SERIESPFN: UNIFYING TIME-SERIES CLASSIFICATION AND EXTRINSIC REGRESSION THROUGH END-TO-END IN-CONTEXT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Time-series classification (TSC) and extrinsic regression (TSER) predict a label or a continuous value from an entire time series, respectively. Unlike conventional approaches that require training a separate model for each dataset, pretrained foundation models such as Prior-Data Fitted Networks (PFNs) enable prediction on unseen tasks through in-context learning. They condition on a task's labeled training examples to predict targets for new series without updating their parameters. Two-stage PFNs achieve strong performance across both tasks, but rely on separate pipelines per task type and incur substantial inference cost. End-to-end time-series PFNs instead predict directly from raw series, but existing methods address classification only and trail the strongest approaches in accuracy. We introduce SeriesPFN, the first end-to-end PFN trained jointly for TSC and TSER. We develop a unified prior that generates classification and regression tasks through a shared construction, with process parameters varying between classes for classification and continuously with the target for regression. With a single checkpoint, SeriesPFN achieves the best mean accuracy rank on the UCR112 and UEA26 classification benchmarks and the second-best mean RMSE rank on the TSER62 regression benchmark. Runtime comparisons place SeriesPFN on the performance–efficiency frontier across all three benchmarks.

1 INTRODUCTION

Time-series classification (TSC) and time-series extrinsic regression (TSER) both map a complete sequence to a single target: a class label in TSC and an external continuous value in TSER. In conventional practice, however, a separate model is fitted, and often tuned, for each time-series dataset (Middlehurst et al., 2024; Xie et al., 2025; Feofanov et al., 2026).

In-context learning (ICL) changes this deployment pattern. Rather than fitting model weights separately for each dataset, a pretrained ICL model conditions on that dataset's labeled training examples at inference and predicts targets for new series without updating its weights. Prior-Data Fitted Networks (PFNs) learn this behavior by pretraining on synthetic tasks sampled from a prior (Müller et al., 2022), an approach established for tabular prediction by TabPFN (Hollmann et al., 2023; 2025).

Recent work has adapted PFNs to time series through both two-stage and end-to-end models. RocketPFN (O'Rourke et al., 2026b), MASHT (Cüppers & Vreeken, 2026), and TS2TabPFN (Merlin & Silva, 2027) extract tabular features from time series and pass them to pretrained TabPFN models. These approaches have achieved strong performance across TSC and TSER, but rely on separate pipelines per task type and can require considerably more time at inference. End-to-end time-series PFNs instead learn temporal representations and perform in-context prediction within a single model (Yeh et al., 2025; Küken et al., 2026; Fang et al., 2026a), but existing approaches address TSC only and trail the strongest methods in accuracy. We argue that TSC and TSER may benefit from shared temporal representations; however, to our knowledge, no existing end-to-end PFN covers both TSC and TSER with competitive predictive performance and low inference cost.

Supporting both TSC and TSER within a single end-to-end PFN requires more than just adding a regression head. A PFN learns from synthetically generated supervised tasks, so its prior must define

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

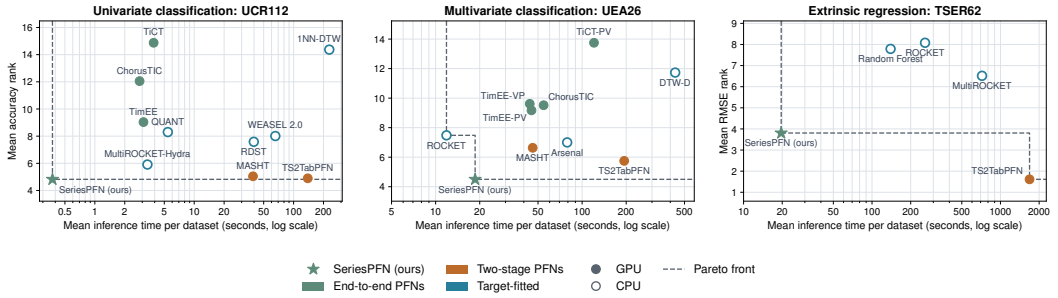


Figure 1: **One checkpoint across classification and regression.** SeriesPFN (ours) handles univariate classification, multivariate classification, and extrinsic regression without per-dataset weight updates, while lying on the measured performance–runtime frontier in each benchmark. The other end-to-end foundation models shown cover classification only. Lower left is better.

both the input series and their relationship to the targets. Existing end-to-end time-series PFNs, however, define this relationship only for categorical targets (Yeh et al., 2025; Küken et al., 2026; Fang et al., 2026a; O'Rourke et al., 2026a). Joint training for TSC and TSER therefore requires a prior that spans both categorical and continuous prediction tasks.

We introduce SeriesPFN, the first end-to-end PFN trained jointly for TSC and TSER. Our unified prior uses the same generative mechanisms for both task types, with process parameters varying between classes for classification and continuously with the target for regression. The architecture interleaves temporal, channel, and sample attention before channel pooling and in-context prediction. SeriesPFN achieves the best mean accuracy rank on the univariate TSC benchmark UCR112 (Dau et al., 2019; Middlehurst et al., 2024) and the multivariate TSC benchmark UEA26 (Bagnall et al., 2018; Ruiz et al., 2021), and the second-best mean RMSE rank on the TSER benchmark TSER62 (Tan et al., 2021; Guijo-Rubio et al., 2024) (Figure 1). Runtime comparisons place SeriesPFN on the performance–efficiency frontier across all three benchmarks. SeriesPFN is also faster than both two-stage PFNs in our runtime comparison: approximately $104\times$ faster than MASHT (Cüppers & Vreeken, 2026) on UCR112, and $10.4\times$ and $85.8\times$ faster than TS2TabPFN (Merlin & Silva, 2027) on UEA26 and TSER62, respectively.

Our contributions are:

- **A multi-task end-to-end PFN for TSC and TSER.** SeriesPFN is the first end-to-end PFN trained jointly for TSC and TSER. It predicts directly from raw time series and labeled context examples, without per-dataset weight updates.
- **A unified time-series prior.** One construction generates both task types: the same generative parameters vary between classes for TSC and continuously with the target for TSER.
- **Strong performance at low inference cost.** With a single checkpoint, SeriesPFN achieves the best mean rank on both TSC benchmarks and the second-best on the TSER benchmark, while lying on the performance–efficiency frontier (Section 5).

2 Background and Related Work

We first formulate TSC and TSER and review conventional task-specific methods (Section 2.1). We then introduce ICL and PFNs and position existing time-series PFNs and their synthetic task priors relative to SeriesPFN (Section 2.2).

2.1 Time-Series Classification and Extrinsic Regression

Task formulation. A time-series classification (TSC) or time-series extrinsic regression (TSER) task consists of a labeled dataset $\mathcal{D} = \{(X_i, y_i)\}_{i=1}^{N}$. The goal is to predict the targets of unseen series. Each $X_i \in \mathbb{R}^{T\times C}$ is a time series with T time steps and C channels, associated with a single target y_i. For TSC, $y_i \in \{1, \ldots, K\}$ is a class label, such as the activity performed during an accelerometer recording; for TSER, $y_i \in \mathbb{R}$ is a continuous value, such as the heart rate during a PPG

recording. Unlike forecasting, which predicts future values of the observed series, TSER predicts a target external to the series (Guijo-Rubio et al., 2024). We consider tasks with rectangular inputs: within each task, all series have the same length and channel count, and corresponding channels represent the same variables in a consistent order. Tasks may differ in dataset size, series length, and channel count; TSC tasks may also differ in the number of classes.

Task-specific methods. Conventional methods fit a separate model to the labeled data of each target dataset. Feature-based approaches first transform each series into a fixed representation and then fit a classifier or regressor to the resulting features. ROCKET (Dempster et al., 2020) and MultiROCKET (Tan et al., 2022) use large collections of random convolutional features, while FreshPRINCE (Middlehurst & Bagnall, 2022) and DrCIF (Middlehurst et al., 2021) derive representations from summary statistics and intervals. HIVE-COTE 2.0 instead combines classifiers built on complementary representations into a meta-ensemble for TSC (Middlehurst et al., 2021). End-to-end architectures such as InceptionTime learn the representation and its mapping to the target jointly from the raw series, but still train a new model for each dataset (Ismail Fawaz et al., 2020). Representation-transfer methods reuse pretrained encoders but likewise fit a new model to their outputs. CauKer (Xie et al., 2025) pretrains such representations on synthetic series, while MantisV2 (Feofanov et al., 2026) pairs a frozen encoder with a task-specific classifier. Thus, all of these approaches require task-specific fitting, while feature-based and representation-transfer methods additionally separate representation construction from target prediction.

2.2 In-Context Learning and Prior-Data Fitted Networks

PFN formulation. In-context learning (ICL) changes how the labeled data of a target dataset are used. In ICL, a pretrained model predicts query targets by conditioning on labeled examples as context, without updating its parameters at inference time. The observed pairs in $\mathcal{D}$ therefore form a context set $\mathcal{D}_{\mathrm{ctx}}$ rather than a dataset on which to fit the model. Given a query input X, the model q_θ produces the predictive distribution $q_\theta(y \mid X, \mathcal{D}_{\mathrm{ctx}})$ over its target y. Prior-Data Fitted Networks (PFNs) learn this behavior by training on synthetic supervised tasks sampled from a task prior p_{task} (Müller et al., 2022). Each training task is divided into a context set and a query example, and the model minimizes the expected query negative log-likelihood:

$$\mathcal{L}(\theta) = \mathbb{E}_{\mathcal{D}_{\mathrm{ctx}} \cup \{(X,y)\} \sim p_{\mathrm{task}}} \left[-\log q_\theta(y \mid X, \mathcal{D}_{\mathrm{ctx}}) \right]. \tag{1}$$

With sufficient capacity and training data, the optimum approximates the posterior predictive distribution induced by the chosen task prior (Müller et al., 2022). Its performance on real datasets therefore depends on how well that prior covers the relationships encountered at inference.

Time-series PFNs. Time-series ICL has developed along two architectural strategies. Two-stage methods first map each series to a fixed-length feature vector and then perform ICL on these vectors with a pretrained tabular PFN such as TabPFN (Hollmann et al., 2025). RocketPFN uses random convolutional features (O'Rourke et al., 2026b), MASHT combines MultiROCKET and Hydra features (Cüppers & Vreeken, 2026), and TIC-FM uses a frozen pretrained time-series encoder (Fang et al., 2026b). These methods address TSC, whereas TS2TabPFN extends the two-stage strategy to both task types, using MultiROCKET with a TabPFN classifier for TSC and tsfresh summary-statistic features (Christ et al., 2018) with a separate TabPFN regressor for TSER (Merlin & Silva, 2027). Two-stage methods achieve strong predictive performance, but cover both task types only through separate pipelines per task type rather than a shared model; our measurements show that the evaluated two-stage methods are substantially slower at inference (Section 5). In contrast, end-to-end methods such as TiCT (Yeh et al., 2025), TimEE (Küken et al., 2026), and ChorusTIC (Fang et al., 2026a) learn the time-series representation and in-context predictor jointly. These models provide direct and efficient prediction from the input series, but remain limited to classification, with accuracy below leading two-stage and supervised methods in our evaluation (Section 5). To our knowledge, no existing end-to-end PFN has yet addressed TSER, leaving open whether a single model can cover both tasks with competitive predictive performance and low inference cost.

Synthetic task priors. Training an end-to-end time-series PFN requires more than a generator of realistic waveforms. Its prior must generate complete supervised tasks, specifying both a distribution over input series and a relationship between the series and their targets. For TSC, this relationship is

categorical: the target identifies one of several conditional distributions over series. For TSER, it is continuous: the distribution over series varies with a real-valued target. Existing priors used to train time-series PFNs are formulated for categorical targets. TiCT constructs classes by mixing synthetic templates (Yeh et al., 2025), TimEE introduces structured shifts into a VARX generator (Küken et al., 2026), and ChorusTIC defines classes through temporal and cross-channel rules (Fang et al., 2026a), while the causal-DAG prior of O'Rourke et al. (2026a) generates multivariate classification tasks by reading a categorical target from the graph. In contrast, the SeriesPFN prior uses target-dependent parameters of one shared generative process for both task types: classification assigns a parameter setting to each class, whereas regression interpolates continuously between two endpoint settings according to the target.

3 A Unified Prior for Time-Series Classification and Regression

A prior over supervised tasks must separate three kinds of variation: what all series in a task share, what changes with the target, and what differs between series with the same target. SeriesPFN draws one structural causal model (SCM) per task and evaluates it separately for each example, with every node producing an entire time series. A Gaussian-process source feeds a random directed acyclic graph of nonlinear transformations, and selected graph nodes are exposed as the observed channels. The SCM structure is shared across context and query examples, while each evaluation includes example-specific stochastic variation. The target changes one set of parameters: either the source parameters or, at selected graph nodes, the parameters used to combine their parents' series. Classification assigns a parameter setting to each class; regression interpolates continuously between two endpoint settings. Figure 2 summarizes this construction.

3.1 Series-Valued SCM

Source. The source node supplies temporal structure shared across examples of a task, together with variation specific to each example. We construct it by combining a shared draw from one Gaussian process with independent per-example draws from a second Gaussian process. The two covariance functions are sampled independently from an adaptation of KernelSynth's compositional grammar (Ansari et al., 2024). For each example i, evaluating the source produces

$$s_i(t) = \sqrt{\rho}\, f(t) + \sqrt{1-\rho}\, r_i(t), \qquad f \sim \mathcal{GP}(0, k_f), \quad r_i \overset{\text{iid}}{\sim} \mathcal{GP}(0, k_r), \tag{2}$$

where f is the shared template, r_i is the per-example draw, and $\rho \sim \mathcal{U}[0,1]$ is drawn once per task to control their variance split. Varying ρ across tasks lets examples governed by the same source parameters range from nearly identical waveforms to independent realizations of the same stochastic process. In 5% of tasks, sparse, randomly signed impulses are added to both f and each r_i (Appendix A.4).

