as few as 100 timesteps, PCA+Linear is able to achieve significantly lower

[1] We filter out datasets that do not appear to be predictive, as well as subsampling overly large dataset due to limited budget. See Appendix A.

[2] See Appendix B.1.

Dataset	PCA+Linear	GP	ARIMA	TCN	llmtime	DecOnly	NAIVE
AirPassengers	30.78	34.67	24.03	54.96	34.37	14.75	81.45
AusBeer	24.11	102.05	17.13	30.90	16.13	10.25	96.35
GasRateCO2	2.20	2.27	2.37	2.64	3.50	2.69	2.29
MonthlyMilk	24.27	30.33	37.19	70.86	9.68	22.46	85.71
Sunspots	61.01	53.74	43.56	51.82	47.34	50.88	48.24
Wine	2882.18	4552.06	2306.70	3287.14	1569.32	2462.11	4075.28
Wooly	478.85	649.98	588.78	1158.79	808.73	917.10	1210.33
HeartRate	5.30	5.65	5.56	5.49	6.21	5.44	5.92
Arithmetic Mean	0.6144	0.8193	**0.6045**	0.8427	0.6641	0.6829	1
Geometric Mean	**0.5065**	0.7509	0.5219	0.7946	0.4882	0.5767	1

Table 1: MAE on the Darts Datasets. To compute the arithmetic and geometric means, we first scale each MAE by dividing by the corresponding MAE of the naive baseline, which simply predicts the last values in the context window repeatedly.

MAE than Moirai on both datasets. For the hourly Electricity dataset, 100 timesteps are just over four days of data. The full results on all six datasets are shown in Appendix Table 6.

Next, we compare PCA+Linear against another state-of-the-art TSFM, DecOnly (Das et al., 2024). We perform the same experiments on the Darts datasets (Herzen et al., 2022). We refer readers to DecOnly (Das et al., 2024) for more detailed experiment settings and baseline descriptions. The results presented in Table 1 show that PCA+Linear achieves the best geometric mean and is second only to ARIMA in terms of arithmetic mean. It outperforms established large foundation models like llmtime (Nate Gruver & Wilson, 2023) and DecOnly. We also show similar observations on the ETT datasets in Appendix Table 5.

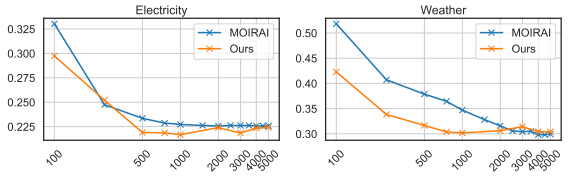

Figure 1: MAE of PCA+Linear against Moirai on increasing context length with a prediction length of 96 on the validation sets of Electricity and Weather.

3. Zero-Shot Evaluation of TSFMs

To assess the practical readiness of existing time series foundation models (TSFMs) for zero-shot forecasting, we analyze the performance of two transformer variants. The first variant, *Encoder-Only*, involves dividing the context window into patches, padding the forecast horizon with mask tokens, and feeding both into a transformer encoder. We linearly project the output embeddings of the mask tokens to get the predictions. The second variant, *Encoder-Flat*, is similar to PatchTST (Nie et al., 2023). We only feed the context window into the encoder-only transformer without the mask tokens, flatten the output embeddings, and linearly project them to get the final predictions. We use the first 80% of each dataset for pretraining, and the last 20% for validation.

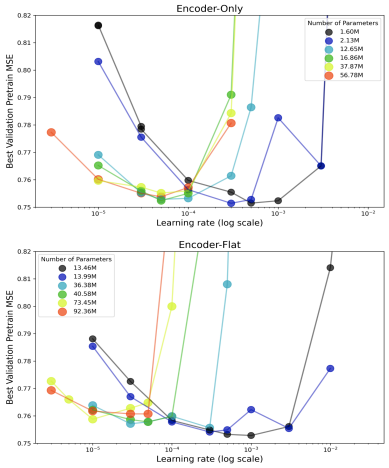

Figure 2: Optimal learning rates for different model sizes.

3.1. Optimal Learning Rates Across Model Scales

We explore six different model sizes for both Encoder-Only and Encoder-Flat architectures, spanning from 1.6 million to 92.4 million parameters (see Appendix A.2). Following the procedure of Edwards et al. (2025), we first determine the optimal learning rate for each model size. Figure 2 shows validation MSE across different learning rates. Note that the MSE reported here is considered to be in-distribution, since we use the same datasets for validation as we did during pretraining.

For all sizes, validation MSE decreases as the learning rate increases, until beyond a certain threshold the model di-

Model	30% Datasets		10% Datasets		30% Timesteps		10% Timesteps	
	%MSE↑	%MAE↑	%MSE↑	%MAE↑	%MSE↑	%MAE↑	%MSE↑	%MAE↑
Encoder-Flat	**12.71%**	**8.14%**	42.82%	**25.06%**	**6.28%**	**5.05%**	**17.72%**	12.83%
Encoder-Only	14.77%	8.52%	**42.06%**	27.06%	8.07%	5.17%	18.59%	**11.58%**

Table 2: Comparison on the number of data sources versus the number of timesteps per data source.

verges. We also find that smaller models generally require higher learning rates than larger models. For both architectures, we find that the in-distribution validation performance varies only slightly with model size under the optimal learning rate. We note that a recent work (Yao et al., 2025) shows that parameterizing a student-t distribution instead of directly predicting the mean as the loss objective can be helpful to stabilize the convergence, due to the many outliers in the pretraining corpus. This will be left as a next step of our work.

