TUNE-AS-INFERENCE: AMORTIZED CONFIGURATION LEARNING FOR TIME SERIES FOUNDATION MODELS

Piyush Gupta
IIIT Hyderabad
piyush.gupta@research.iiit.ac.in

Doyen Sahoo
Salesforce AI Research
dsahoo@salesforce.com

Manpreet Singh
Salesforce
manpreet.singh@salesforce.com

Vikram Pudi
IIIT Hyderabad
vikram@iiit.ac.in

ABSTRACT

Foundation models for time series forecasting are highly sensitive to configuration parameters such as context length and patch size, which substantially influence predictive performance. These parameters are typically chosen via static defaults or per-series hyperparameter search using AutoML, the latter requiring repeated evaluations of a large model. We reinterpret configuration selection as an amortized learning problem: instead of optimizing configurations independently for each new series, we learn a lightweight learning-to-rank model that predicts high-performing configurations. Using the Moirai 1.1-R-small foundation model on the Uber TLC and Electricity benchmarks, Tune-as-Inference achieves accuracy within 3–5% of 20-trial Bayesian optimization while reducing per-series configuration time by approximately 20 times. These results suggest that configuration adaptation for time series foundation models are better treated as inference rather than iterative search.

Track: Research

1 INTRODUCTION

Foundation models are increasingly adopted for time series forecasting (Woo et al., 2024; Ansari et al., 2024; Garza et al., 2023; Rasul et al., 2023), leveraging large-scale pretraining for cross-domain generalization (Wen et al., 2023). However, their predictive performance remains highly sensitive to configuration parameters such as context length and patch size (Madhusudhanan et al., 2024). These parameters determine the effective receptive field of the model and the granularity at which temporal structure is represented. Even small changes in these settings can lead to substantial differences in forecasting accuracy.

Time series data are inherently heterogeneous. Urban mobility, retail demand, and electricity consumption exhibit distinct seasonalities, trends, and volatility patterns. As a result, a configuration that performs well for one series may perform poorly for another. A single global configuration is therefore rarely optimal across diverse datasets.

A common solution is per-series hyperparameter optimization. Search-based methods such as Bayesian optimization (Snoek et al., 2012; Bergstra et al., 2011) can identify strong configurations, but their computational cost scales linearly with the number of trials. Each trial requires repeated evaluations of a large foundation model, making iterative search expensive in deployment settings with many time series.

We reinterpret configuration selection as an amortized learning problem. Instead of optimizing configurations independently for each new series, we learn a supervised learning-to-rank model that predicts high-performing configurations. This converts hyperparameter tuning into a single inference step.

Our formulation is related to meta-learning approaches that leverage experience across tasks (Hospedales et al., 2021) and feature-based forecast selection methods such as FFORMA (Montero-Manso et al., 2020). Unlike prior work that selects among different forecasting models, we focus on selecting configurations of a single time series foundation model.

Our contributions are:

- We formalize configuration selection for time series foundation models as a supervised learning-to-rank problem over a discrete configuration space.
- We propose a lightweight rank-based meta-learner that predicts high-performing configurations from statistical meta-features.
- We demonstrate that amortized configuration learning achieves performance comparable to search-based tuning while reducing per-series configuration cost by approximately 20 times relative to a 20-trial search budget.

2 Problem Formulation

Let $X = (x_1, \ldots, x_T) \in \mathcal{X}$ denote a time series drawn from a heterogeneous distribution, and let $\mathcal{C}$ be a finite configuration space. Each $c \in \mathcal{C}$ defines a model instance F_c.

Given X, the model produces forecasts $\hat{Y} = F_c(X)$ over horizon H with targets $Y = (y_1, \ldots, y_H)$. We define the loss

$$L(X, c) = \frac{1}{H} \sum_{h=1}^{H} \ell(y_h, \hat{y}_h),$$

where ℓ is a pointwise forecasting loss.

The optimal configuration is

$$c^*(X) = \arg\min_{c \in \mathcal{C}} L(X, c).$$

Standard hyperparameter optimization approximates $c^*(X)$ independently per series via iterative search.

We instead amortize configuration selection across series. Let $\psi(X) \in \mathbb{R}^d$ denote statistical meta-features. Given profiling data $\{(X_i, c, L(X_i, c))\}$ from training series $\{X_i\}_{i=1}^N$, we learn a scoring function $s(\psi(X), c)$ that predicts configuration ranks (lower is better). At inference,

$$\hat{c}(X) = \arg\min_{c \in \mathcal{C}} s(\psi(X), c).$$

This converts per-series hyperparameter search into a single inference step.