Dynamics Graph. The source generates temporal variation; the dynamics graph determines how that variation appears across related channels. Inspired by CauKer (Xie et al., 2025), we sample an ordered Erdős–Rényi DAG, independently drawing edges from earlier to later nodes, and attach its roots to the shared source. Every node therefore descends from the source. For each example i, node v produces a time series $x_{i,v}$, and $\mathrm{pa}(v)$ denotes the node's parents. At each non-root node v, a deterministic mechanism f_v transforms the parent series $(x_{i,u})_{u\in\mathrm{pa}(v)}$. We add small Gaussian exogenous noise $\xi_{i,v}$, then use $z(\cdot)$ to standardize jointly over all examples and time steps in the task:

$$x_{i,v} = z\left(f_v\big((x_{i,u})_{u\in\mathrm{pa}(v)}\big) + \xi_{i,v}\right). \tag{3}$$

The nodes are evaluated in topological order. Each mechanism composes a combiner c_v, calibration step κ_v, and activation block ϕ_v as $f_v = \phi_v \circ \kappa_v \circ c_v$. The combiner c_v produces a raw output. The calibration $\kappa_v(q) = (q - \mu_v)/s_v$ centers and scales that output using reference inputs. The activation block ϕ_v then applies a node-specific bias and gain together with an activation function. The resulting series is multiplied by a random sign that is sampled once per node and shared across all examples. Without calibration, incidental differences in the combiners' raw scales would determine whether their activations operate near-linearly or in saturation and could destabilize deeper compositions. The bias and gain instead set their operating points explicitly. The final pooled standardization keeps output scales from compounding through graph depth. Temporal structure enters

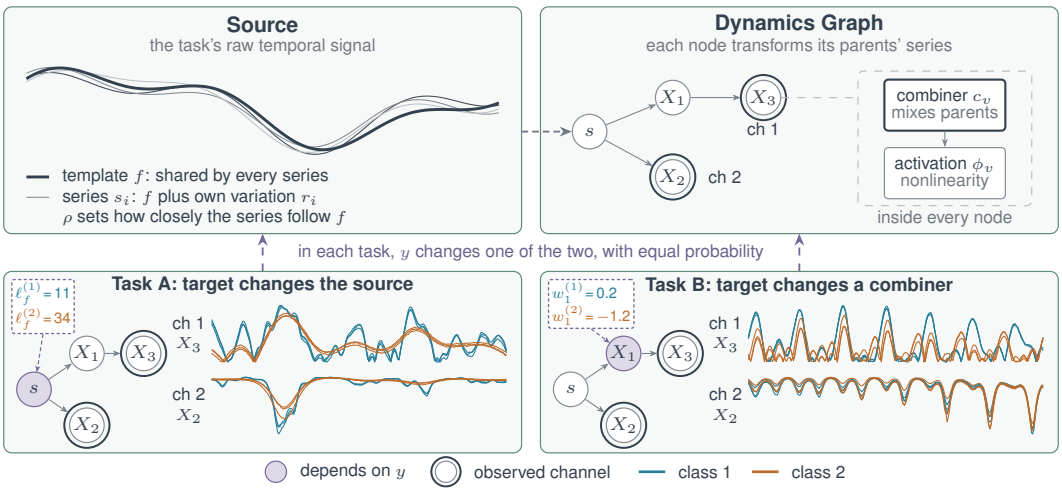


Figure 2: Overview of the SeriesPFN prior. A Gaussian-process source supplies a template f shared by every series plus a per-series variation r_i (Eq. 2). A random graph transforms the source node by node, each node applying a combiner and an activation to its parents' series (Eq. 3); dark-ringed nodes are observed as channels. In each task, y changes either the source parameters (Task A) or the combiner parameters of selected nodes (Task B), with equal probability (Section 3.2). Dashed boxes show, for each class k, one of several parameters that vary: a length scale $\ell_f^{(k)}$ of k_f or a weight $w_1^{(k)}$ of c_1. Curves are three prior draws per class.

through the source and through nodes that apply temporal convolutions; every other node acts pointwise in time. Selecting distinct graph nodes as channels exposes different, potentially nonlinear or temporally filtered views of common source dynamics; the source itself may also be observed.

3.2 Target Dependence

For both task types, target-dependent variation occurs either in the source parameters or in the combiner parameters, chosen with equal probability. All other SCM parameters remain shared across values of y.

Source Parameters. When the source carries target dependence, the target changes the hyperparameters of the two covariance functions k_f and k_r, such as periods and length scales, while their kernel composition and ρ remain fixed (Appendix A.6). The changes to k_f and k_r are sampled independently, so different target values can have distinct shared templates but similar per-example variation, vice versa, or differ in both (Figure 2, left).

Combiner Parameters. When the combiner parameters are target-dependent, the source and topology remain shared. A random nonempty subset of the observed nodes and their non-source ancestors carries the variation (Figure 2, right). These nodes keep their combiner type, activation, gain, and bias; only their combiner parameters vary. Making both groups target-dependent in every task performed worse in our ablations (Appendix E.1).

3.3 Classification

We define synthetic classes by varying parameters of a shared data-generating process. Each task draws its number of classes and class proportions, which range from uneven to nearly balanced (Table 2). For each class k, we form a random vector $\boldsymbol{u}_k$, with one coordinate per target-dependent parameter, by mixing a task-shared standard-normal vector $\boldsymbol{u}$ with an independent class-specific standard-normal vector $\tilde{\boldsymbol{u}}_k$:

$$\boldsymbol{u}_k = \sqrt{\gamma}\,\boldsymbol{u} + \sqrt{1-\gamma}\,\tilde{\boldsymbol{u}}_k. \tag{4}$$

Each coordinate is then mapped to its parameter through that parameter's own sampling distribution; for example, a parameter drawn log-uniformly from $[a, b]$ becomes $a(b/a)^{\Phi(u)}$, where Φ is the

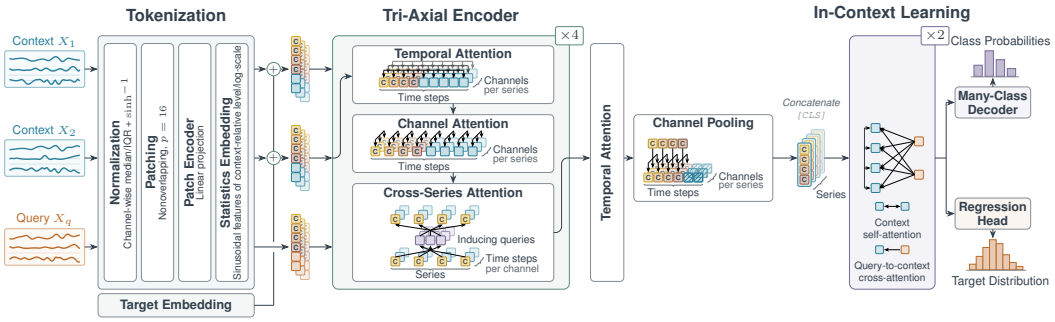


Figure 3: SeriesPFN architecture, read left to right. Each channel is normalized, divided into patches, and embedded; context patch embeddings additionally receive target embeddings, and four `[CLS]` tokens (boxes marked C) are prepended to every channel. Each tri-axial encoder block applies temporal, channel, and inducing-point cross-series attention. A final temporal layer updates the `[CLS]` tokens from the patch representations, after which learned queries pool them across channels and the four pooled tokens are concatenated into one representation per series. The series representations then pass through the ICL stack and into task-specific heads, which produce class probabilities or a continuous target distribution. Token counts are schematic.

standard-normal CDF. This preserves the marginal parameter distribution of each class, while $\gamma \sim \mathcal{U}[0, 1]$ controls similarity between class settings (Appendix A.6). At $\gamma = 1$, all classes share one setting and cannot be distinguished; at $\gamma = 0$, their settings are independent. When the source carries target dependence, k_f and k_r each draw their own γ (Section 3.2); otherwise, all selected combiners share one γ.

3.4 Regression

A regression task replaces the finite set of class parameter vectors with a continuous path between two endpoints. We use linear interpolation as a simple parameterization of this target-dependent variation. Writing these endpoint settings as $\boldsymbol{\psi}^{(0)}$ and $\boldsymbol{\psi}^{(1)}$, series i uses

$$\boldsymbol{\psi}(y_i) = (1 - y_i)\boldsymbol{\psi}^{(0)} + y_i\boldsymbol{\psi}^{(1)}, \qquad y_i \in [0, 1]. \tag{5}$$

The endpoints belong to the selected parameter group and are drawn by the same procedure as a two-class task.

To obtain the targets, we draw a second task from the same prior, without target dependence, containing a single example with one channel whose length equals the number of examples in the regression task. After rescaling this series to $[0, 1]$, its i-th value is the target y_i. The target distribution therefore varies between tasks without an additional mechanism: compared with independently sampled uniform targets, these targets can be skewed, clustered, or leave gaps (Appendix E.1).

4 SeriesPFN Architecture and In-Context Prediction

SeriesPFN embeds patches from each channel as tokens, then updates them through temporal, channel, and cross-series attention. The resulting tokens are pooled into one representation per series and passed to an ICL stack and a task-specific prediction head, as summarized in Figure 3.

4.1 Input Representation

To handle differences in units and numerical scales across datasets, SeriesPFN normalizes each channel within each series so that its patch embeddings primarily represent temporal shape, following Chronos-2 and TimEE (Ansari et al., 2025; Küken et al., 2026). We center each channel by its temporal median and scale it by its interquartile range (IQR), then clip and apply an inverse hyperbolic sine transformation ($\sinh^{-1}$) to limit extreme values. We divide each channel into nonoverlapping patches, reducing sequence length while retaining local temporal structure. A linear

projection maps each patch to a fixed-width token embedding. To retain channel-wise median and IQR information after normalization, we encode these statistics relative to channel-specific references computed from the context series. We add the resulting statistics embedding to every patch token (Appendix B.1). Four learned `[CLS]` tokens are then prepended to each channel's patch-token sequence.

Before the encoder, we add an embedding of each context series' target to every patch token: a learned class-label embedding for classification, or an embedding computed from the scalar target value for regression. Query patch tokens receive no target embedding. Early target conditioning, also used by TimEE for class labels (Küken et al., 2026), makes the target available to SeriesPFN's temporal, channel, and cross-series attention rather than first constructing a target-agnostic representation.

4.2 Tri-Axial Encoder

The three token axes have different structure: temporal positions are ordered, channel identities are assumed aligned within a task, and context series form a set. Flattening them into one sequence would mix these structures and make attention quadratic in their product. Following prior time-series models that factor attention across axes (Ansari et al., 2025; Küken et al., 2026; Chen et al., 2026), each SeriesPFN block applies temporal, channel, and cross-series attention, operating respectively within each channel, across aligned positions of a series, and across series at the same channel-position slot. Cross-series attention uses an inducing-point self-attention block (ISAB) (Lee et al., 2019) with a fixed number of inducing queries, preserving permutation equivariance over series while making its cost linear in their number. The inducing summaries use context series only, so each prediction is independent of the other queries in the batch.

Within the encoder, the four `[CLS]` tokens in each channel gather information and may provide a register-like workspace for nonlocal information (Darcet et al., 2024). To form a series representation, we pool across channels separately at each `[CLS]` position: the same learned query attends to that position in every channel and produces one 128-dimensional summary. Concatenating the four summaries gives the 512-dimensional series representation passed to the ICL blocks. Before this pooling, a final temporal layer lets the `[CLS]` tokens absorb the channel- and cross-series-informed patch states that would otherwise be discarded.

4.3 In-Context Learning and Task Heads

After pooling, the ICL blocks model interactions between whole-series representations, complementing the encoder's interactions at aligned channel and patch positions. Following TabPFN (Hollmann et al., 2025), the mask lets context series interact, while each query attends only to the context and itself. This ensures that its prediction does not depend on the other queries in the batch.

For classification, we use the multi-head retrieval decoder of TabPFN-3 (Grinsztajn et al., 2026), which treats prediction as soft nearest-neighbor retrieval over the labeled context set. Using context labels as attention values makes the decoder equivariant to class-index permutations and non-parametric in the number of classes.

For regression, an MLP predicts a full-support Riemann distribution over the context-standardized target, as in PFN regression (Müller et al., 2022; 2023): probability masses assigned to target buckets define uniform densities within the interior buckets and half-normal tails at the two edges. The resulting full-support density can be asymmetric or multimodal. SeriesPFN uses the same tokenizer, tri-axial encoder, and ICL stack for classification and regression; only the target embeddings and output heads differ.

5 Results

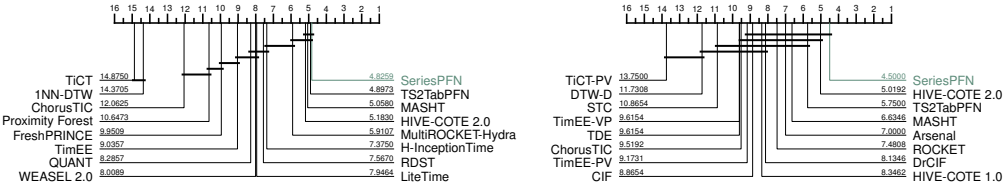
We evaluate whether one jointly trained SeriesPFN is competitive across univariate classification, multivariate classification, and extrinsic regression, and how its predictive performance trades off against inference cost.

5.1 Experimental Setup

We evaluate on UCR112, the 112 equal-length datasets from the 128-dataset univariate UCR archive (Dau et al., 2019; Middlehurst et al., 2024); UEA26, the 26 equal-length datasets from the 30-dataset multivariate UEA archive (Bagnall et al., 2018; Ruiz et al., 2021); and TSER62, the expanded TSER archive's common 62-dataset comparison (Tan et al., 2021; Guijo-Rubio et al., 2024). Empirical checks during development used a 36-dataset meta-validation partition fixed in advance from public metadata; the remaining 164 meta-test datasets were not used to select the method, checkpoint, or evaluation protocol (Appendix D.1). We follow each benchmark's 30-resample protocol and report mean per-dataset accuracy ranks for classification and RMSE ranks for regression. At inference, SeriesPFN averages eight transformed views. Training used 500,000 steps (17 million synthetic tasks; 85.3 GPU-hours on one NVIDIA RTX PRO 6000). Appendices C, D, and F provide evaluation details, complete metric comparisons, and per-dataset results, respectively.