3.2. Out-Of-Distribution Performance

Using optimal learning rate we identified in Section 3.1, we evaluate each of the model sizes on the widely recognized long-sequence prediction benchmark (Wu et al., 2022). We first verify the correctness of our implementation by comparing it with Moirai (Woo et al., 2024). The results are shown in Table 7 and Table 8 in the Appendix. Next, in Figure 3, we plot the percentage increase in MAE and MSE, scaled using the minimum MAE or MSE across the six sizes and two architectures.

We find that as we increase the model size for Encoder-Flat, the OOD performance generally improves. However, this pattern is less consistent on Encoder-Only. This discrepancy might stem from previous observations (Li et al., 2024) indicating that linear layers significantly influence large transformer models' performance in time series forecasting, and the flattening operation in Encoder-Flat allows the final linear layer to access the entire context window.

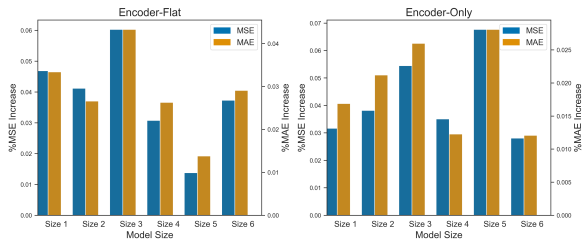

Figure 3: Performance on the OOD downstream datasets.

3.3. Insights for Expanding Pretraining Data

Lastly, increasing the size of pretraining corpora is known to be crucial for foundation models. Therefore, we want to provide some insights on what to focus on when expanding the pretraining data. We compare two strategies for data expansion: (1) adding more distinct data sources versus (2) gathering more timesteps from each data source. To answer this question, we randomly choose 30% and 10% from all the pretraining datasets. For comparison, for each pretraining dataset, we only use the first 30% and 10% of the available timesteps. Due to the large variance when subsampling the datasets, we repeat the experiment three times and report the average performance across the forecast horizons $\in \{96, 192, 336, 720\}$ on the OOD downstream datasets. The results are shown in Table 2.

Our results show that incorporating a greater diversity of data sources yields substantially better OOD performance than simply increasing the number of timesteps within the same sources. This might imply that the foundation model is able to quickly generalize a pattern without having to see similar patterns repeatedly. However, when confronted with patterns that deviate strongly from the pretraining distribution, it can fail to generalize well to this new pattern. Additionally, we observe that Encoder-Flat is more robust to the limited sizes of the pretraining data than Encoder-Only. We suspect that allowing the linear layer to access the entire context window might be critical for it to generalize well, although this comes at the cost of many extra weight parameters.

4. Conclusion

The conventional expectations of a time series foundation model are the abilities to handle flexible input length and predict an arbitrary number of steps into the future. For the popular zero-shot forecasting task, we present a simple baseline with PCA serving as the role of a foundation model for feature extraction and a linear head as the predictor. We show that it can achieve comparable performances against state-of-the-art TSFMs. Additionally, we benchmark the OOD performance of TSFMs based on two transformer-encoder variants, and show that the final linear head might still play a critical role in the generalization ability of TSFMs. Our findings also underline the importance of diverse data sources in pretraining over extensive data volume from limited sources.

References

Ansari, A. F., Stella, L., Turkmen, C., Zhang, X., Mercado, P., Shen, H., Shchur, O., Rangapuram, S. S., Pineda Arango, S., Kapoor, S., Zschiegner, J., Maddix, D. C., Wang, H., Mahoney, M. W., Torkkola, K., Gordon Wilson, A., Bohlke-Schneider, M., and Wang, Y. Chronos: Learning the language of time series. *arXiv preprint arXiv:2403.07815*, 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. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=YH5w12OUuU.

Darlow, L. N., Deng, Q., Hassan, A., Asenov, M., Singh, R., Joosen, A., Barker, A., and Storkey, A. DAM: Towards a foundation model for forecasting. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=4NhMhElWqP.

Das, A., Kong, W., Sen, R., and Zhou, Y. A decoder-only foundation model for time-series forecasting. In *Forty-first International Conference on Machine Learning*, 2024. URL https://openreview.net/forum?id=jn2iTJas6h.

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.

Ekambaram, V., Jati, A., Dayama, P., Mukherjee, S., Nguyen, N. H., Gifford, W. M., Reddy, C., and Kalagnanam, J. Tiny time mixers (ttms): Fast pre-trained models for enhanced zero/few-shot forecasting of multivariate time series, 2024. URL https://arxiv.org/abs/2401.03955.

Godahewa, R. W., Bergmeir, C., Webb, G. I., Hyndman, R., and Montero-Manso, P. Monash time series forecasting archive. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021. URL https://openreview.net/forum?id=wEc1mgAjU-.

Herzen, J., Lässig, F., Piazzetta, S. G., Neuer, T., Tafti, L., Raille, G., Van Pottelbergh, T., Pasieka, M., Skrodzki, A., Huguenin, N., et al. Darts: User-friendly modern machine learning for time series. *Journal of Machine Learning Research*, 23(124):1–6, 2022.