3 Method

We convert configuration tuning into a two-phase amortized learning framework consisting of an offline profiling phase and an online inference phase, as illustrated in Figure 1.

3.1 Offline Phase: Profiling and Rank Learning

Given a representative set of time series $\{X_i\}_{i=1}^N$ and a discrete configuration space $\mathcal{C}$, we evaluate the foundation model under all configurations $c \in \mathcal{C}$ to obtain empirical losses

$$\{L(X_i, c)\}.$$

For each series, these losses induce a ranking over $\mathcal{C}$.

We compute statistical meta-features $\psi(X_i) \in \mathbb{R}^d$ for every series that capture *scale, variability, trend strength, and seasonality characteristics*. These features are inexpensive and model-agnostic.

Using the profiling dataset $\{(X_i, c, L(X_i, c))\}$, we train a scoring function $s(\psi(X), c)$ that assigns higher scores to configurations with lower empirical loss. Rather than directly regressing $L(X, c)$,

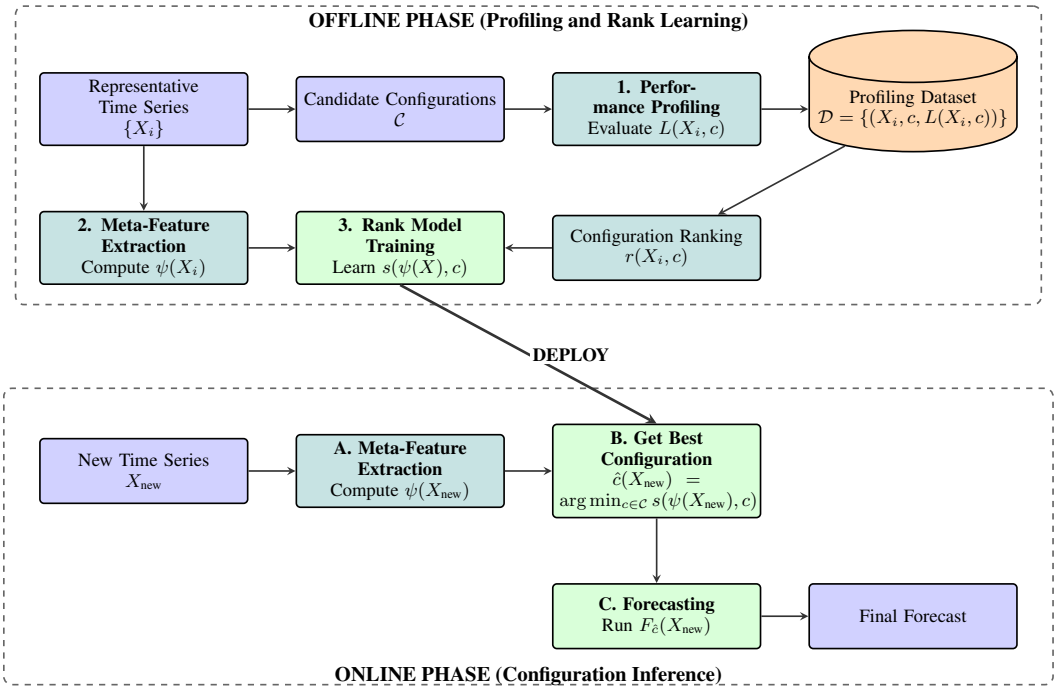


Figure 1: Tune-as-Inference framework. The offline phase profiles configurations, induces empirical rankings $r(X_i, c)$, and trains a scoring model $s(\psi(X), c)$. The online phase directly selects the best configuration for a new series without iterative search.

we formulate training as supervised learning-to-rank over $\mathcal{C}$, allowing the model to learn relative configuration preferences across heterogeneous series. In practice, we implement $s(\cdot)$ using a lightweight gradient-boosted tree regressor, LightGBM (Ke et al., 2017) trained to predict empirical configuration ranks.

The output of the offline phase is a trained meta-learner capable of ranking configurations from meta-features alone.

3.2 Online Phase: Single-Step Configuration Inference

Given a new time series X_{new}, we compute meta-features $\psi(X_{\text{new}})$ and evaluate the learned scoring function over all configurations:

$$\hat{c}(X_{\text{new}}) = \arg\min_{c \in \mathcal{C}} s(\psi(X_{\text{new}}), c).$$

The foundation model is then executed once under $\hat{c}(X_{\text{new}})$. No iterative search is performed; configuration selection reduces to a single forward pass of the meta-learner.