5.2 Benchmark Results

Figure 4 shows that SeriesPFN achieves the lowest mean accuracy rank on both classification benchmarks, with 4.83 on UCR112 and 4.50 on UEA26. The next-best ranks are 4.90 for TS2TabPFN on UCR112 and 5.02 for HIVE-COTE 2.0 on UEA26. SeriesPFN's mean accuracies are 89.30% and 75.72%, compared with 88.64% and 74.48% for those respective alternatives. Appendix Figure 7 compares per-dataset accuracy with TS2TabPFN on UCR112 and HIVE-COTE 2.0 on UEA26.



(a) UCR112 (b) UEA26

Figure 4: Mean accuracy ranks on UCR112 (left) and UEA26 (right). Solid bars connect groups whose pairwise two-sided Wilcoxon signed-rank tests on per-dataset accuracy differences do not reject zero after Holm correction over all method pairs in each benchmark at $\alpha = 0.05$; SeriesPFN is highlighted.

Figure 5 shows that the same jointly trained SeriesPFN checkpoint obtains the second-best mean RMSE rank on TSER62, 3.81, behind TS2TabPFN at 1.61 and ahead of DrCIF at 4.55.

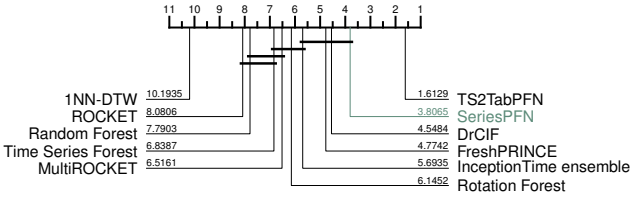


Figure 5: Mean RMSE ranks on TSER62. Solid bars connect groups whose pairwise two-sided Wilcoxon signed-rank tests on per-dataset log RMSE differences do not reject zero after Holm correction over all method pairs at $\alpha = 0.05$.

Figure 1 compares the ranks above with mean inference time. SeriesPFN lies on the Pareto frontier in all three benchmarks. It is the fastest plotted method on UCR112; on UEA26, ROCKET is faster but has a higher mean rank. On TSER62, TS2TabPFN has a lower mean rank, but SeriesPFN reduces mean inference time per dataset from 1684.18 to 19.63 seconds. Appendix D.6 reports the timing protocol and peak-memory results.

5.3 Ablations

Table 1 reports three-seed averages for each 100k-step ablation and matched control. Significance uses two-sided Wilcoxon signed-rank tests across datasets. Appendix E.1 defines each intervention, reports the full statistical results and five additional ablations, and provides expanded interpretations.

Table 1: Ablation minus matched control: accuracy points (UCR, UEA) and percentage change in geometric-mean RMSE (TSER). Bold values are significant after Holm correction within each benchmark at $\alpha = 0.05$.

Ablation	UCR112 ↑	UEA26 ↑	TSER62 ↓
No dynamics graph	$\mathbf{-4.81}$	$\mathbf{-8.21}$	$\mathbf{+19.0\%}$
Source-only	$\mathbf{-1.37}$	$\mathbf{-3.05}$	$\mathbf{+8.1\%}$
Combiner-only	$\mathbf{-0.54}$	-1.09	$+0.0\%$
Both target-dependent	$\mathbf{-0.69}$	$\mathbf{-2.31}$	$+1.6\%$
Classification only	$\mathbf{-0.39}$	-1.88	–
Regression only	–	–	-0.9%

Dynamics graph. Replacing the graph with an independent target-dependent source per channel causes the largest loss on UEA26 and TSER62 and the second-largest on UCR112 (Appendix E.1). The loss on univariate UCR112 suggests that the graph matters beyond multivariate dependence. We interpret it as the prior's main source of series diversity: its nonlinear and filtered views of a target-dependent source act as random augmentations that preserve the target.

Location of target dependence. Restricting target dependence to the source in every task hurts all three benchmarks, whereas restricting it to the combiner parameters hurts only UCR112. Combiner target dependence varies only selected channels, so source-only training never presents tasks in which only some channels are informative; its larger UEA26 loss is consistent with the model learning channel selection from combiner-dependent tasks. Making both parameter groups target-dependent in every task also hurts classification; we hypothesize that the model learns to rely on whichever cue is easier.

Joint training. Classification-only training sees more classification tasks than the control, yet loses 0.39 points on UCR112; UEA26 moves in the same direction without reaching significance. We hypothesize that classification benefits from representations learned on regression tasks. Regression-only training shows no detectable difference on TSER62, so joint training has no detectable cost for regression.

6 Conclusion and Future Work

We introduced SeriesPFN, the first end-to-end PFN trained jointly for time-series classification and extrinsic regression. To enable this joint training, we designed a unified prior that uses the same generative mechanisms for both task types, with process parameters varying between classes for classification and continuously with the target for regression. A single checkpoint achieves the best mean accuracy rank on UCR112 and UEA26 and the second-best mean RMSE rank on TSER62, while lying on the performance–efficiency frontier across all three benchmarks. Together, these results show that end-to-end time-series ICL can unify classification and regression with competitive predictive performance and low inference cost.

The main remaining performance gap is on TSER62, where SeriesPFN ranks behind TS2TabPFN. The prior interpolates linearly between two parameter endpoints, so its regression tasks follow straight paths through parameter space; nonlinear or multi-anchor paths would test whether prior expressivity explains part of this gap. The prior also excludes tasks in which target dependence enters through both the source and the combiners. Making both independently target-dependent hurts classification in our ablation, but other ways of combining the two remain unexplored. Other directions include scaling the model and training on longer series, more channels, and larger context sets.

AI use statement

In this work, we used generative AI tools, specifically Anthropic's Claude Code and OpenAI's Codex, to help refine the conceptual framing of the unified prior, propose and refine research hy-

potheses, provide feedback on experimental methodology, implement the synthetic-task generator and other methods, and assist with interpreting results. The implemented generator sampled the synthetic tasks programmatically; the LLMs did not supply task examples. We did not use these tools to formulate mathematical claims or clean or reformat datasets. Formal proofs, translation, and qualitative or thematic data analysis were not part of this work. Additionally, these tools assisted with literature search and synthesis, checking the support for novelty and related-work claims, source-code investigation, brainstorming, suggesting experimental parameters, creating and editing software, code review and debugging, experiment execution and monitoring, visualization tooling, preparing scientific figures, drafting research notes and manuscript text, editing manuscript text for readability, structuring the paper, and formatting references. We have reviewed all AI-assisted work, including the code, research reasoning, interpretation of results, and manuscript text. We supplemented this review with automated tests, numerical checks, and benchmark-reproduction checks. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

Reproducibility statement

Appendix A specifies the synthetic prior; Appendix B specifies the model and training; Appendix C describes the comparison methods and score provenance; and Appendices D and F give additional results. The accompanying evaluation package contains a checkpoint, inference and scoring code, and a runner with prepared examples from UCR, UEA, and TSER.

References

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

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

Michael Arbel, David Salinas, and Frank Hutter. EquiTabPFN: A target-permutation equivariant prior fitted network. In *Advances in Neural Information Processing Systems*, volume 38, 2025. doi: 10.52202/085713-2097. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/5a66c7adffdbde9dd5e78820cbf6935c-Abstract-Conference.html.

Anthony Bagnall, Hoang Anh Dau, Jason Lines, Michael Flynn, James Large, Aaron Bostrom, Paul Southam, and Eamonn Keogh. The UEA multivariate time series classification archive, 2018. *arXiv preprint arXiv:1811.00075*, 2018. URL https://arxiv.org/abs/1811.00075.

Shane Bergsma, Nolan Dey, Gurpreet Gosal, Gavia Gray, Daria Soboleva, and Joel Hestness. Straight to zero: Why linearly decaying the learning rate to zero works best for LLMs. In *International Conference on Learning Representations*, 2025. URL https://arxiv.org/abs/2502.15938.

Tao Chen, Yexu Zhou, Zhi Gong, Hengwei He, Hongda Li, Zhewei Chen, Dongjing Wang, Xin Zhang, Decheng Liu, Chunlei Peng, Zheng Chen, and Wenyue Ding. Trio: Learning time-series forecasting with temporal-spatial-sample attention and structural causal priors. *arXiv preprint arXiv:2606.07291*, 2026. URL https://arxiv.org/abs/2606.07291.

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. doi: 10.1016/j.neucom.2018.03.067.

Joscha Cüppers and Jilles Vreeken. In-context time series classification with random convolutional features. *arXiv preprint arXiv:2607.19234*, 2026. URL https://arxiv.org/abs/2607.19234.

Gabriel da Costa Merlin and Diego Furtado Silva. TS2TabPFN: Time series classification and extrinsic regression through feature extraction and a tabular foundation model. In *Machine Learning and Knowledge Discovery in Databases. Research Track. ECML PKDD 2026*, volume 16947 of *Lecture Notes in Computer Science*, pp. 555–572, 2027. doi: 10.1007/978-3-032-37676-3_32. URL https://arxiv.org/abs/2608.04174.

Timothée Darcet, Maxime Oquab, Julien Mairal, and Piotr Bojanowski. Vision transformers need registers. In *International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=2dnO3LLiJ1.

Hoang Anh Dau, Anthony Bagnall, Kaveh Kamgar, Chin-Chia Michael Yeh, Yan Zhu, Shaghayegh Gharghabi, Chotirat Ann Ratanamahatana, and Eamonn Keogh. The UCR time series archive. *IEEE/CAA Journal of Automatica Sinica*, 6(6):1293–1305, 2019. doi: 10.1109/JAS.2019.1911747.

Aaron Defazio. Why gradients rapidly increase near the end of training. *arXiv preprint arXiv:2506.02285*, 2025. URL https://arxiv.org/abs/2506.02285.

Angus Dempster, François Petitjean, and Geoffrey I. Webb. ROCKET: Exceptionally fast and accurate time series classification using random convolutional kernels. *Data Mining and Knowledge Discovery*, 34(5):1454–1495, 2020. doi: 10.1007/s10618-020-00701-z.

Juntao Fang, Shifeng Xie, Ruichu Cai, Shengji Zheng, Zijian Li, Keli Zhang, Lujia Pan, Themis Palpanas, and Zhifeng Hao. ChorusTIC: Training-free multivariate time series classification via chorus in-context learning. *arXiv preprint arXiv:2608.24033*, 2026a. URL https://arxiv.org/abs/2608.24033.

Juntao Fang, Shifeng Xie, Shengbin Nie, Yuhui Ling, Yuming Liu, Zijian Li, Keli Zhang, Lujia Pan, Themis Palpanas, and Ruichu Cai. Rethinking zero-shot time series classification: From task-specific classifiers to in-context inference. *arXiv preprint arXiv:2602.00620*, 2026b. URL https://arxiv.org/abs/2602.00620.

Vasilii Feofanov, Songkang Wen, Jianfeng Zhang, Lujia Pan, and Ievgen Redko. MantisV2: Closing the zero-shot gap in time series classification with synthetic data and test-time strategies. *arXiv preprint arXiv:2602.17868*, 2026. URL https://arxiv.org/abs/2602.17868.

Léo Grinsztajn, Klemens Flöge, Oscar Key, Felix Birkel, Philipp Jund, Brendan Roof, Mihir Manium, Shi Bin Hoo, Magnus Bühler, Anurag Garg, Dominik Safaric, Jake Robertson, Benjamin Jäger, Simone Alessi, Adrian Hayler, Vladyslav Moroshan, Lennart Purucker, Philipp Singer, Alan Arazi, Julien Siems, Jan Hendrik Metzen, Georg Grab, Nick Erickson, Siyuan Guo, Eliott Kalfon, Simon Bing, David Salinas, Clara Cornu, Lilly Charlotte Wehrhahn, Diana Kriuchkova, Kursat Kaya, Lydia Sidhoum, Marie Salmon, Jerry Chen, Madelon Hulsebos, Yann LeCun, Samuel Müller, Bernhard Schölkopf, Sauraj Gambhir, Noah Hollmann, and Frank Hutter. TabPFN-3: Technical report. *arXiv preprint arXiv:2605.13986*, 2026. URL https://arxiv.org/abs/2605.13986.

David Guijo-Rubio, Matthew Middlehurst, Guilherme Arcencio, Diego Furtado Silva, and Anthony Bagnall. Unsupervised feature based algorithms for time series extrinsic regression. *Data Mining and Knowledge Discovery*, 38(4):2141–2185, 2024. doi: 10.1007/s10618-024-01027-w.

Noah Hollmann, Samuel Müller, Katharina Eggensperger, and Frank Hutter. TabPFN: A transformer that solves small tabular classification problems in a second. In *International Conference on Learning Representations*, 2023.

Noah Hollmann, Samuel Müller, Lennart Purucker, Arjun Krishnakumar, Max Körfer, Shi Bin Hoo, Robin Tibor Schirrmeister, and Frank Hutter. Accurate predictions on small data with a tabular foundation model. *Nature*, 637:319–326, 2025.

Hassan Ismail Fawaz, Benjamin Lucas, Germain Forestier, Charlotte Pelletier, Daniel F. Schmidt, Jonathan Weber, Geoffrey I. Webb, Lhassane Idoumghar, Pierre-Alain Muller, and François Petitjean. InceptionTime: Finding AlexNet for time series classification. *Data Mining and Knowledge Discovery*, 34(6):1936–1962, 2020. doi: 10.1007/s10618-020-00710-y.

Jaris Küken, Shi Bin Hoo, Martin Mráz, Frank Hutter, and Lennart Purucker. TimEE: End-to-end time series classification via in-context learning. *arXiv preprint arXiv:2607.07500*, 2026. URL https://arxiv.org/abs/2607.07500.

Juho Lee, Yoonho Lee, Jungtaek Kim, Adam R. Kosiorek, Seungjin Choi, and Yee Whye Teh. Set Transformer: A framework for attention-based permutation-invariant neural networks. In *International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 3744–3753, 2019. URL https://proceedings.mlr.press/v97/lee19d.html.

Matthew Middlehurst and Anthony Bagnall. The FreshPRINCE: A simple transformation based pipeline time series classifier. In *Pattern Recognition and Artificial Intelligence*, volume 13364 of *Lecture Notes in Computer Science*, pp. 150–161, 2022. doi: 10.1007/978-3-031-09282-4_13.