Jin, M., Wang, S., Ma, L., Chu, Z., Zhang, J. Y., Shi, X., Chen, P.-Y., Liang, Y., Li, Y.-F., Pan, S., and Wen, Q. Time-LLM: Time series forecasting by reprogramming large language models. In *International Conference on Learning Representations (ICLR)*, 2024.

Lai, G., Chang, W.-C., Yang, Y., and Liu, H. Modeling long- and short-term temporal patterns with deep neural networks. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, SIGIR '18, pp. 95–104, New York, NY, USA, 2018. Association for Computing Machinery. ISBN 9781450356572. doi: 10.1145/3209978.3210006. URL https://doi.org/10.1145/3209978.3210006.

Li, Z., Qi, S., Li, Y., and Xu, Z. Revisiting long-term time series forecasting: An investigation on affine mapping, 2024. URL https://openreview.net/forum?id=T97kxctihq.

Liu, Y., Zhang, H., Li, C., Huang, X., Wang, J., and Long, M. Timer: Generative pre-trained transformers are large time series models. In *Forty-first International Conference on Machine Learning*.

Nate Gruver, Marc Finzi, S. Q. and Wilson, A. G. Large Language Models Are Zero Shot Time Series Forecasters. In *Advances in Neural Information Processing Systems*, 2023.

Nie, Y., H. Nguyen, N., Sinthong, P., and Kalagnanam, J. A time series is worth 64 words: Long-term forecasting with transformers. In *International Conference on Learning Representations*, 2023.

Tan, M., Merrill, M. A., Gupta, V., Althoff, T., and Hartvigsen, T. Are language models actually useful for time series forecasting? In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=DV15UbHCY1.

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

Woo, G., Liu, C., Kumar, A., Xiong, C., Savarese, S., and Sahoo, D. Unified training of universal time series forecasting transformers. *arXiv preprint arXiv:2402.02592*, 2024.

Wu, H., Hu, T., Liu, Y., Zhou, H., Wang, J., and Long, M. Timesnet: Temporal 2d-variation modeling for general time series analysis. *arXiv preprint arXiv:2210.02186*, 2022.

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. URL https://openreview.net/forum?id=uCqxDfLYrB.

Zeng, A., Chen, M., Zhang, L., and Xu, Q. Are transformers effective for time series forecasting? In *Proceedings of the Thirty-Seventh AAAI Conference on Artificial Intelligence and Thirty-Fifth Conference on Innovative Applications of Artificial Intelligence and Thirteenth Symposium on Educational Advances in Artificial Intelligence*, AAAI'23/IAAI'23/EAAI'23. AAAI Press, 2023. ISBN 978-1-57735-880-0. doi: 10.1609/aaai.v37i9.26317. URL https://doi.org/10.1609/aaai.v37i9.26317.

Zhang, X., Chowdhury, R. R., Zhang, J., Hong, D., Gupta, R. K., and Shang, J. Unleashing the power of shared label structures for human activity recognition. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*, pp. 3340–3350, 2023.

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

A. Experiment Settings

We collect our pretraining data from three sources: LOTSA (Woo et al., 2024), DAM (Darlow et al., 2024), and Monash (Godahewa et al., 2021).

LOTSA is a large-scale, open archive of time series datasets, containing over 17 billion timesteps across 39 datasets. Due to computational constraints, we subsample and only use the first 5% of some overly large datasets: Azure VM Traces 2017, Borg Cluster Data 2011, Buildings900K, Residential Load Power. For CMIP6 and ERA5, we only use the first 5% of CMIP6_1850, CMIP6_2010, ERA5_1989, and ERA5_2018. We exclude the PEMS datasets under LibCity and the LargeST dataset since they contain data very similar to the traffic dataset that is reserved for OOD evaluation.

For Monash, we manually examine and filter out datasets that do not appear to be predictable. For DAM, we only select datasets that are not already included in LOTSA and Monash: UCI Power, HuaweiPubM, HuaweiPvtM, UCI Metro Traffic Volume, UCI Tetouan City Power, UCI Beijing PM2.5 Data, UCI Beijing Air Quality, and UCI Air Quality. Notably, we remove datasets related to stocks since they are not predictable.

All experiments are conducted on NVIDIA 3090, 4090, and A100 GPUs.

A.1. PCA + Linear

For a given context window, we use a smaller context window $\in \{96, 144, 192, 240, 288, 336, 512\}$ to train the linear layer. The prediction patch length, or the output size of the linear layer, is a multiplier of the training context window size $\in \{0.1, 0.25, 0.5, 0.75, 1\}$. We reserve the last 20% of the original given context window for validation. We use an input patch length of 24, and tune the number of principal components $\in \{15, 17, 19, 21, 23, 24\}$.

Note that only using the context window for the training and validation set is different from the few-shot forecasting setting in prior works (Jin et al., 2024; Zhou et al., 2023), where the first 5% or 10% of the original train set is used as the new train set and the entire validation set is used. We believe that our setting better aligns with the practical scenarios of limited data settings, where both the train set and the validation set need to scale down, and need to come immediately before the test set.

A.2. Encoder-Only & Encoder-Flat

Patch size and stride size are both 32, and dropout is set to 0.1. The maximum length of context window and forecast horizon are set to 2016 and 736, respectively, resulting in 63 and 23 patches.