4 Experiments

We evaluate Tune-as-Inference along two axes: predictive performance and computational efficiency. Our objective is to assess whether amortized configuration learning can approach search-based tuning while substantially reducing per-series configuration cost.

4.1 Experimental Setup

We conduct experiments on two real-world forecasting benchmarks: **Uber TLC** (urban mobility demand) and **Electricity (NIPS)** (energy consumption). Both datasets contain heterogeneous seasonal patterns and variability levels.

Table 1: Full test set results. Best Fixed denotes the strongest static baseline.

Dataset	Method	MAE	Time (s)	Relative Cost
Uber TLC	Best Fixed	5.259	5.1	1×
Uber TLC	Tune-as-Inference	4.857	5.1	1×
Uber TLC	Bayesian Opt. (20)	4.680	102.0	20×
Electricity	Best Fixed	131.954	6.2	1×
Electricity	Tune-as-Inference	121.446	6.2	1×
Electricity	Bayesian Opt. (20)	116.370	125.8	20×

For each dataset, we perform a cross-series split: 80% of the time series are used for profiling and meta-learner training, and the remaining 20% unseen series are used for evaluation. Each series is treated as an independent forecasting task.

We use the pretrained **Moirai 1.1-R-small** foundation forecasting model. The configuration space $\mathcal{C}$ consists of context lengths $\{7, 14, 21, 28, 35, 42, 49\}$ (in days) and patch sizes $\{8, 16, 32, 64, 128\}$ (35 total configurations). Context lengths are converted to timesteps according to the native frequency of each dataset. The same configuration space is used for all adaptive methods.

Meta-features are extracted using TSFresh (Christ et al., 2018) and include summary statistics, autocorrelation coefficients, trend strength, and seasonality indicators. Performance is measured using mean absolute error (MAE). Efficiency is measured as wall-clock time required for configuration selection per series.

4.2 Baselines

Best Fixed Configuration. We evaluate static context lengths $\{7, 15, 30, 45, 60\}$ using Moirai's automatic patch-size selection and report the strongest performing fixed configuration on the full test set.

Bayesian Optimization (Optuna). For each series, 20 trials are performed over $\mathcal{C}$. This represents a strong search-based tuning baseline.

4.3 Results

Table 1 reports results on the full test sets. Tune-as-Inference consistently improves over the strongest fixed configuration baseline on both datasets, demonstrating the necessity of adaptive configuration selection.

Compared to 20-trial Bayesian optimization, Tune-as-Inference achieves MAE within 3–5% while requiring only a single model evaluation per series. Search-based tuning scales linearly with the number of trials, whereas our method maintains constant per-series configuration cost.

4.4 Efficiency Perspective

Search-based tuning scales linearly with the number of trials due to repeated model evaluations, incurring significant GPU/CPU cost. Tune-as-Inference profiles configurations offline and reduces deployment-time selection to meta-feature extraction and a single model execution, enabling scalable adaptation across heterogeneous time series.

We further evaluated the framework on PatchTST (Nie et al., 2023), a supervised transformer architecture trained per dataset rather than a pretrained foundation model. Similar performance trends are observed, indicating that amortized configuration learning generalizes across architectures[1].

5 Conclusion

We introduced Tune-as-Inference, a framework that reformulates configuration selection for time series foundation models as amortized learning. By casting configuration selection as a supervised

[1]Full PatchTST results are provided in the Appendix.

learning-to-rank problem over a discrete design space, hyperparameter tuning is reduced to a single inference step.

Experiments with Moirai 1.1-R-small show that amortized configuration learning achieves performance close to 20-trial Bayesian optimization while requiring only one model evaluation per series. These findings suggest that configuration adaptation can be treated as a cross-series generalization problem rather than repeated per-series search.

REFERENCES

Abdul Fatir Ansari, Lorenzo Stella, Can Turkmen, Xiyuan Zhang, Pedro Mercado, Hantian Shen, Oleksandr Shchur, Syama Sundar Rangapuram, et al. Chronos: Learning the language of time series. *Transactions on Machine Learning Research*, 2024.

James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for hyper-parameter optimization. In *Advances in Neural Information Processing Systems*, 2011.

Maximilian Christ, Nils Braun, Julius Neuffer, and Andreas W. Kempa-Liehr. Time series feature extraction on basis of scalable hypothesis tests (tsfresh – a python package). *Neurocomputing*, 307:72–77, 2018.

Alberto Garza, Cristian Challu, and Manuel Mergenthaler-Canseco. Timegpt-1. *arXiv preprint arXiv:2310.03589*, 2023.