Matthew Middlehurst, James Large, Michael Flynn, Jason Lines, Aaron Bostrom, and Anthony Bagnall. HIVE-COTE 2.0: A new meta ensemble for time series classification. *Machine Learning*, 110(11–12):3211–3243, 2021. doi: 10.1007/s10994-021-06057-9.

Matthew Middlehurst, Patrick Schäfer, and Anthony Bagnall. Bake off redux: A review and experimental evaluation of recent time series classification algorithms. *Data Mining and Knowledge Discovery*, 38(4):1958–2031, 2024. doi: 10.1007/s10618-024-01022-1.

Samuel Müller, Noah Hollmann, Sebastian Pineda Arango, Josif Grabocka, and Frank Hutter. Transformers can do Bayesian inference. In *International Conference on Learning Representations*, 2022. URL https://arxiv.org/abs/2112.10510.

Samuel Müller, Matthias Feurer, Noah Hollmann, and Frank Hutter. PFNs4BO: In-context learning for Bayesian optimization. In *International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 25444–25470, 2023. URL https://proceedings.mlr.press/v202/muller23a.html.

Franco Martino O'Rourke, Ana Trisovic, and Dimitris Bertsimas. A causal DAG prior for synthetic time-series classification datasets. *arXiv preprint arXiv:2606.21776*, 2026a. URL https://arxiv.org/abs/2606.21776.

Franco Martino O'Rourke, Ana Trisovic, and Dimitris Bertsimas. RocketPFN: Accurate time series classification via in-context learning. *arXiv preprint arXiv:2606.21786*, 2026b. URL https://arxiv.org/abs/2606.21786.

Zihan Qiu, Zekun Wang, Bo Zheng, Zeyu Huang, Kaiyue Wen, Songlin Yang, Rui Men, Le Yu, Fei Huang, Suozhi Huang, Dayiheng Liu, Jingren Zhou, and Junyang Lin. Gated attention for large language models: Non-linearity, sparsity, and attention-sink-free. *arXiv preprint arXiv:2505.06708*, 2025. URL https://arxiv.org/abs/2505.06708.

Jingang Qu, David Holzmüller, Gaël Varoquaux, and Marine Le Morvan. TabICLv2: A better, faster, scalable, and open tabular foundation model. *arXiv preprint arXiv:2602.11139*, 2026. URL https://arxiv.org/abs/2602.11139.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. Technical report, OpenAI, 2019. URL https://cdn.openai.com/better-language-models/language-models.pdf.

Alejandro Pasos Ruiz, Michael Flynn, James Large, Matthew Middlehurst, and Anthony Bagnall. The great multivariate time series classification bake off: A review and experimental evaluation of recent algorithmic advances. *Data Mining and Knowledge Discovery*, 35(2):401–449, 2021. doi: 10.1007/s10618-020-00727-3.

Noam Shazeer. GLU variants improve transformer. *arXiv preprint arXiv:2002.05202*, 2020. URL https://arxiv.org/abs/2002.05202.

Jianlin Su, Yu Lu, Shengfeng Pan, Ahmed Murtadha, Bo Wen, and Yunfeng Liu. RoFormer: Enhanced transformer with rotary position embedding. *arXiv preprint arXiv:2104.09864*, 2021. URL https://arxiv.org/abs/2104.09864.

Chang Wei Tan, Christoph Bergmeir, François Petitjean, and Geoffrey I. Webb. Time series extrinsic regression: Predicting numeric values from time series data. *Data Mining and Knowledge Discovery*, 35(3):1032–1060, 2021. doi: 10.1007/s10618-021-00745-9.

Chang Wei Tan, Angus Dempster, Christoph Bergmeir, and Geoffrey I. Webb. MultiRocket: Multiple pooling operators and transformations for fast and effective time series classification. *Data Mining and Knowledge Discovery*, 36(5):1623–1646, 2022. doi: 10.1007/s10618-022-00844-1.

Philippe Tillet, H. T. Kung, and David Cox. Triton: An intermediate language and compiler for tiled neural network computations. In *Proceedings of the 3rd ACM SIGPLAN International Workshop on Machine Learning and Programming Languages*, pp. 10–19, 2019. doi: 10.1145/3315508.3329973.

Nikhil Vyas, Depen Morwani, Rosie Zhao, Mujin Kwun, Itai Shapira, David Brandfonbrener, Lucas Janson, and Sham Kakade. SOAP: Improving and stabilizing Shampoo using Adam. *arXiv preprint arXiv:2409.11321*, 2024. URL https://arxiv.org/abs/2409.11321.

Shifeng Xie, Vasilii Feofanov, Ambroise Odonnat, Lei Zan, Marius Alonso, Jianfeng Zhang, Themis Palpanas, Lujia Pan, Keli Zhang, and Ievgen Redko. CauKer: Classification time series foundation models can be pretrained on synthetic data. *arXiv preprint arXiv:2508.02879*, 2025. URL https://arxiv.org/abs/2508.02879.

Chin-Chia Michael Yeh, Uday Singh Saini, Junpeng Wang, Xin Dai, Xiran Fan, Jiarui Sun, Yujie Fan, and Yan Zheng. TiCT: A synthetically pre-trained foundation model for time series classification. *arXiv preprint arXiv:2511.19694*, 2025. URL https://arxiv.org/abs/2511.19694.

In-Kwon Yeo and Richard A. Johnson. A new family of power transformations to improve normality or symmetry. *Biometrika*, 87(4):954–959, 2000. doi: 10.1093/biomet/87.4.954.

A PRIOR SPECIFICATION

A.1 PRIOR OVERVIEW

A supervised time-series prior specifies a distribution over tasks rather than over isolated series. Within each task, context and query series must belong to a common generating family, while the target must identify systematic variation within that family. The prior must therefore distinguish variation shared across the task, variation associated with the target, and residual variation between series with the same target.

SeriesPFN makes this distinction through a series-valued structural causal model. Each task draws a source construction, a directed graph of nonlinear mechanisms, and a set of graph nodes that form the observed channels. This system is shared by the context and query sets. Individual series retain their own stochastic variation, while target-dependent parameters occur in one of two locations: either in the source or in the combiners of selected graph nodes. All remaining parameters are shared across target values.

The target determines how these related parameter settings are indexed. Classification draws a finite collection of correlated class settings, whereas regression draws two endpoint settings and lets the target select a point along the path between them. Crossing the target type with the location of target-dependent variation gives the four regimes summarized in Section 3. The overall sampling procedure is given next; the remainder of this appendix then specifies its components in sampling order.

A.2 SAMPLING A PRIOR TASK

Algorithm 1 summarizes one draw from the prior. The generator first draws the task type and the location of target-dependent variation independently. It then constructs a common source–graph system and a target-indexed family of parameter settings within that system. The same rendering procedure applies in all four regimes. Quantities drawn at the task level are shared by the context and query sets; only quantities explicitly indexed by the target or by an individual series vary within the task.

Algorithm 1 Sampling one SeriesPFN prior task.

Ensure: context set $\{(\mathbf{X}_i, y_i)\}_{i=1}^{N}$ and query set $\{(\mathbf{X}_i, y_i)\}_{i=N+1}^{N+M}$
Draw the task type Q, target owner O, and a feasible task shape: (C, T, K, N, M) for classification or (C, T, N, M) for regression, with $N = M$
Draw a source construction, a dynamics graph, and an ordered set of C observed graph nodes
if $O =$ source **then**
  Designate the source parameters as target-dependent
else
  Select a nonempty subset of eligible graph nodes and designate their combiner parameters as target-dependent
end if
if $Q =$ classification **then**
  Draw correlated class settings, class proportions, and one label y_i per series
else
  Draw two correlated endpoint settings and obtain the coordinates $y_i \in [0, 1]$ from an independent single-output SCM
end if
Instantiate each series at its target setting and render the complete source–graph system
Crop the requested window and collect the observed nodes as channels
Form the context and query sets, retaining one context series per class when required, and draw and apply one time-reversal choice to the entire task
return context and query sets

A.3 TASK SHAPE AND COMPOSITION

Each task independently draws its prediction type and the location of target-dependent variation. Both choices are uniform, yielding four equally likely regimes: classification or regression with

target dependence in either the source parameters or the combiner parameters. The remaining draws determine the number of channels, series length, sample count, and, for classification, the number and relative frequency of classes.

Let $N_{\min}$ be the minimum context count: $N_{\min} = K$ for classification and $N_{\min} = 2$ for regression. For the sampled channel count and series length, $N_{\max}$ is the largest context count permitted by the curriculum's context-token cap at that stage (Appendix B.7). The rows involving K, α, and $\boldsymbol{\pi}$ below apply only to classification.

Quantity	Distribution
Task type	$p(\text{classification}) = p(\text{regression}) = 1/2$
Target owner	$p(\text{source}) = p(\text{structure}) = 1/2$
Channels C	$p(C = c) = c^{-3/2} / \sum_{j=1}^{64} j^{-3/2}, \quad c \in \{1, \dots, 64\}$
Length T	$\log \widetilde{T} \sim \mathcal{U}[\log 2, \log 2048], \quad T = \text{round}(\widetilde{T})$
Classes K	$p(K = k) = k^{-3/2} / \sum_{j=2}^{64} j^{-3/2}, \quad k \in \{2, \dots, 64\}$
Class concentration α	$\log \alpha \sim \mathcal{U}[0, \log 100]$
Class proportions $\boldsymbol{\pi}$	$\boldsymbol{\pi} \mid \alpha, K \sim \text{Dirichlet}(\alpha \mathbf{1}_K)$
Context/query counts N, M	$B_N \sim \text{Beta}(2, 1), \quad N = M = \text{round}\big[N_{\min}(N_{\max}/N_{\min})^{B_N}\big]$

Table 2: Task-size and class-composition distributions.

Channels, length, and class count are proposed independently, but proposals that cannot fit at least one context series per class under the token cap are rejected jointly. Thus, the accepted shape distribution is the distribution in Table 2 conditioned on feasibility. For classification, one series is first reserved for each class and the remaining series are apportioned according to $\boldsymbol{\pi}$ by largest fractional remainder. The context/query split is repaired if needed to retain one context series from every class; a class represented by one series consequently has no query example.

A.4 Source Construction

The source must provide both structure shared across the task and stochastic variation between series governed by the same setting. Equation 2 assigns these roles to a task-frozen realization f and independent series-fresh realizations r_i, with ρ controlling their variance split. At any fixed target setting, each role uses a covariance law shared by all series. The frozen role applies that law to one Gaussian innovation shared across the task, whereas the fresh role draws an independent Gaussian innovation for every series. Their covariance expressions are drawn independently so that persistent and series-specific variation need not occur at the same temporal scales.

Each covariance follows a compositional grammar. The number of atoms is $A = \min\{5, 1 + \text{Poisson}(1)\}$. Each atom independently selects one of five families with equal probability, and each of the $A - 1$ operators is addition or multiplication with equal probability. Parameters described below as relative scales are independent draws of $2^{Z/2}$ for $Z \sim \mathcal{N}(0, 1)$. Conditional on its family, an atom draws:

- **periodic**: period and smoothness independently distributed as relative scales;
- **squared exponential**: one relative length scale;
- **rational quadratic**: shape $\eta_{\text{RQ}} \sim \text{Exp}(1)$ and one relative length scale;
- **Matérn**: smoothness $\nu = \frac{1}{2} + U/(1 - U)$ for $U \sim \mathcal{U}[0, 1)$ and length scale drawn as a relative scale; or
- **spectral mixture**: $Q \sim \text{Unif}\{2, 3, 4, 5\}$ modes, locations $\omega_q \sim \text{Gamma}(2, 1)$, coherences $c_q = 2 + \text{Gamma}(2, 2)$, bandwidths ω_q / c_q, and weights $\boldsymbol{w} \sim \text{Dirichlet}(2\mathbf{1}_Q)$.

The Gamma distributions above are parameterized by shape and scale. The atom families supply periodic, smooth, rough, and multi-frequency covariance structure, while addition and multiplication combine these atoms into composite covariance structures. The atom-level scales are dimensionless and determine the relative temporal scales within each composite. We define the correlation time of the resulting composite as the first lag at which the decaying part of its normalized covariance

reaches $1/e$, and uniformly dilate the composite so that this lag equals the sampled τ_f or τ_r. Correlation time therefore has the same meaning across kernel families and composite expressions. The remaining source-law distributions are listed below.

Quantity	Distribution
Correlation times τ_f, τ_r	$\log \tau_f, \log \tau_r \overset{\text{iid}}{\sim} \mathcal{U}[\log 2, \log T]$
Shared-template variance share ρ	$\rho \sim \mathcal{U}[0, 1]$
Sparse-impulse gate	Bernoulli with probability 0.05
Expected impulse spacing s	$\log s \sim \mathcal{U}[\log 10, \log \max(10, T)]$
Base impulse amplitude a	$\log a \sim \mathcal{U}[\log 0.5, \log 10]$
Positive-sign probability p_+	$p_+ \sim \mathcal{U}[0, 1]$
Amplitude jitter J_t	$J_t \sim \mathcal{U}[-1/4, 1/4]$ octaves

Table 3: Source distributions.

One task-level impulse gate applies to both source roles. This adds occasional non-smooth departures without making impulses a separate source family. When active, each role draws its own s, a, and p_+, and adds

$$e(t) = B_t a 2^{J_t} S_t, \quad B_t \sim \text{Bernoulli}(\min\{1, 1/s\}), \quad S_t \in \{-1, 1\}, \quad p(S_t = 1) = p_+. \tag{6}$$

The event draws are independent across time. Each GP-plus-impulse term is rescaled to unit second moment before the variance split in Equation 2; hence ρ retains its interpretation despite the different covariance and event laws.