	Size 1	Size 2	Size 3	Size 4	Size 5	Size 6
num_layers	3	4	6	8	8	12
d_model	256	256	512	512	768	768
d_ff	512	512	1024	1024	1536	1536
num_heads	8	8	16	16	24	24
Encoder–Only	1.60 M	2.13 M	12.65 M	16.86 M	37.87 M	56.78 M
Encoder–Flat	13.46 M	13.99 M	36.38 M	40.58 M	73.45 M	92.36 M

Table 3: The list of hyperparameters in the six model sizes for Encoder-Only and Encoder-Flat, along with the total parameter counts.

B. Model Details

B.1. Pretrain Dataset Weighting and Sampling Procedure

Let $\mathcal{D} = \{\mathbf{X}_1, \ldots, \mathbf{X}_M\}$ be the collection of (uni- or multi-variate) time-series datasets for pretraining. For each dataset $\mathbf{X}_d$, let it contain variables $x_{d,1}, \ldots, x_{d,n_d}$. We first store each variable as a separate file. Let $T_{d,i}$ be the number of timesteps in $x_{d,i}$ in the corresponding split (train or validation), so that the total number of timesteps in the dataset $\mathbf{X}_d$ is

$T_d = \sum_{i=1}^{n_d} T_{d,i}$. We first compute a raw dataset weight

$$\tilde{w}_d = \min\Big(10^{-3}, \frac{T_d}{\sum_{d'=1}^{M} T_{d'}}\Big).$$

Here, we cap the weight of any dataset to 10^{-3} in order to prevent some large datasets to dominate training. Next, we renormalize the weights across all datasets:

$$w_d = \frac{\tilde{w}_d}{\sum_{d'=1}^{M} \tilde{w}_{d'}}.$$

Within each dataset, we further split weight by variable:

$$w_{d,i} = w_d \times \frac{T_{d,i}}{T_d}, \qquad \sum_{d=1}^{M}\sum_{i=1}^{n_d} w_{d,i} = 1.$$

During pretraining we repeatedly draw one variable from a dataset according to the assigned weights $w_{d,i}$. Then, we sample a training window of total length $L + H$ (context L, forecast horizon H) as follows:

1. Let $T = T_{d,i}$. Uniformly pick a starting index

$$s \sim \mathrm{Unif}\{0, \ldots, T - (L + H)\}.$$

2. If the context or the forecast horizon contains fewer than two unique values (which we observe is quite common in some variables of certain datasets), we reject and resample a different window.

Note that in this process, since even a single variable can be very long in some datasets, we access the variable as a memory-mapped file and only load the window into the memory, allowing us to support much larger pretrain datasets given limited compute resources. If 20 consecutive rejections occur, we skip this variable and move on to the next variable.

B.2. Encoder-Only and Encoder-Flat

We explain the architectures of Encoder-Only and Encoder-Flat in more detail. An overview is illustrated in Figure 4.

Each input begins as a univariate time series window sampled according to the procedures described in Section B.1. This window includes both a context segment and a forecast horizon. Prior to feeding the data into the transformer, we apply instance normalization to the context window, followed by patching, which divides the context into fixed-length patches.

For the Encoder-Only architecture, we append special [MASK] tokens to represent each patch in the forecast horizon. The combined sequence of context patches and [MASK] tokens is then fed into a transformer encoder. After processing, we extract the output embeddings corresponding only to the [MASK] positions. Each of these embeddings is passed through a linear layer to reconstruct the associated patch in the forecast horizon.

For the Encoder-Flat architecture, we process only the context patches without any [MASK] tokens. All output embeddings from the transformer are flattened and concatenated into a single vector. A single linear layer is then applied to this flattened representation to predict the entire forecast horizon in one step.

C. Additional Experiment Results

C.1. Ablation Study of PCA+Linear

We conduct an ablation study on the Darts datasets (Herzen et al., 2022). We compare our full PCA+Linear model against two ablated variants:

- *w/o PCA*: raw timesteps are fed directly into the linear head instead of the features extracted by PCA, and
- *w/o pretrain*: PCA is fitted only on each context window rather than on the larger pretraining dataset.

Table 4 shows that both applying PCA and pretraining it consistently improve performance. PCA is particularly beneficial for dimension reduction, especially in the Darts datasets where the number of available time steps is very limited. Also, if we only fit PCA on the context window, PCA cannot reconstruct the original context window very well, causing the overall model to behave worse than using the raw time steps directly.

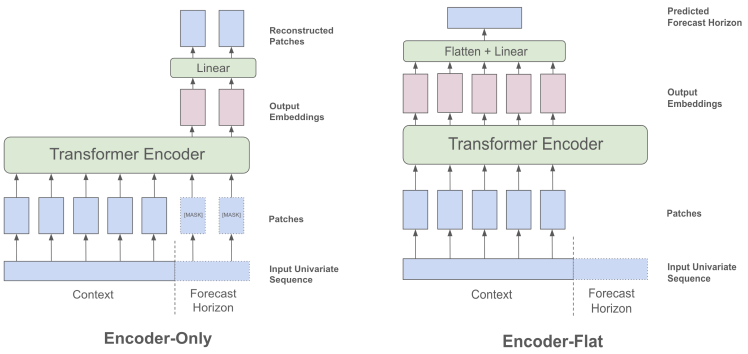

Figure 4: The architectures of Encoder-Only and Encoder-Flat.