Timothy M. Hospedales, Antreas Antoniou, Paul Micaelli, and Amos Storkey. Meta-learning in neural networks: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. Lightgbm: A highly efficient gradient boosting decision tree. In *Advances in Neural Information Processing Systems*, 2017.

Kiran Madhusudhanan, Shayan Jawed, and Lars Schmidt-Thieme. Hyperparameter tuning mlps for probabilistic time series forecasting. *arXiv preprint arXiv:2403.04477*, 2024.

Pablo Montero-Manso, George Athanasopoulos, Rob J. Hyndman, and Thiyanga Talagala. Fforma: Feature-based forecast model averaging. *International Journal of Forecasting*, 2020.

Yuqi Nie, Nam H. Nguyen, Phanwadee Sinthong, and Jayant Kalagnanam. A time series is worth 64 words: Long-term forecasting with transformers. In *International Conference on Learning Representations*, 2023.

Kashif Rasul, Arjun Ashok, Adam R. Williams, Amir Khorasani, et al. Lag-llama: Towards foundation models for probabilistic time series forecasting. *arXiv preprint arXiv:2310.08278*, 2023.

Jasper Snoek, Hugo Larochelle, and Ryan P. Adams. Practical bayesian optimization of machine learning algorithms. In *Advances in Neural Information Processing Systems*, 2012.

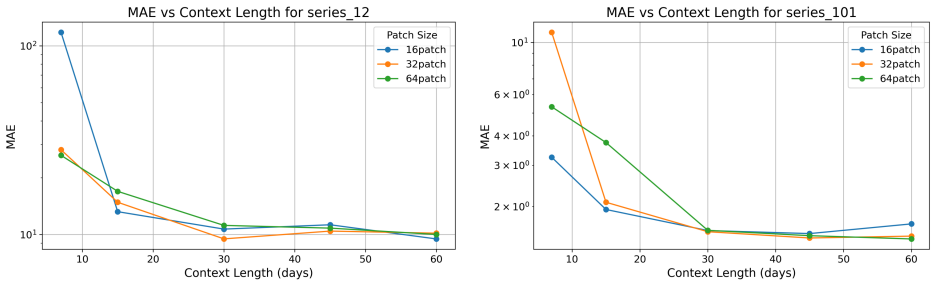
Qingsong Wen, Tian Zhou, Chaoli Zhang, Weiqi Chen, Ziqiang Ma, Jing Yan, and Liang Sun. Transformers in time series: A survey. In *Proceedings of the 32nd International Joint Conference on Artificial Intelligence*, 2023.

Gerald Woo, Chenghao Liu, Akshat Kumar, Caiming Xiong, Silvio Savarese, and Doyen Sahoo. Unified training of universal time series forecasting transformers. In *Proceedings of the 41st International Conference on Machine Learning*, 2024.

A APPENDIX

A.1 CONFIGURATION SENSITIVITY ANALYSIS

Figure 2 illustrates the sensitivity of forecasting performance to context length and patch size on two Uber TLC series using Moirai 1.1-R-small. The optimal configuration differs substantially across series, demonstrating that no single global configuration is consistently optimal. This motivates adaptive configuration selection.



(a) Performance on Uber TLC series_12

(b) Performance on Uber TLC series_101

Figure 2: Sensitivity of forecasting performance to context length and patch size on two Uber TLC series. The optimal configuration varies significantly across series, motivating adaptive configuration selection.

A.2 Meta-Feature Extraction Details

Meta-features are extracted using **TSFresh**. We use the MinimalFCParameters configuration, which includes lightweight statistical descriptors. Representative features include:

- Mean, variance, and standard deviation
- Absolute energy
- Skewness and kurtosis
- Autocorrelation coefficients
- Partial autocorrelation
- Number of peaks
- Linear trend coefficient
- Fourier entropy
- Ratio beyond standard deviation
- Time reversal asymmetry statistic

All features are standardized before training the rank model.

A.3 Generalization Beyond Foundation Models

To evaluate generality beyond pretrained time series foundation models, we apply the same meta-learning framework to **PatchTST**, a supervised transformer architecture trained per dataset.

Table 2 and Table 3 report results on Uber TLC and Electricity. Across patch sizes and evaluation metrics (MAE, MASE, SMAPE), Tune-as-Inference consistently matches or outperforms the strongest fixed context-length configuration.

These results indicate that amortized configuration learning is not tied to a specific foundation model such as Moirai. Instead, it captures cross-series structural regularities that transfer across datasets and across model architectures, including supervised transformer models.