A.5 Dynamics Graph

Topology and observation. The dynamics graph turns the common source into related channels by exposing different nodes of one nonlinear system. Channels in a multivariate time series can reflect shared drivers and direct, potentially delayed dependencies; sampling and partially observing a graph allows the prior to express nonlinear and lagged cross-channel relationships. Hidden nodes permit transformations to compose before observation, while selecting observed nodes in random order avoids assigning a privileged semantic role to channel position. For C observed channels, the prior proposes $C + S$ nodes, including the source, where S is geometric with mean 16:

$$p(S = s) = \frac{1}{17}\left(\frac{16}{17}\right)^s, \qquad s \in \{0, 1, \ldots\}. \tag{7}$$

The remaining $n = C + S - 1$ nodes form an ordered Erdős–Rényi graph. For $n > 1$, the prior draws an edge intensity λ from an exponential distribution with mean 2 and retains each possible forward edge independently with probability

$$p_{\text{edge}} = 1 - \exp\left(-\frac{2\lambda}{n-1}\right). \tag{8}$$

For $n \leq 1$, the non-source graph has no edges. Every natural root of the non-source graph is connected to the source. Finally, C distinct nodes are selected uniformly without replacement, in random order, as the observed channels. When combiner parameters are target-dependent, proposals with no observed non-source node are rejected and resampled.

Node mechanisms. Each non-root mechanism separates the operation that combines its parents from the nonlinearity applied afterward. This permits the same temporal or instantaneous relation to appear under different operating regimes, rather than coupling every combiner family to one activation. At each non-root node, the prior samples:

- a **combiner type**, uniformly from {temporal, instantaneous};
- for the temporal type, a **temporal convolution** with length sampled uniformly from $\{3, 5, 7, 9, 11\}$ and unit-normalized Gaussian taps;

- for the instantaneous type, a **combiner** uniformly from {linear map, tree, model tree, random Fourier feature map, multilayer perceptron, bilinear product, sigmoid-gated linear unit};
- family-specific dimensions: tree depth uniformly from 3 to 8, RFF width uniformly from 16 to 64, MLP width uniformly from 4 to 32, and GLU gate sharpness log-uniformly from 1 to 8;
- an independent **activation**, uniformly from {identity, sine, tanh, absolute value, square, ReLU, quantization, exponential}; and
- input bias $b_v \sim \mathcal{N}(0,1)$, gain $\log_2 g_v \sim \mathcal{N}(0,1)$, and an independent output sign sampled uniformly from $\{-1,1\}$.

The RFF length scale for each parent is log-uniform from 1 to 32; the MLP hidden activation is uniform over tanh, GELU, and ReLU. A fair task-level coin chooses centered or causal convolution alignment. For filter length ℓ, the dilation is 1 when $T \le \ell$ and otherwise is $\lfloor 2^U \rfloor$ for $U \sim \mathcal{U}[0, \log_2((T-1)/(\ell-1))]$.

Temporal support. To prevent padding-induced edge artifacts in the observed series, we evaluate the source and dynamics graph on an expanded time canvas. For a convolution with filter length ℓ and dilation d, let $e = (\ell-1)d$. Relative to the largest context required by any parent, a causal convolution adds $(e, 0)$ time steps of left and right context, while a centered convolution adds $(\lfloor e/2 \rfloor, e - \lfloor e/2 \rfloor)$. We propagate these requirements through the graph, take the largest left and right contexts among the observed nodes, and extend the source by those amounts. After evaluating every node on the full canvas, we retain only the requested T-step interior window, so padding at the outer canvas boundary cannot enter an observed channel.

Scale control and process noise. Nonlinear combiners are calibrated on 256 standard-normal reference inputs; linear maps use their weight norm and convolution filters are already unit-normalized. Calibration reduces the influence of a combiner's incidental raw scale on whether its activation operates near linearly or in saturation; the sampled gain and bias primarily control that regime. Calibration spreads are floored at 10^{-6}. Quantization and exponential activations standardize their preactivation once more, and exponential inputs are capped at 5.

The source and each evaluated non-root node independently draw a process-noise fraction q log-uniformly from $[10^{-4}, 10^{-2}]$. Gaussian noise is scaled by $\widehat{\sigma}\sqrt{q/(1-q)}$, where $\widehat{\sigma}$ is the pooled MAD scale of the signal at that site. Each non-root output is finally standardized over all task series and the full rendered time support, including convolution context. Pooling across the task controls numerical scale without separately normalizing away differences between series, and standardizing the full support prevents a locally flat observed crop from setting an arbitrarily small normalization scale.

A.6 Target Dependence

After the common source–graph system has been drawn, exactly one parameter block is designated as target-bearing. The owner is either the source or a nonempty set of graph combiners; it is never both within one task. Parameters outside the designated block remain shared across target values. Consequently, the target indexes systematic variation inside a common generating family rather than selecting an unrelated generator for every series.

Source-parameter variation. The target-dependent parameter settings of the shared-template and per-series source roles are drawn independently. Target-dependent quantities are the periods, length scales, and shape or smoothness parameters of the kernel atoms; the locations, coherences, and weights of spectral mixtures; the correlation times; and, when impulses are active, their spacing, amplitude, and sign probability. The kernel grammar, spectral component counts, variance share ρ, impulse gate, and amplitude-jitter distribution remain fixed.

Combiner-parameter variation. The candidate nodes are the non-source nodes in the union of the observed channels' ancestries, including the observed nodes themselves. Given n_{cand} candidates, the prior draws $n_{\mathrm{var}} \sim \mathrm{Unif}\{1, \ldots, n_{\mathrm{cand}}\}$ and selects that many nodes uniformly. Each

selected node varies only its combiner parameters: linear and convolution weights; tree leaf values; model-tree leaf weights and, for fitted model trees, leaf biases; RFF amplitudes; MLP weights and hidden biases; or the two weight vectors of a bilinear or GLU combiner. The source, graph topology, combiner family and dimensions, tree partitions, RFF frequencies and phases, activation, gain, input bias, and output sign remain fixed. This changes relations within a shared dynamical system without also changing the source that drives it.

Classification. For classification, each varying group draws $\gamma \sim \mathcal{U}[0, 1]$ and uses Equation 4. Standard-normal coordinates are transformed through the inverse sampling laws of their parameters. Each class therefore retains the corresponding base marginal distribution, while γ controls similarity between class settings. Tasks with source-parameter target dependence draw separate values of γ for the frozen and fresh roles, and the kernel and event parameters of each role use its own value. All class templates apply their class-specific covariance to the same Gaussian innovation, so their differences reflect source parameters rather than independent task-frozen realizations. A task with combiner-parameter target dependence uses one γ for the coefficients of all selected nodes.

Regression. For regression, the targets $y_{1:N+M}$ are obtained from a length-$(N + M)$ trajectory of a separate, independent single-output SCM and min–max scaled to $[0, 1]$; constant or nonfinite trajectories are redrawn. This varies the empirical target distribution across tasks without deriving it from the observed source–graph system. The two endpoint settings are sampled by the same correlated law as two classes, and series i uses y_i to interpolate every varied parameter according to Equation 5. The discrete construction of the selected component remains fixed along the path.

A.7 Rendering and Task Assembly

Once the target-indexed settings have been constructed, all four task regimes use the same rendering procedure. For each series, the generator selects its class setting or interpolates its endpoint settings at y_i, realizes and combines the frozen and fresh source roles, adds active source events and process noise, and evaluates the required graph nodes in topological order. Node mechanisms apply their calibrated combiner and activation before process noise and pooled standardization. The expanded temporal support is then cropped to the requested window, and the sampled observed nodes are collected in their sampled order as channels.

The rendered series are assigned to equal-sized context and query sets using the composition rules above. Classification retains at least one context example from every class. Finally, one time-reversal decision is applied to every series in the task, preserving channel alignment and the context–query relationship, and the two sets are returned.

A.8 Prior Samples

Figure 6 shows one sampled task for each combination of task type and target owner.

A.9 Scope and Limitations

The SeriesPFN prior is a coverage-oriented distribution over supervised time-series tasks, not an estimate of the empirical prevalence of particular data-generating mechanisms. Its equal weighting of task types and target owners, and its fixed sampling weights over source and graph mechanisms, are design choices. Consequently, prior samples illustrate behaviors available to the model during training rather than a fitted distribution over real datasets.

Every task contains one source and one shared dynamics graph, and its observed channels are nodes within that system. This construction can represent common drivers, hidden nonlinear transformations, and instantaneous or lagged dependencies between channels, but not multiple independent source systems or arbitrary groups of unrelated channels within one task. The sampled graph is a generative device for producing multivariate dependence; it is not intended as an identifiable causal explanation of a real dataset. Tasks also use a common set of observed channels and a regular time grid across all series within the task.

Target dependence is deliberately localized. Within one task, either the source parameters or selected graph-combiner parameters vary with the target, but not both. Classification uses a finite collection

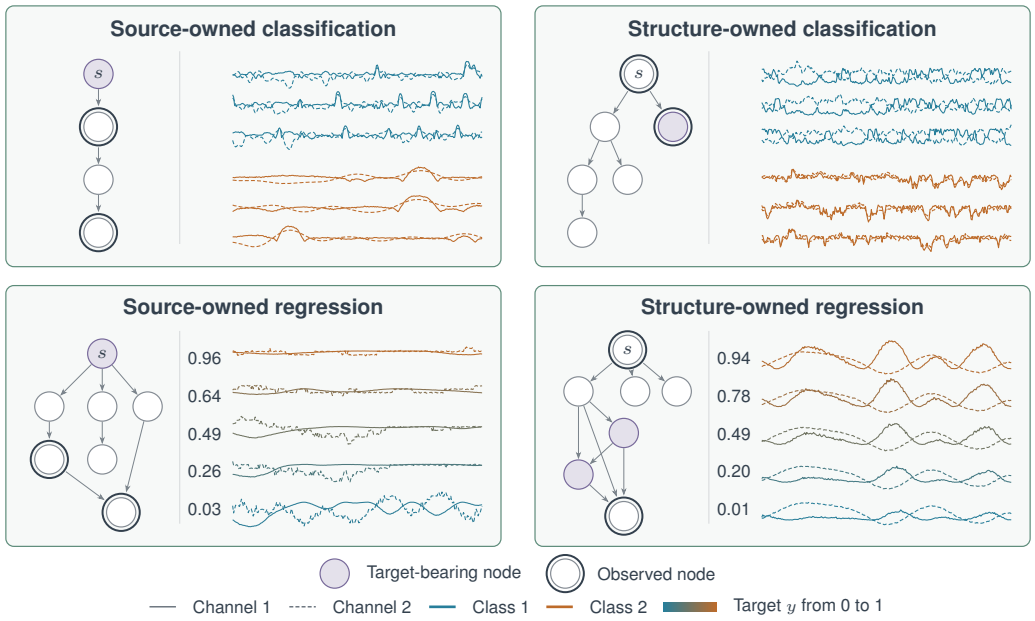


Figure 6: Prior samples for the four combinations of task type and target owner, each with two channels and 256 time steps. Purple nodes carry the target, and dark rings mark the observed channels.

of correlated parameter settings, while regression follows a one-dimensional path between two endpoint settings. The prior therefore excludes tasks with mixed source and structural target dependence and restricts continuous target dependence to a single path through parameter space.

These restrictions isolate where target-dependent variation enters each task and preserve a common generative structure across its examples. Natural extensions include multiple sources, mixed target ownership, richer continuous parameter paths, and irregular or partially observed time grids.

B Model Details

B.1 Tokenization and Target Conditioning

Consider a task with N context and M query series, indexed so that $i \le N$ denotes context. Each series has T time steps and C aligned channels. A patch contains p time steps, so each channel produces $P = \lceil T/p \rceil$ patch tokens. Each channel begins with R learned `[CLS]` register tokens, and each cross-series block uses I inducing queries.

For series i and channel c, let m_{ic} be the temporal median and let s_{ic} be the interquartile range, falling back to the population standard deviation when the interquartile range is zero. Values are normalized as

$$\widetilde{x}_{ict} = \sinh^{-1}\left(\mathrm{clip}\left(\frac{x_{ict} - m_{ic}}{s^{+}_{ic}}, -100, 100\right)\right), \qquad s^{+}_{ic} = \begin{cases} s_{ic}, & s_{ic} > 0, \\ 1, & s_{ic} = 0. \end{cases} \tag{9}$$

Positions beyond T in the final patch are set to zero after normalization. For each task and channel, the reference level $\bar{m}_c$ is the median of m_{ic} over context series, and the reference scale $\bar{s}_c$ is the median of their positive s_{ic}. The level and scale features are

$$\ell_{ic} = \frac{\sinh^{-1}((m_{ic} - \bar{m}_c)/\bar{s}_c)}{\ln 2}, \qquad r_{ic} = \log_2(s_{ic}/\bar{s}_c). \tag{10}$$

If every context scale is zero, a zero level difference maps to zero and a nonzero difference to ± 100; positive and zero series scales map to 100 and zero, respectively. When $\bar{s}_c > 0$ but $s_{ic} = 0$, we set $r_{ic} = -100$.

Each scalar $z \in \{\ell_{ic}, r_{ic}\}$ is clipped to $[-100, 100]$ and encoded as

$$\phi(z) = \left[\sin\left(\frac{2\pi z}{\lambda_j}\right), \cos\left(\frac{2\pi z}{\lambda_j}\right)\right]_{j=1}^{12}, \tag{11}$$

where the wavelengths λ_j are geometrically spaced from 0.5 to 1024. For normalized patch j, the resulting token is

$$\mathbf{z}_{icj} = W_{\text{patch}}\widetilde{\mathbf{x}}_{icj} + W_{\text{stat}}[\phi(\ell_{ic}); \phi(r_{ic})] + \mathbf{t}_i \in \mathbb{R}^{d_{\text{enc}}}. \tag{12}$$

The patch and statistics projections are shared across channels. For a context series, $\mathbf{t}_i$ is its target-conditioning vector; for a query series, $\mathbf{t}_i = \mathbf{0}$. Classification uses a learned embedding of the assigned class slot. For regression, the model standardizes the optionally transformed targets using the mean and population standard deviation of the finite context targets. Applying ϕ followed by a bias-free linear projection gives $\mathbf{t}_i$. The same vector is added to every patch token of the series. The R learned `[CLS]` tokens precede the patch tokens in each channel and receive neither the statistics embedding nor the target-conditioning vector. The same R learned vectors are used for every channel and series.

B.2 Tri-Axial Encoder

Each attention block applies RMS pre-normalization, per-head QK normalization, a head-specific sigmoid gate on the attention output, a residual connection, and a SwiGLU feed-forward sublayer (Qiu et al., 2025; Shazeer, 2020). Temporal attention uses RoPE (Su et al., 2021). The temporal, channel, and cross-series operations use this block design with separate parameters, and parameters are not shared across encoder blocks.