Dataset	PCA+Linear	w/o PCA	w/o Pretrain
AirPassengers	32.18	30.78	33.41
AusBeer	16.79	24.11	24.89
GasRateCO2	2.21	2.20	2.21
MonthlyMilk	23.42	24.27	24.72
Sunspots	56.07	61.01	61.17
Wine	3016.00	2882.18	2788.00
Wooly	375.11	478.85	769.41
HeartRate	5.30	5.30	5.31
Arithmetic Mean	**0.6144**	0.6419	0.6759
Geometric Mean	**0.5065**	0.5481	0.5895

Table 4: Ablation study of PCA+Linear on the Darts datasets.

C.2. Comparing PCA+Linear against DecOnly

Following llmtime (Nate Gruver & Wilson, 2023), we report the MAE on the last test window of the original test split on the four ETT datasets in Table 5. The context window has a length of 512, and we evaluate on two forecast horizons of 96 and 192 steps. We see that PCA+Linear is able to outperform models with orders of more weight parameters.

Dataset	llmtime(ZS)	PatchTST	FEDFormer	AutoFormer	TimesFM(ZS)	PCA+Linear
ETTh1 (horizon=96)	0.42	0.41	0.58	0.55	0.45	**0.38**
ETTh1 (horizon=192)	0.50	0.49	0.64	0.64	0.53	**0.43**
ETTh2 (horizon=96)	0.33	0.28	0.67	0.65	0.35	**0.24**
ETTh2 (horizon=192)	0.70	0.68	0.82	0.82	0.62	**0.37**
ETTm1 (horizon=96)	0.37	0.33	0.41	0.54	**0.19**	0.32
ETTm1 (horizon=192)	0.71	0.31	0.49	0.46	**0.26**	0.47
ETTm2 (horizon=96)	0.29	**0.23**	0.36	0.29	0.24	**0.23**
ETTm2 (horizon=192)	0.31	0.25	0.25	0.30	**0.27**	0.33
Number of Wins	0	1	0	0	3	**5**

Table 5: MAE on the ETT Datasets.

C.3. Comparing PCA+Linear against Moirai

We show the full results on the six ETT, Electricity, and Weather datasets in Table 6. For each setting, we highlight the minimum context length required by our model in order to outperform Moirai, which requires a context length of

5000 timesteps. If Moirai performs the best across these context lengths, we highlight Moirai instead. We see that the simple PCA+Linear can perform comparably with Moirai, which has 91 million parameters. Under quite a few settings, PCA+Linear only needs as few as 1000 historical time steps in order to outperform Moirai.

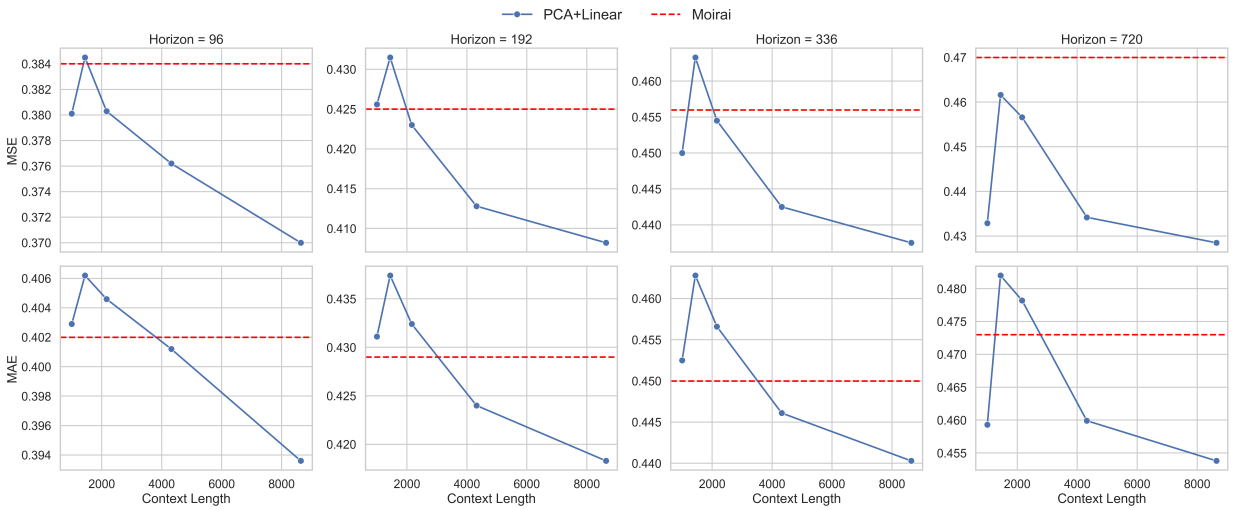

Figure 5: Plot of MSEs/MAEs over increasing context window lengths, with a forecast horizon of 96, 192, 336 and 720 steps on the ETTh1 dataset. The red line indicates the performances of the Moirai base model with a context length of 5000.