Table 3: PatchTST results on Electricity (NIPS). Lower is better. Best per patch size is **bolded**.

Patch 16	Tune-as-Inference	CL=7	CL=15	CL=30	CL=45	CL=60
MAE	**219.597**	2983.122	340.471	292.636	293.482	307.131
MASE	**1.634**	44.841	2.810	1.983	2.062	2.343
SMAPE	**17.811**	160.490	24.549	19.469	21.863	24.936
Patch 32	Tune-as-Inference	CL=7	CL=15	CL=30	CL=45	CL=60
MAE	**219.597**	1818.174	383.221	292.449	250.840	251.429
MASE	**1.634**	29.896	2.828	1.837	1.742	1.748
SMAPE	**17.811**	147.070	26.850	18.185	19.494	19.427
Patch 128	Tune-as-Inference	CL=7	CL=15	CL=30	CL=45	CL=60
MAE	**219.597**	1933.263	451.069	314.099	272.902	229.888
MASE	**1.634**	16.352	3.399	1.805	1.838	1.641
SMAPE	**17.811**	141.449	29.597	19.722	19.826	17.825

Table 2: PatchTST results on Uber TLC. Lower is better. Best per patch size is **bolded**.

Patch 16	Tune-as-Inference	CL=7	CL=15	CL=30	CL=45	CL=60
MAE	**5.504**	71.231	7.500	5.991	6.110	6.117
MASE	**0.905**	9.768	1.198	0.956	0.973	0.985
SMAPE	**53.107**	152.572	75.694	55.731	54.529	54.884
Patch 32	Tune-as-Inference	CL=7	CL=15	CL=30	CL=45	CL=60
MAE	**5.504**	50.510	8.875	5.838	5.715	5.642
MASE	**0.905**	7.032	1.330	0.946	0.919	0.908
SMAPE	**53.107**	155.521	85.300	55.661	53.457	52.653
Patch 128	Tune-as-Inference	CL=7	CL=15	CL=30	CL=45	CL=60
MAE	**5.504**	20.865	10.604	6.497	5.909	5.739
MASE	**0.905**	3.089	1.567	1.000	0.926	0.907
SMAPE	**53.107**	138.223	97.799	57.241	53.766	53.616

Algorithm 1 Offline Profiling and Rank Learning

Input: Training series $\{X_i\}_{i=1}^{N}$, configuration space $\mathcal{C}$
Output: Trained scoring function $s(\psi(X), c)$
for do
each series X_i
Compute meta-features $\psi(X_i)$
for each configuration $c \in \mathcal{C}$ **do**
Evaluate loss $L(X_i, c)$
end for
Compute empirical ranks $r(X_i, c)$ over $\mathcal{C}$
Add training tuples $([\psi(X_i); c], r(X_i, c))$ to dataset
end for
Train regression model $s(\psi(X), c)$ to predict ranks

return s

Algorithm 2 Online Configuration Inference

Input: New time series X_{new}, trained scorer s, configuration space $\mathcal{C}$
Output: Selected configuration $\hat{c}(X_{\text{new}})$
Compute meta-features $\psi(X_{\text{new}})$
for each configuration $c \in \mathcal{C}$ **do**
    Predict rank $\hat{r}(X_{\text{new}}, c) = s(\psi(X_{\text{new}}), c)$
end for
Select configuration with minimum predicted rank:
    $\hat{c}(X_{\text{new}}) = \arg\min_{c \in \mathcal{C}} \hat{r}(X_{\text{new}}, c)$
Run forecasting model $F_{\hat{c}}(X_{\text{new}})$
return $\hat{c}(X_{\text{new}})$

A.4 Rank Model Implementation

Given profiling data $\{(X_i, c, L(X_i, c))\}$, we compute empirical ranks:

$$r(X_i, c) = \text{rank}_{c \in \mathcal{C}}\big(L(X_i, c)\big),$$

where lower loss corresponds to better rank.

The training dataset consists of tuples:

$$[\psi(X_i); c] \to r(X_i, c).$$

We implement the scoring function $s(\psi(X), c)$ using a **LightGBM regression model** trained with mean squared error to predict configuration ranks directly. At inference time, the configuration with the lowest predicted rank is selected.

A.5 Train/Test Protocol

Time series are treated as independent tasks. For each dataset, we perform an 80/20 split at the series level: 80% of series are used for offline profiling and meta-learner training, and the remaining 20% are held out for evaluation.

No temporal leakage occurs between training and test series. Context lengths are specified in days and converted to the appropriate frequency for each dataset.