Each of the L_{enc} encoder blocks applies three operations in order. Temporal attention acts on the $R + P$ tokens of one channel. Channel attention then acts across channels at a fixed token position within one series. Cross-series attention finally acts across series at a fixed channel-position slot.

The cross-series operation is an inducing-point self-attention block (ISAB) (Lee et al., 2019). For channel c and token position s, let $\mathbf{Z}_{cs}$ contain the states of all $N + M$ series and let $\mathbf{Z}_{cs}^{\text{ctx}}$ select its context rows. Each encoder block has its own I learned inducing queries $\mathbf{U}$ and applies

$$\mathbf{H}_{cs} = \text{MAB}(\mathbf{U}, \mathbf{Z}_{cs}^{\text{ctx}}), \qquad \mathbf{Z}'_{cs} = \text{MAB}(\mathbf{Z}_{cs}, \mathbf{H}_{cs}), \tag{13}$$

where $\text{MAB}(Q, K/V)$ denotes the pre-normalized attention and feed-forward block above, with a residual on Q. The first operation uses the query-aware scalable softmax (QASSMax) of Qu et al. (2026); it rescales each inducing query using the number of context keys and a learned query-dependent factor. The second operation uses ordinary scaled softmax. Because only context rows construct $\mathbf{H}_{cs}$, query series cannot affect one another through the encoder.

B.3 Series Compression and Whole-Series ICL

After the final tri-axial block, one additional temporal block lets each `[CLS]` token read the final patch states of its channel after the last channel and cross-series operations. We then pool across channels. At each of the R `[CLS]` positions, a learned query attends to the corresponding `[CLS]` token from every channel; patch tokens do not enter this step. Concatenating the pooled vectors gives

$$\mathbf{r}_i = [\mathbf{g}_{i1}; \ldots; \mathbf{g}_{iR}] \in \mathbb{R}^{Rd_{\text{enc}}} = \mathbb{R}^{d_{\text{icl}}}. \tag{14}$$

The L_{icl} whole-series ICL blocks use QASSMax and the attention mask

$$A_{ij} = 1 \quad \Longleftrightarrow \quad j \le N \text{ or } (i = j \text{ and } i > N). \tag{15}$$

Thus context series attend to context series, while each query attends to all context series and itself. Context series do not attend to queries, and queries do not attend to one another. An RMS normalization produces the final whole-series representation passed to the prediction heads.

B.4 Prediction Heads

For classification, the multi-head retrieval decoder projects query and context representations into H_{clf} pairs of query and key spaces, following attention-weighted target decoding (Arbel et al., 2025; Grinsztajn et al., 2026). Unlike TabPFN-3, SeriesPFN applies per-head QK normalization to prevent uncontrolled query/key norm growth, which we observed during development. The value of each context series is the one-hot vector of its class slot, and the retrieval attention has no output gate. If $a_{qi}^{(h)}$ is the attention weight from query q to context series i in head h, then

$$p_q(c) = \frac{1}{H_{\text{clf}}} \sum_{h=1}^{H_{\text{clf}}} \sum_{i=1}^{N} a_{qi}^{(h)} \mathbf{1}[y_i = c]. \tag{16}$$

The same retrieval projections serve any $K \leq K_{\max}$, where $K_{\max}$ is the checkpoint's class-slot capacity. Each task assigns its labels to distinct slots, with learned embeddings for target conditioning and one-hot values for retrieval. Following TabPFN-3, the loss uses $\log(\max(p_q(c), 10^{-5}) + 3 \times 10^{-5})$ for numerical stability.

For regression, a two-layer SiLU MLP maps the d_{icl}-dimensional query representation through a d_{reg}-dimensional hidden layer to B logits. Their softmax probabilities π_{qb} parameterize a density over the context-standardized target z_q,

$$p(z_q \mid \mathbf{r}_q) = \sum_{b=1}^{B} \pi_{qb} f_b(z_q). \tag{17}$$

The bucket borders are equal-mass quantiles of context-standardized query targets from $4{,}096$ independently drawn regression tasks, with equal total weight per task. The interior f_b are uniform within their buckets. The two edge components are half-normal tails whose scales equal the adjacent bucket width divided by the median of a unit half-normal. The distribution therefore has support on the full real line, as in PFN regression (Müller et al., 2022; 2023). For each ensemble member, we invert the target transform and context standardization before computing its distribution mean, then average these means in the original target units.

The tokenizer, tri-axial encoder, compression layers, ICL blocks, and output normalization are shared by classification and regression. Classification has its own class-slot embeddings and retrieval head; regression has its own target projection and density head.

B.5 Computational Structure and Cached Inference

Let $S = R + P$ be the number of tokens per channel and d_h the classification retrieval head dimension. Table 4 gives the attention mixing and prediction-head costs.

Stage	Interaction	Complexity
Temporal encoder	Tokens within a channel	$O(L_{\text{enc}}(N+M)CS^2 d_{\text{enc}})$
Channel encoder	Channels at one token position	$O(L_{\text{enc}}(N+M)SC^2 d_{\text{enc}})$
Cross-series ISAB	Series at one channel-position slot	$O(L_{\text{enc}}CSI(2N+M)d_{\text{enc}})$
Final temporal block	Tokens within a channel	$O((N+M)CS^2 d_{\text{enc}})$
Channel pooling	Channels at each `[CLS]` position	$O((N+M)RCd_{\text{enc}})$
Whole-series ICL	Context–context and query–context	$O(L_{\text{icl}}(N^2+MN)d_{\text{icl}})$
Classification head	Query–context retrieval	$O(H_{\text{clf}}MNd_h)$
Regression head	Dense query readout	$O(Md_{\text{reg}}(d_{\text{icl}}+B))$

Table 4: Attention mixing and prediction-head complexity by model stage. Attention rows omit projection and feed-forward costs.

The ISAB is linear in the number of series for fixed I. When queries require multiple chunks, a context pass caches the context-derived inducing summaries for every encoder block and the context keys and values for every ICL block. It also retains the final context representations and, for classification, their retrieval keys and one-hot values. Query chunks reuse these states and follow the same mask in Equation 15. The context-side ISAB reduction can itself be streamed with online

softmax, avoiding materialization of its full attention matrix without changing the inducing summaries. Evaluation limits query and forward work to budgets of $1{,}048{,}576$ and $2{,}097{,}152$ tokens, respectively, without capping the number of query rows.

B.6 Model Configuration and Initialization

Tokenizer and attention projections, the regression-target projection, and learned inducing and pooling queries use a zero-mean truncated-normal initialization with standard deviation 0.02 and truncation at two standard deviations. Residual-writing attention and feed-forward projections instead use standard deviation $0.02/\sqrt{40}$, following the depth-scaled residual initialization of Radford et al. (2019) to account for residual updates accumulating across blocks. The `[CLS]` vectors are initialized from $\mathcal{N}(0, d_{\text{enc}}^{-1})$, while class-slot embeddings are initialized orthogonally. Other affine layers retain their PyTorch defaults. The model uses no dropout; weight decay is specified below.

Hyperparameter	Value
Tokenization and embeddings	
Patch size p	16
Patch stride	16
Encoder width d_{enc}	128
`[CLS]` tokens per channel R	4
Scalar-encoding wavelengths	12
Scalar-encoding wavelength range	$[0.5, 1024]$
Tri-axial encoder	
Tri-axial blocks L_{enc}	4
Inducing queries I	64
Query / key-value heads	4 / 4
Head dimension	32
SwiGLU hidden width	384
Compression and in-context learning	
Final temporal blocks	1
Channel-pooling attention blocks	1
Channel-pooling queries	4
Width d_{icl}	512
Blocks L_{icl}	2
Query / key-value heads	8 / 8
Head dimension	64
SwiGLU hidden width	1,408
Output normalization	RMSNorm
Prediction heads	
Classification retrieval heads H_{clf}	8
Classification retrieval head dimension d_h	64
Maximum class slots $K_{\max}$	64
Regression head type	Full-support Riemann distribution
Regression hidden width d_{reg}	768
Regression buckets B	1,024
Regression hidden activation	SiLU
Shared attention settings	
SDPA output gate (G_1)	Head-specific elementwise sigmoid
QASSMax MLP hidden width	64
QASSMax MLP activation	GELU
RMSNorm ϵ	10^{-6}
Temporal RoPE base	10^4
Model size	
Parameters (total / trainable)	12,977,400 / 12,977,400

Table 5: Model hyperparameters.

B.7 Training Procedure and Hyperparameters

Training tasks are drawn on demand from the synthetic prior, with classification and regression tasks entering the same stream of packed batches. Each task has equal numbers of context and query series. We reverse all series in a task as one augmentation, randomly map classification labels to learned class slots, and use either the identity or a context-fitted Yeo–Johnson transform for regression targets (Yeo & Johnson, 2000).

A context-size curriculum raises the per-task context-token cap linearly from $8{,}192$ to $32{,}768$ over the first half of training, then holds it at $32{,}768$. For a sampled series length and channel count, this cap determines the largest permitted context set; the length and channel sampling ranges do not change during training.

We optimize the model jointly for both tasks. Classification uses query-label negative log-likelihood, while regression uses cross-entropy over buckets of context-standardized targets. We average query losses within each task and then across tasks. We optimize with SOAP (Vyas et al., 2024) and a linear decay-to-zero schedule (Bergsma et al., 2025). Weight decay is corrected for the learning-rate schedule on normalized paths (Defazio, 2025); unnormalized output weights retain ordinary decay, and one-dimensional gains and biases are not decayed. Parameters are stored in FP32 and model computation uses BF16 autocasting. Table 6 gives the numerical settings.

Hyperparameter	Value
Task sampling	
Classification / regression probability	0.5 / 0.5
Context / query series ratio	1 / 1
Classes per classification task	2–64
Maximum series length	$2{,}048$
Maximum channels	64
Regression target transforms	Identity, Yeo–Johnson
Transform probabilities	0.5 / 0.5
Regression-border calibration draws	$4{,}096$
Curriculum	
Context token cap, start / end	$8{,}192$ / $32{,}768$
Context-token schedule	Linear
Curriculum fraction of training	0.5
Optimization	
Token capacity per optimizer step	$524{,}288$
Gradient accumulation steps	1
Optimizer	SOAP
β_1 / β_2	0.9 / 0.999
Optimizer ϵ	10^{-8}
Preconditioner refresh interval	10 steps
Base weight decay	0.1
Peak learning rate	3×10^{-4}
Scheduled training steps	$500{,}000$
Warmup steps	$5{,}000$
Stable steps	0
Decay steps	$495{,}000$
Learning-rate decay	Linear to zero
Parameter / compute precision	FP32 / BF16
Training seed	0

Table 6: Training hyperparameters.

B.8 Efficient Training Execution

Drawing synthetic tasks online places prior generation on the training critical path. We implement the prior in native C++ and run it across 15 persistent data-loader workers. As tasks become available, Python groups differently sized tasks into flattened batches subject to limits on total tokens and

cross-series attention work, avoiding padding every task to the largest shape. Prefetching overlaps this CPU pipeline with GPU optimization.

The model forward pass uses `torch.compile`. Attention layouts are constructed once per packed batch and reused across encoder blocks. Our custom kernels, written in Triton (Tillet et al., 2019), handle short and single-query ragged-attention operations without materializing dense block-diagonal masks. The run processed about 17 million synthetic tasks over $500{,}000$ optimizer steps, taking 85.3 GPU-hours on one NVIDIA RTX PRO 6000 Blackwell Server Edition with 16 allocated CPU cores and 96 GiB of host memory.

C Comparison Methods and Score Sources

C.1 Compared Prediction Systems

The comparisons span methods with different ways of using the labeled target dataset. Conventional archive methods fit a classifier or regressor to each dataset. Two-stage PFNs extract fixed time-series features and pass them to a tabular PFN. End-to-end time-series PFNs learn the representation and in-context predictor together. SeriesPFN uses one jointly trained encoder and ICL stack for classification and regression, with task-specific target conditioning and prediction heads; its eight-prediction inference policy is specified in Appendix D.2.

Table 7: **Positioning of SeriesPFN among time-series prediction methods.** No target fitting means that no model parameters are estimated from the target dataset; end-to-end means jointly learning the series representation and target-prediction mechanism; and multi-task means that one jointly trained model natively supports both TSC and TSER, rather than using separate task-specific approaches under one name.

Method	Classification	Regression	Multi-task	No target fitting	End-to-end
SeriesPFN (ours)	✓	✓	✓	✓	✓
TS2TabPFN	✓	✓	–	✓	–
HIVE-COTE 2.0	✓	–	–	–	–
ROCKET / MultiROCKET / DrCIF / FreshPRINCE	✓	✓	–	–	–
H-InceptionTime / InceptionTime ensemble	✓	✓	–	–	✓
TIC-FM / RocketPFN / MASHT	✓	–	–	✓	–
TiCT / TimEE / ChorusTIC	✓	–	–	✓	✓

Two-stage PFNs. TS2TabPFN uses MultiROCKET features and a TabPFN 2.5 classifier for TSC, but pooled tsfresh 0.21.1 `EfficientFCParameters` features and a separate TabPFN 2.5 regressor for TSER (Merlin & Silva, 2027). We locally rerun the released classifier on all UCR112 and UEA26 datasets and the released regressor on all TSER62 datasets, retaining predictions across all 30 resamples for each benchmark. MASHT combines MultiROCKET and Hydra features with a TabPFN-3 classifier (Cüppers & Vreeken, 2026). Its UCR112 and UEA26 results come from our runs of the released MASHT feature extractor and TabPFN-3 classifier checkpoint.