Model		Moirai Base		PCA + Linear (Ours)											
Context Window		5000		840		1000		1440		2160		4320		8640	
Dataset	Steps	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTh1	96	0.384	0.402	0.4062	0.4172	**0.3801**	**0.4029**	0.3845	0.4062	0.3803	0.4046	0.3762	0.4012	0.3700	0.3936
ETTh1	192	0.425	0.429	0.4461	0.4405	**0.4256**	**0.4311**	0.4315	0.4374	0.4230	0.4324	0.4128	0.4240	0.4082	0.4183
ETTh1	336	0.456	0.450	0.4633	0.4592	**0.4500**	**0.4525**	0.4633	0.4628	0.4545	0.4566	0.4425	0.4461	0.4375	0.4403
ETTh1	720	0.470	0.473	**0.4441**	**0.4674**	0.4329	0.4593	0.4616	0.4820	0.4566	0.4782	0.4342	0.4599	0.4285	0.4538
ETTh2	96	0.277	**0.327**	0.3223	0.3806	0.3332	0.3961	0.2950	0.3605	0.2871	0.3508	**0.2784**	0.3353	0.2724	0.3354
ETTh2	192	0.340	**0.374**	0.3724	0.4200	0.4238	0.4600	0.3633	0.4130	0.3529	0.4011	**0.3359**	0.3770	0.3296	0.3761
ETTh2	336	0.371	0.401	0.3893	0.4376	0.4509	0.4821	0.3918	0.4388	0.3758	0.4248	**0.3526**	**0.3965**	0.3467	0.3946
ETTh2	720	**0.394**	**0.426**	0.4473	0.4725	0.5020	0.5091	0.4663	0.4844	0.4403	0.4665	0.4153	0.4408	0.3996	0.4335
ETTm1	96	0.335	0.360	**0.3151**	**0.3521**	0.3226	0.3533	0.3294	0.3491	0.3149	0.3426	0.2980	0.3391	0.3058	0.3464
ETTm1	192	0.366	0.379	0.3751	0.3852	0.3721	0.3844	**0.3633**	**0.3753**	0.3565	0.3708	0.3363	0.3634	0.3435	0.3712
ETTm1	336	0.391	0.394	0.4272	0.4154	0.4171	0.4106	0.4038	0.4071	0.4113	0.4088	**0.3747**	**0.3899**	0.3811	0.3968
ETTm1	720	**0.434**	**0.419**	0.5629	0.4823	0.5415	0.4732	0.4777	0.4525	0.4935	0.4592	0.4442	0.4348	0.4458	0.4373
ETTm2	96	0.195	0.269	0.2049	0.2861	0.2004	0.2838	**0.1896**	0.2755	0.1797	**0.2662**	0.1748	0.2615	0.1665	0.2536
ETTm2	192	0.247	0.303	0.2967	0.3409	0.2894	0.3368	0.2761	0.3294	0.2593	0.3187	0.2490	0.3135	**0.2320**	**0.3009**
ETTm2	336	**0.291**	**0.333**	0.3623	0.3761	0.3530	0.3713	0.3503	0.3700	0.3423	0.3638	0.3490	0.3685	0.3276	0.3552
ETTm2	720	**0.355**	**0.377**	0.4351	0.4203	0.4255	0.4157	0.4193	0.4126	0.4107	0.4069	0.4216	0.4116	0.4148	0.4063
Weather	96	**0.167**	**0.203**	0.1972	0.2476	0.1900	0.2397	0.1840	0.2331	0.1770	0.2272	0.1803	0.2302	0.1786	0.2257
Weather	192	**0.209**	**0.241**	0.2471	0.2863	0.2402	0.2797	0.2343	0.2737	0.2286	0.2687	0.2261	0.2669	0.2235	0.2625
Weather	336	**0.256**	**0.276**	0.3027	0.3238	0.2950	0.3174	0.2888	0.3113	0.2820	0.3060	0.2757	0.3022	0.2722	0.2976
Weather	720	**0.321**	**0.323**	0.3632	0.3652	0.3584	0.3615	0.3549	0.3572	0.3493	0.3538	0.3461	0.3510	0.3432	0.3476
Electricity	96	0.158	0.248	**0.1457**	**0.2434**	0.1455	0.2429	0.1446	0.2422	0.1444	0.2416	0.1445	0.2410	0.1429	0.2394
Electricity	192	0.174	0.263	**0.1674**	**0.2631**	0.1669	0.2626	0.1664	0.2625	0.1642	0.2596	0.1627	0.2578	0.1597	0.2547
Electricity	336	0.191	0.278	**0.1926**	0.2864	0.1919	0.2858	0.1907	0.2848	0.1914	0.2850	0.1862	0.2802	0.1833	**0.2774**
Electricity	720	0.229	**0.307**	**0.2302**	0.3170	0.2306	0.3178	0.2306	0.3185	0.2298	0.3175	0.2296	0.3169	0.2278	0.3149

Table 6: Comparison over different context lengths across the common long-term forecasting datasets. For each setting, we highlight the minimum context length required by our model in order to outperform Moirai, which requires 5000 timesteps as context. If Moirai performs the best across these context lengths, we highlight Moirai instead.