End-to-end classification PFNs. We evaluate the released TimEE code and checkpoint on UCR112 (Küken et al., 2026). Its ensemble averages raw and first-differenced inputs interpolated to 256 and 512 time steps. On UEA26, TimEE-PV applies the model to each channel and averages probabilities, while TimEE-VP uses the released multivariate-fine-tuned checkpoint and learned channel pooling. We also run ChorusTIC's released multivariate model with its full-context prediction configuration (Fang et al., 2026a). For UCR, our implementation caches ChorusTIC's signal-level representations within each prediction call and reuses them across class-shift ensemble members; the released evaluator recomputes those identical single-channel representations for each member. The TimEE execution used for our measurements tiles attention while preserving its prediction rule. On UCR112, our run of TiCT's released classifier and checkpoint reaches 74.21%

mean accuracy over 30 resamples, compared with 79.60% for the TiCT row tabulated by ChorusTIC (Yeh et al., 2025; Fang et al., 2026a). TiCT's released evaluator pools the archive train and test files before making an $80/10/10$ train/validation/test split, using the first two partitions for retrieval; our resamples retain the archive's train/test sizes. On UEA26, TiCT-PV applies the released checkpoint independently to each channel and averages probabilities.

C.2 Score Sources

The conventional classification rows are imported from the published UCR and UEA result archives (Middlehurst et al., 2024; 2021); the conventional TSER rows come from the expanded regression archive (Guijo-Rubio et al., 2024). They are published scores, not locally rerun implementations. Table 8 distinguishes these imports from local predictions. Full method slates appear in Tables 17–19.

Benchmark	Published score sources	Local predictions
UCR112	Archive classifiers	SeriesPFN, TimEE, TiCT, ChorusTIC, MASHT, TS2TabPFN
UEA26	Archive classifiers	SeriesPFN, TimEE-PV, TimEE-VP, TiCT-PV, ChorusTIC, MASHT, TS2TabPFN
TSER62	Archive regressors	SeriesPFN, TS2TabPFN

Table 8: Origin of the active comparison scores. Published entries are imported scores; local entries are predictions generated for this study.

D Additional Evaluation Results

D.1 Development and Held-Out Partitions

Before model development, we selected the meta-validation datasets in Table 9 using only public metadata, assigning datasets that share subjects or an acquisition process to the same partition. During development, no decision about the prior, architecture, training recipe, checkpoint, inference protocol, metrics, or comparison slate used meta-test data. Beyond the public metadata used to define the split, we did not inspect meta-test series, empirical summaries or visualizations, or SeriesPFN predictions or scores. We first examined the meta-test results after freezing the checkpoint and evaluation plan for the paper, and did not revise the primary method or evaluation in response.

The split registry covers the full archives, whereas the reported results use the benchmark subsets defined above. InsectWingbeat is assigned to meta-validation but lies outside UEA26; AustraliaRainfall is likewise assigned to meta-validation but lies outside TSER62.

Benchmark	Meta-validation datasets
UCR18	SmoothSubspace, Chinatown, ItalyPowerDemand, ECG200, GunPoint, ECG5000, Coffee, Wafer, ShapesAll, FordA, ElectricDevices, DiatomSizeReduction, HandOutlines, Rock, PigAirwayPressure, PigArtPressure, PigCVP, Phoneme
UEA6	AtrialFibrillation, Heartbeat, EigenWorms, DuckDuckGeese, PhonemeSpectra, PenDigits
TSER12	NaturalGasPricesSentiment, AppliancesEnergy, ParkingBirmingham, BenzeneConcentration, IEEEPPG, BIDMC32HR, BIDMC32RR, BIDMC32SpO2, GasSensorArrayAcetone, GasSensorArrayEthanol, NewsHeadlineSentiment, NewsTitleSentiment

Table 9: Meta-validation datasets in the evaluated benchmarks. Every other dataset in each benchmark belongs to meta-test. The same assignments are provided machine-readably in the accompanying `meta_splits.yaml`.

	Meta-validation			Meta-test		
Benchmark	Datasets	Primary rank ↓	NLL rank ↓	Datasets	Primary rank ↓	NLL rank ↓
UCR	18	4.19	2.89	94	4.95	3.39
UEA	6	1.17	3.33	20	5.50	3.20
TSER	12	4.00	N/A	50	3.76	N/A

Table 10: SeriesPFN mean ranks on the meta-validation and meta-test partitions. Ranks are recomputed within each partition. Primary rank denotes accuracy rank for UCR and UEA and RMSE rank for TSER. NLL ranks compare the twelve UCR and fifteen UEA methods with soft probability outputs; NLL does not apply to regression.

Table 10 shows how SeriesPFN's relative position changes with dataset composition by recomputing ranks within each partition against the method slate for the complete benchmark. The shifts are not consistent across benchmarks or metrics. The large UEA accuracy difference may reflect SeriesPFN's strength on the particular structural extremes represented in the six-dataset development partition. Because the pattern is absent in UEA NLL and does not extend consistently to other extreme UEA datasets, we interpret it as partition-specific rather than a general development-set advantage.

D.2 Evaluation Protocol

At evaluation, SeriesPFN uses the full training split as context and the test split as queries, without updating its weights. Series longer than $2{,}048$ time steps are antialiased and resampled to that limit; inputs with more than 64 channels use PCA fitted and standardized on context inputs.

Each prediction averages eight ensemble members spanning all combinations of three choices: using the capped series length or resampling to 512 time steps, using the original values or first differences, and keeping or inverting the sign. The same input transformation is applied to the context and query series. For classification, each member uses a different random assignment between the observed class labels and the model's learned class-label embeddings; we undo this assignment before averaging probabilities. For regression, four members leave the targets unchanged and four apply a context-fitted Yeo–Johnson transform (Yeo & Johnson, 2000). We invert the target transform and context standardization before averaging predicted means in the original target units.

D.3 Inference Ensemble Sensitivity

The protocol above averages eight predictions from the trained model. These predictions can differ through context selection, input preprocessing, and task-specific target representations. We separate these sources to ask what the reported inference policy contributes beyond the trained model itself.

The first comparison concerns context length. In the sampled-context configuration, each member receives a subset of the labeled training examples that fits within the context-token budget used during training. Full-context inference instead supplies the entire training split, including when it exceeds that training budget. Comparing these configurations therefore asks both whether context sampling omits useful labeled examples and whether SeriesPFN can use longer contexts than those encountered during training.

The members of the original-series ensemble are not identical evaluations. For classification, each member assigns the observed classes to different learned label slots. The retrieval head is equivariant to permutations of class indices, and we restore the dataset's class order before averaging probabilities. The preceding ICL body, however, receives the learned label embeddings and is not equivariant to exchanging those embeddings. Different class-to-slot assignments can therefore produce different predictions even when the series and context rows are unchanged. For regression, members alternate between the identity target representation and a context-fitted Yeo–Johnson transformation (Yeo & Johnson, 2000). We transform every predicted distribution back to the original target units before averaging.

The remaining comparisons ask whether temporal preprocessing adds useful diversity to those existing ensemble mechanisms. The four-member 256/512-step variant applies TimEE's specific recipe to SeriesPFN: raw and first-differenced series at both resolutions (Küken et al., 2026). This experiment tests whether a preprocessing strategy developed for TimEE also works for our model. The selected eight-member ensemble follows the evaluation protocol above and was chosen based on meta-validation performance.

Across results computed on the partitions and on the complete benchmarks (Tables 11 and 12), moving from sampled to full context helps most clearly on UEA and TSER, showing that access to additional labeled examples matters independently of preprocessing. With full context fixed, the TimEE preprocessing improves UCR but does not transfer uniformly to UEA and TSER. The selected ensemble retains the UCR improvement while leaving the held-out UEA and TSER primary metrics close to original-series, full-context inference. Its top-label ECE is higher on both held-out classification partitions, and its TSER interval-coverage error is larger.

		Meta-validation		Meta-test	
Benchmark	Inference variant	Primary	Calibration	Primary	Calibration
UCR112	Original series, sampled context	89.12	0.037	87.35	0.044
	Original series, full context	89.17	0.034	87.39	0.042
	256/512-step preprocessing	89.87	0.062	88.33	0.064
	Preprocessing ensemble	91.18	0.064	88.94	0.067
	Selected minus original full context	+2.01 pp	+0.029	+1.55 pp	+0.025
UEA26	Original series, sampled context	65.98	0.135	75.73	0.069
	Original series, full context	69.58	0.098	77.43	0.061
	256/512-step preprocessing	69.15	0.101	76.78	0.073
	Preprocessing ensemble	69.74	0.103	77.54	0.076
	Selected minus original full context	+0.16 pp	+0.005	+0.10 pp	+0.015
TSER62	Original series, sampled context	0.551	0.054	0.492	0.039
	Original series, full context	0.500	0.046	0.475	0.033
	256/512-step preprocessing	0.467	0.061	0.505	0.042
	Preprocessing ensemble	0.465	0.066	0.490	0.046
	Selected minus original full context	-0.0358	+0.020	+0.0155	+0.013

Table 11: Inference-policy comparison on the meta-validation and meta-test datasets. Primary scores are accuracy for UCR112 and UEA26 and relative RMSE for TSER62. Calibration is measured by ten-bin top-label ECE for classification and absolute 90% coverage error for regression. The final row gives the selected preprocessing ensemble minus original-series, full-context inference on the same datasets; lower is better except for accuracy.

Benchmark	Inference variant	Primary	Calibration
UCR112	Original series, sampled context	87.63	0.043
	Original series, full context	87.68	0.041
	256/512-step preprocessing	88.58	0.064
	Preprocessing ensemble	89.30	0.067
	Selected minus original full context	+1.63 pp	+0.026
UEA26	Original series, sampled context	73.48	0.084
	Original series, full context	75.62	0.070
	256/512-step preprocessing	75.02	0.079
	Preprocessing ensemble	75.74	0.082
	Selected minus original full context	+0.11 pp	+0.012
TSER62	Original series, sampled context	0.504	0.042
	Original series, full context	0.480	0.036
	256/512-step preprocessing	0.497	0.046
	Preprocessing ensemble	0.485	0.050
	Selected minus original full context	+0.0055	+0.014

Table 12: The inference-policy comparison on the complete UCR112, UEA26, and TSER62 benchmarks. Values combine the two partitions in Table 11, weighted by their dataset counts.

Prediction budget. The preceding comparison changes both the temporal views and the number of predictions being averaged. We therefore compare the selected preprocessing ensemble with the original-series ensemble at matched budgets of one, two, four, and eight predictions. This is not a comparison against repeated identical forwards: the original-series members still vary their learned label assignments for classification and their identity or Yeo–Johnson target representations for regression. The comparison asks whether temporal views add useful diversity beyond those task-specific ensemble mechanisms.

For budgets below eight, we average prespecified subsets of the captured predictions without selecting subsets from their scores. The original-series control uses every subset of a given size, while the preprocessing subsets cover different combinations of transformations. Thus each column compares the complete inference policies at the same prediction count.

Across Tables 13–15, the preprocessing ensemble overtakes the original-series ensemble on UCR from two predictions onward on both partitions, so its advantage is not explained by averaging more predictions. UEA approaches parity only as more views are included. On held-out TSER, preprocessing remains worse in relative RMSE at every budget, although the gap narrows as more views are added. The corresponding calibration errors generally increase as views are added, making the value of temporal preprocessing dependent on the task and prediction budget.

Benchmark	Metric	Ensemble	Predictions 1	2	4	8
UCR18	Accuracy (%)	Original-series ensemble	88.69	88.95	89.10	89.16
		Preprocessing ensemble	88.05	89.98	90.99	91.17
	ECE	Original-series ensemble	0.033	0.033	0.034	0.034
		Preprocessing ensemble	0.034	0.050	0.061	0.063
UEA6	Accuracy (%)	Original-series ensemble	68.88	69.30	69.49	69.58
		Preprocessing ensemble	67.62	68.80	69.52	69.60
	ECE	Original-series ensemble	0.105	0.101	0.099	0.097
		Preprocessing ensemble	0.101	0.099	0.102	0.103
TSER12	Relative RMSE	Original-series ensemble	0.502	0.501	0.501	0.501
		Preprocessing ensemble	0.492	0.477	0.468	0.465
	Coverage error	Original-series ensemble	0.044	0.045	0.046	0.046
		Preprocessing ensemble	0.044	0.058	0.065	0.066

Table 13: Prediction-budget comparison on the UCR18, UEA6, and TSER12 meta-validation datasets. Each cell averages over prespecified member subsets, then over 30 resamples within each dataset, and finally over datasets. Accuracy is reported in percent; lower values are better for all other metrics. The eight-member values come from the separate member-capture evaluation rather than the primary result runs.

Benchmark	Metric	Ensemble	Predictions 1	2	4	8
UCR94	Accuracy (%)	Original-series ensemble	86.81	87.14	87.31	87.39
		Preprocessing ensemble	86.10	87.87	88.72	88.94
	NLL	Original-series ensemble	0.348	0.340	0.336	0.334
		Preprocessing ensemble	0.367	0.337	0.324	0.321
	ECE	Original-series ensemble	0.040	0.041	0.042	0.042
		Preprocessing ensemble	0.042	0.056	0.065	0.067
UEA20	Accuracy (%)	Original-series ensemble	76.45	77.00	77.28	77.42
		Preprocessing ensemble	74.69	76.28	77.14	77.46
	NLL	Original-series ensemble	0.546	0.532	0.526	0.522
		Preprocessing ensemble	0.580	0.542	0.525	0.521
	ECE	Original-series ensemble	0.059	0.059	0.060	0.061
		Preprocessing ensemble	0.061	0.067	0.073	0.076
TSER50	Relative RMSE	Original-series ensemble	0.481	0.478	0.476	0.475
		Preprocessing ensemble	0.517	0.502	0.494	0.490
	Coverage error	Original-series ensemble	0.033	0.033	0.033	0.033
		Preprocessing ensemble	0.030	0.039	0.044	0.046

Table 14: Prediction-budget comparison on the UCR94, UEA20, and TSER50 meta-test datasets. Aggregation and metric conventions follow Table 13.