Model		Moirai (91M)		Size 1 (13.46M)		Size 2 (13.99M)		Size 3 (36.38M)		Size 4 (40.58M)		Size 5 (73.45M)		Size 6 (92.36M)	
Dataset	Steps	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTh1	96	0.384	0.402	0.3761	0.4102	0.3667	0.4004	0.3819	0.4102	0.3677	0.4013	0.3656	0.4007	0.3758	0.4051
ETTh1	192	0.425	0.429	0.4157	0.4372	0.3996	0.4212	0.4212	0.4351	0.3997	0.4225	0.3925	0.4184	0.4104	0.4265
ETTh1	336	0.456	0.45	0.4501	0.4581	0.4208	0.4331	0.4511	0.4529	0.4211	0.4368	0.4085	0.4284	0.4359	0.4421
ETTh1	720	0.47	0.473	0.4924	0.4932	0.4567	0.4611	0.4997	0.4937	0.4506	0.4655	0.4352	0.4540	0.4814	0.4774
ETTh2	96	0.277	0.327	0.3015	0.3496	0.2891	0.3448	0.2922	0.3524	0.2760	0.3367	0.2701	0.3313	0.2820	0.3422
ETTh2	192	0.34	0.374	0.3765	0.3929	0.3588	0.3889	0.3565	0.3940	0.3361	0.3778	0.3274	0.3707	0.3420	0.3836
ETTh2	336	0.371	0.401	0.4087	0.4142	0.3999	0.4144	0.3907	0.4197	0.3796	0.4090	0.3635	0.3967	0.3797	0.4125
ETTh2	720	0.394	0.426	0.4216	0.4338	0.4177	0.4377	0.4548	0.4664	0.4309	0.4479	0.3952	0.4226	0.4309	0.4487
ETTm1	96	0.335	0.36	0.3221	0.3655	0.3231	0.3639	0.3239	0.3626	0.3132	0.3606	0.3117	0.3602	0.3169	0.3606
ETTm1	192	0.366	0.379	0.3464	0.3830	0.3480	0.3816	0.3495	0.3808	0.3383	0.3776	0.3361	0.3769	0.3419	0.3780
ETTm1	336	0.391	0.394	0.3657	0.3976	0.3675	0.3961	0.3665	0.3938	0.3564	0.3906	0.3540	0.3901	0.3598	0.3920
ETTm1	720	0.434	0.419	0.3977	0.4211	0.3978	0.4170	0.3940	0.4136	0.3876	0.4123	0.3811	0.4083	0.3883	0.4130
ETTm2	96	0.195	0.269	0.1899	0.2824	0.1949	0.2877	0.1966	0.2878	0.1945	0.2871	0.1913	0.2834	0.1904	0.2844
ETTm2	192	0.247	0.303	0.2442	0.3205	0.2486	0.3250	0.2538	0.3270	0.2508	0.3250	0.2441	0.3200	0.2466	0.3241
ETTm2	336	0.291	0.333	0.2927	0.3536	0.2923	0.3555	0.3045	0.3603	0.3008	0.3580	0.2892	0.3500	0.2955	0.3577
ETTm2	720	0.355	0.377	0.3693	0.4031	0.3643	0.4028	0.3840	0.4093	0.3770	0.4063	0.3554	0.3931	0.3747	0.4089
Weather	96	0.167	0.203	0.1747	0.2334	0.1789	0.2356	0.1761	0.2332	0.1695	0.2260	0.1681	0.2234	0.1679	0.2231
Weather	192	0.209	0.241	0.2300	0.2798	0.2376	0.2838	0.2313	0.2793	0.2234	0.2726	0.2220	0.2703	0.2206	0.2694
Weather	336	0.256	0.276	0.2910	0.3209	0.3027	0.3259	0.2915	0.3191	0.2853	0.3147	0.2859	0.3127	0.2814	0.3113
Weather	720	0.321	0.323	0.3816	0.3796	0.4010	0.3877	0.3886	0.3786	0.3782	0.3729	0.3777	0.3686	0.3758	0.3696
Electricity	96	0.158	0.248	0.1378	0.2366	0.1379	0.2355	0.1433	0.2437	0.1418	0.2412	0.1422	0.2416	0.1438	0.2439
Electricity	192	0.174	0.263	0.1552	0.2524	0.1551	0.2513	0.1611	0.2599	0.1590	0.2565	0.1585	0.2558	0.1611	0.2592
Electricity	336	0.191	0.278	0.1735	0.2702	0.1736	0.2692	0.1789	0.2772	0.1767	0.2732	0.1761	0.2727	0.1788	0.2761
Electricity	720	0.229	0.307	0.2157	0.3058	0.2174	0.3059	0.2210	0.3125	0.2199	0.3084	0.2179	0.3079	0.2212	0.3110
Traffic	96	N/A	N/A	0.3786	0.2737	0.3767	0.2700	0.3817	0.2767	0.3841	0.2795	0.3881	0.2810	0.3864	0.2795
Traffic	192	N/A	N/A	0.3940	0.2794	0.3928	0.2759	0.3982	0.2832	0.4000	0.2856	0.4014	0.2845	0.4019	0.2852
Traffic	336	N/A	N/A	0.4086	0.2856	0.4077	0.2823	0.4140	0.2904	0.4142	0.2918	0.4139	0.2890	0.4156	0.2905
Traffic	720	N/A	N/A	0.4464	0.3048	0.4447	0.3016	0.4524	0.3106	0.4504	0.3109	0.4479	0.3067	0.4505	0.3080

Table 7: Performance comparison of various sizes of Encoder-Flat against the Moirai Base model on the OOD datasets. Metrics for Traffic are not provided for Moirai since many similar traffic datasets are used during pretraining. We also highlight the number of parameters in each model in parenthesis. Our implementation is comparable to the performance of Moirai under most settings, sometimes even with a much smaller model size.

Model		Moirai (91M)		Size 1 (1.60M)		Size 2 (2.13M)		Size 3 (12.65M)		Size 4 (16.86M)		Size 5 (37.87M)		Size 6 (56.78M)	
Dataset	Steps	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTh1	96	0.384	0.402	0.3681	0.4029	0.3732	0.4087	0.3810	0.4059	0.3757	0.4035	0.3746	0.3988	0.3731	0.3991
ETTh1	192	0.425	0.429	0.4027	0.4255	0.4081	0.4312	0.4212	0.4300	0.4160	0.4286	0.4198	0.4269	0.4140	0.4229
ETTh1	336	0.456	0.450	0.4260	0.4391	0.4328	0.4459	0.4510	0.4478	0.4478	0.4468	0.4536	0.4463	0.4384	0.4345
ETTh1	720	0.470	0.473	0.4459	0.4611	0.4692	0.4746	0.4937	0.4814	0.4854	0.4749	0.5023	0.4832	0.4692	0.4589
ETTh2	96	0.277	0.327	0.2875	0.3396	0.2784	0.3346	0.2829	0.3370	0.2741	0.3328	0.2886	0.3397	0.2743	0.3357
ETTh2	192	0.340	0.374	0.3504	0.3785	0.3395	0.3736	0.3532	0.3806	0.3406	0.3747	0.3652	0.3882	0.3365	0.3785
ETTh2	336	0.371	0.401	0.3830	0.4022	0.3774	0.4002	0.3997	0.4120	0.3852	0.4050	0.4215	0.4248	0.3734	0.4045
ETTh2	720	0.394	0.426	0.4123	0.4318	0.4083	0.4281	0.4285	0.4398	0.4382	0.4407	0.4805	0.4668	0.4164	0.4378
ETTm1	96	0.335	0.360	0.3381	0.3630	0.3345	0.3626	0.3377	0.3637	0.3323	0.3617	0.3471	0.3653	0.3344	0.3615
ETTm1	192	0.366	0.379	0.3571	0.3774	0.3597	0.3798	0.3613	0.3817	0.3606	0.3814	0.3755	0.3846	0.3606	0.3807
ETTm1	336	0.391	0.394	0.3705	0.3885	0.3788	0.3932	0.3767	0.3945	0.3722	0.3934	0.3904	0.3975	0.3754	0.3934
ETTm1	720	0.434	0.419	0.3997	0.4068	0.4118	0.4141	0.4009	0.4127	0.3909	0.4079	0.4109	0.4141	0.3983	0.4100
ETTm2	96	0.195	0.269	0.1843	0.2757	0.1944	0.2846	0.1936	0.2822	0.1828	0.2719	0.1904	0.2797	0.1861	0.2743
ETTm2	192	0.247	0.303	0.2354	0.3122	0.2503	0.3241	0.2512	0.3223	0.2361	0.3103	0.2509	0.3206	0.2407	0.3134
ETTm2	336	0.291	0.333	0.2817	0.3445	0.2990	0.3575	0.2995	0.3563	0.2813	0.3418	0.3038	0.3561	0.2866	0.3456
ETTm2	720	0.355	0.377	0.3594	0.3949	0.3699	0.4054	0.3747	0.4056	0.3547	0.3903	0.3801	0.4077	0.3584	0.3936
Weather	96	0.167	0.203	0.1758	0.2299	0.1725	0.2252	0.1707	0.2243	0.1698	0.2212	0.1732	0.2251	0.1642	0.2187
Weather	192	0.209	0.241	0.2318	0.2772	0.2266	0.2716	0.2268	0.2730	0.2293	0.2718	0.2341	0.2779	0.2205	0.2680
Weather	336	0.256	0.276	0.2926	0.3180	0.2847	0.3104	0.2948	0.3169	0.2950	0.3160	0.3002	0.3232	0.2846	0.3107
Weather	720	0.321	0.323	0.3643	0.3656	0.3653	0.3621	0.4034	0.3789	0.3882	0.3746	0.3967	0.3816	0.3694	0.3638
Electricity	96	0.158	0.248	0.1426	0.2440	0.1406	0.2414	0.1403	0.2402	0.1404	0.2394	0.1385	0.2368	0.1412	0.2418
Electricity	192	0.174	0.263	0.1590	0.2582	0.1571	0.2562	0.1574	0.2552	0.1570	0.2541	0.1561	0.2528	0.1579	0.2569
Electricity	336	0.191	0.278	0.1757	0.2742	0.1740	0.2726	0.1748	0.2715	0.1736	0.2702	0.1742	0.2703	0.1740	0.2726
Electricity	720	0.229	0.307	0.2157	0.3076	0.2150	0.3069	0.2163	0.3053	0.2127	0.3038	0.2162	0.3058	0.2122	0.3051
Traffic	96	N/A	N/A	0.3808	0.2736	0.3824	0.2741	0.3800	0.2730	0.3771	0.2675	0.3743	0.2678	0.3805	0.2734
Traffic	192	N/A	N/A	0.3962	0.2787	0.3981	0.2797	0.3971	0.2797	0.3930	0.2737	0.3912	0.2745	0.3959	0.2788
Traffic	336	N/A	N/A	0.4088	0.2836	0.4119	0.2858	0.4120	0.2862	0.4081	0.2807	0.4058	0.2807	0.4101	0.2850
Traffic	720	N/A	N/A	0.4442	0.3024	0.4474	0.3042	0.4484	0.3057	0.4438	0.2998	0.4435	0.3005	0.4452	0.3033

Table 8: Performance comparison of various sizes of Encoder-Only against the Moirai Base model on the OOD datasets. Metrics for Traffic are not provided for Moirai since many similar traffic datasets are used during pretraining. We also highlight the number of parameters in each model in parenthesis. Our implementation is comparable to the performance of Moirai under most settings, sometimes even with a much smaller model size.