Benchmark	Metric	Ensemble	Predictions 1	2	4	8
UCR112	Accuracy (%)	Original-series ensemble	87.11	87.43	87.60	87.67
		Preprocessing ensemble	86.41	88.21	89.09	89.30
	NLL	Original-series ensemble	0.358	0.349	0.345	0.343
		Preprocessing ensemble	0.377	0.343	0.328	0.325
	ECE	Original-series ensemble	0.039	0.040	0.041	0.041
		Preprocessing ensemble	0.041	0.055	0.064	0.067
UEA26	Accuracy (%)	Original-series ensemble	74.70	75.23	75.48	75.61
		Preprocessing ensemble	73.06	74.55	75.38	75.65
	NLL	Original-series ensemble	0.610	0.596	0.588	0.585
		Preprocessing ensemble	0.641	0.599	0.581	0.577
	ECE	Original-series ensemble	0.070	0.069	0.069	0.069
		Preprocessing ensemble	0.071	0.075	0.080	0.082
TSER62	Relative RMSE	Original-series ensemble	0.485	0.482	0.481	0.480
		Preprocessing ensemble	0.512	0.497	0.488	0.485
	Coverage error	Original-series ensemble	0.035	0.035	0.035	0.036
		Preprocessing ensemble	0.033	0.042	0.048	0.050

Table 15: Prediction-budget comparison on the UCR112, UEA26, and TSER62 complete benchmarks. Aggregation and metric conventions follow Table 13.

D.4 Complete Accuracy and RMSE Comparisons

Table 16 reports dataset-level wins, ties, losses, and corrected paired tests against the principal alternatives. Tables 17–19 give the complete accuracy and RMSE comparison slates used for the main-text rank diagrams.

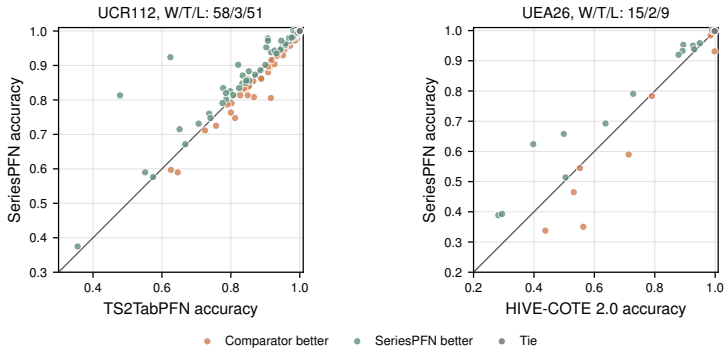


Figure 7: Per-dataset mean accuracy against the closest-ranked alternative on UCR112 (left) and UEA26 (right). Each point is one dataset; points above the diagonal favor SeriesPFN. W/T/L reports SeriesPFN wins, ties, and losses.

Benchmark	Comparator	SeriesPFN W/T/L	Holm p
UCR112	TS2TabPFN	58/3/51	0.7433
	MASHT	60/3/49	0.7433
	HIVE-COTE 2.0	60/3/49	0.5578
UEA26	HIVE-COTE 2.0	15/2/9	0.4400
	TS2TabPFN	18/1/7	0.4400
	MASHT	17/1/8	0.2406
	TiCT-PV	23/0/3	0.0004
TSER62	TS2TabPFN	13/0/49	0.000027

Table 16: Dataset-level wins, ties, and losses and two-sided Wilcoxon signed-rank tests against the principal alternatives. Classification rows compare accuracy; the TSER62 row compares RMSE. Holm correction covers the full SeriesPFN comparison slate for the corresponding benchmark and metric.

Method	Accuracy ↑	Mean rank ↓
SeriesPFN	**89.30%**	**4.83**
TS2TabPFN	88.64%	4.90
MASHT	89.19%	5.06
HIVE-COTE 2.0	89.14%	5.18
MultiROCKET–Hydra	88.40%	5.91
H-InceptionTime	87.61%	7.38
RDST	87.63%	7.57
LiteTime	86.93%	7.95
WEASEL 2.0	87.38%	8.01
QUANT	86.71%	8.29
TimEE	85.95%	9.04
FreshPRINCE	85.54%	9.95
Proximity Forest	83.70%	10.65
ChorusTIC	82.98%	12.06
1NN-DTW	75.55%	14.37
TiCT	74.21%	14.88

Table 17: UCR112 accuracy and mean-rank results.

Method	Accuracy ↑	Mean rank ↓
SeriesPFN	**75.72%**	**4.50**
HIVE-COTE 2.0	74.48%	5.02
TS2TabPFN	74.74%	5.75
MASHT	73.34%	6.63
Arsenal	72.21%	7.00
ROCKET	71.96%	7.48
DrCIF	72.43%	8.13
HIVE-COTE 1.0	72.23%	8.35
CIF	72.77%	8.87
TimEE-PV	70.95%	9.17
ChorusTIC	69.71%	9.52
TDE	70.21%	9.62
TimEE-VP	70.41%	9.62
STC	70.32%	10.87
DTW-D	66.26%	11.73
TiCT-PV	63.53%	13.75

Table 18: UEA26 accuracy and mean-rank results.

Method	RMSE rank ↓
TS2TabPFN	**1.61**
SeriesPFN	3.81
DrCIF	4.55
FreshPRINCE	4.77
InceptionTime ensemble	5.69
Rotation Forest	6.15
MultiROCKET	6.52
Time Series Forest	6.84
Random Forest	7.79
ROCKET	8.08
1NN-DTW	10.19

Table 19: TSER62 RMSE ranks for eleven methods.

D.5 Additional Published-Protocol Comparisons

The main classification comparisons use the complete UCR112 and UEA26 30-resample benchmarks. RocketPFN's published TabPFN-v2.5 pipeline instead supports at most ten classes, so its authors report a smaller benchmark set rather than adding a multiclass decomposition (O'Rourke et al., 2026b). TS2TabPFN uses an explicit many-class wrapper and is therefore evaluated on the full benchmarks (Merlin & Silva, 2027). TimEE also tabulates TiCT on a smaller UEA supplied-split intersection (Küken et al., 2026). TIC-FM (Fang et al., 2026b) and MantisV2-LR (Feofanov et al., 2026) have per-dataset supplied-split scores tabulated by TimEE and ChorusTIC (Küken et al., 2026; Fang et al., 2026a), but no published results covering a complete 30-resample benchmark. The causal-DAG study publishes per-dataset supplied-split accuracies for its prior-finetuned TabPFN on a smaller eligible subset (O'Rourke et al., 2026a). CauKer evaluates synthetically pretrained encoders but reports aggregate UCR accuracies, not the per-dataset values needed for a matched comparison (Xie et al., 2025). The cited TSER comparators with aligned archive results appear in Table 19.

SeriesPFN's wins/ties/losses against RocketPFN ($G = 10$), using published 30-resample dataset means, are 52/2/38 on UCR92 ($p = 0.069$) and 14/0/6 on the 20 shared UEA datasets ($p = 0.245$). On the supplied split, the UCR112 counts are 99/6/7 against TIC-FM ($p < 0.001$), and 96/9/7 against MantisV2-LR ($p < 0.001$); TIC-FM has no published UEA row. On UEA, the supplied-split counts are 19/1/2 against TiCT (22 datasets; $p < 0.001$) and 22/3/1 against MantisV2-LR (26 datasets; $p < 0.001$). Against the causal-DAG study's eight-member prior-finetuned TabPFN, the counts are 41/3/16 on 60 UCR datasets ($p < 0.001$) and 3/0/0 on 3 UEA datasets ($p = 0.250$). All p-values are Holm-adjusted within each benchmark and score protocol.

D.6 Inference Time and Memory

We time prediction on each dataset from an initialized process. One fixed synthetic task initializes weights and runtime libraries outside the timed region; the measured pass uses the supplied benchmark split. Timing includes method-specific preprocessing or feature extraction, dataset-specific fitting or context construction, and prediction. It excludes process launch, checkpoint loading, initialization, transport, result validation, and artifact writes. Mean runtime is the arithmetic mean across datasets. Performance coordinates in Figure 1 come from the separate 30-resample evaluations.

Some methods are absent from Figure 1 because their timing runs were prohibitively slow and did not complete the benchmark within the evaluation budget. Because mean runtime is defined over all datasets, an incomplete run does not provide a runtime coordinate. We do not extrapolate the missing timings, so we omit these runtime points; their performance results remain included in the ranks.

GPU methods use one NVIDIA RTX PRO 6000 Blackwell Server Edition and 16 Intel Xeon 6724P cores. CPU-only methods use Intel Xeon 6724P or AMD EPYC 7543 nodes; their per-method core allocations are recorded with the timing data. The GPU marker includes methods with CPU

preprocessing; all timed stages count toward their plotted runtime. SeriesPFN uses eager model execution with native FlexAttention and Triton decoder kernels.

Benchmark	Deployment	Mean time (s) ↓	Median peak GPU memory (MB) ↓
UCR112	SeriesPFN	0.375	382
	TimEE	3.087	182
	ChorusTIC	2.820	14537
	TiCT	3.907	9444
	TS2TabPFN	138.83	412
	MASHT	39.06	881
	MultiROCKET-Hydra	3.383	N/A
	QUANT	5.43	N/A
	WEASEL 2.0	65.49	N/A
	RDST	39.78	N/A
	1NN-DTW	228.66	N/A
UEA26	SeriesPFN	18.61	1881
	TimEE-PV	45.12	118
	TimEE-VP	43.91	609
	ChorusTIC	54.59	30588
	TiCT-PV	120.55	9444
	TS2TabPFN	193.81	259
	MASHT	45.98	556
	ROCKET	11.86	N/A
	Arsenal	79.07	N/A
	DTW-D	431.90	N/A
TSER62	SeriesPFN	19.63	854
	TS2TabPFN	1684.18	642
	ROCKET	258.70	N/A
	MultiROCKET	718.38	N/A
	Random Forest	139.49	N/A

Table 20: Inference runtimes for all deployments in Figure 1 and peak allocated GPU memory where applicable. Runtime uses the figure's aggregation; peak GPU memory is the median across datasets. N/A denotes CPU-only deployments.

D.7 Additional Metrics

Tables 21 and 22 compare balanced accuracy and AUROC alongside NLL, multiclass Brier score, and ECE. Balanced accuracy gives each class equal weight. AUROC follows the time-series bake-off convention: the minority class for binary datasets and a training-frequency-weighted one-vs-rest average for multiclass datasets. Both tables use methods with complete values for all five metrics. The UCR archive provides one-hot outputs for MultiROCKET–Hydra, RDST, WEASEL 2.0, and 1NN-DTW; the UEA archive does so for DTW-D. These methods cannot enter the corresponding probability-metric ranks. We evaluate TS2TabPFN and TiCT with retained probabilities over the same 30 resamples. For regression, Table 23 compares point-prediction metrics, while Table 24 reports the two TSER62 methods with retained full predictive distributions.

Method	Mean per-dataset rank ↓				
	Bal. acc.	AUROC	NLL	Brier	ECE
SeriesPFN	3.85	3.62	3.31	3.70	5.29
TS2TabPFN	**3.78**	4.02	3.61	3.76	**4.35**
MASHT	4.34	**3.60**	**3.01**	**3.49**	4.61
HIVE-COTE 2.0	4.45	3.97	5.64	5.16	7.53
H-InceptionTime	5.34	6.35	6.92	5.72	5.58
LiteTime	5.76	6.67	6.14	5.56	4.79
QUANT	6.75	6.63	7.97	7.63	9.07
TimEE	6.86	6.76	5.98	6.71	5.91
FreshPRINCE	7.88	7.50	8.02	7.96	8.36
Proximity Forest	8.17	8.52	8.64	8.25	6.94
ChorusTIC	9.46	8.92	7.23	8.26	4.50
TiCT	11.35	11.43	11.52	11.80	11.09

Table 21: Mean per-dataset classification-metric ranks on UCR112; lower is better.

Method	Mean per-dataset rank ↓				
	Bal. acc.	AUROC	NLL	Brier	ECE
SeriesPFN	**4.31**	**3.75**	**3.23**	**3.85**	4.54
TS2TabPFN	5.52	5.35	4.42	4.96	5.88
MASHT	6.44	5.71	4.12	4.54	5.04
HIVE-COTE 2.0	5.10	4.42	6.31	6.35	9.00
Arsenal	6.90	10.96	12.92	9.31	9.50
ROCKET	7.46	10.42	12.31	9.00	8.96
DrCIF	7.77	7.33	7.19	7.42	8.73
HIVE-COTE 1.0	8.00	7.21	9.35	9.92	10.92
CIF	8.81	7.92	7.77	8.04	9.54
ChorusTIC	9.21	8.00	5.77	6.88	**4.15**
TiCT-PV	13.08	13.12	12.27	12.96	10.50
TimEE-PV	8.81	8.77	8.31	8.96	8.92
TimEE-VP	8.87	8.71	7.19	8.54	7.73
TDE	9.23	8.83	9.19	8.81	6.54
STC	10.50	9.50	9.65	10.46	10.04

Table 22: Mean per-dataset classification-metric ranks on UEA26; lower is better.

Relative RMSE divides each method's RMSE by its sum with the median RMSE of the fixed primary comparison slate on that dataset, making errors comparable across target scales. We average relative RMSE across datasets and report mean per-dataset ranks for MAE and R^2.

Method	Relative RMSE ↓	MAE rank ↓	R^2 rank ↓
TS2TabPFN	**0.430**	**1.48**	**1.58**
SeriesPFN	0.486	3.26	3.26
DrCIF	0.492	4.34	4.15
FreshPRINCE	0.479	4.34	4.18
InceptionTime ensemble	0.498	5.19	4.97
Rotation Forest	0.515	5.13	5.31
Random Forest	0.537	6.37	6.48
ROCKET	0.545	7.15	6.73
1NN-DTW	0.597	7.74	8.35

Table 23: TSER62 secondary point-prediction metrics over the complete common comparison. Relative RMSE is averaged across datasets; MAE and R^2 are summarized by mean rank.

Metric ↓	SeriesPFN mean [95% CI]	TS2TabPFN mean [95% CI]
Train-standardized CRPS	0.2578 [0.2548, 0.2609]	0.2359 [0.2326, 0.2392]
90% interval score	2.2342 [2.1947, 2.2737]	1.9769 [1.9311, 2.0227]
Absolute 90% coverage error	0.0499 [0.0492, 0.0507]	0.0380 [0.0370, 0.0390]
PIT–KS distance	0.1111 [0.1100, 0.1122]	0.1014 [0.1000, 0.1027]

Table 24: Benchmark means and resample-based confidence intervals for the two TSER62 methods that emit full predictive distributions.

E Ablations

E.1 Ablation Protocol and Results

We compare each ablation with a matched control trained on the full prior for 100k steps. Every arm and the control use three training seeds and the same evaluation protocol. The unit of analysis is the dataset: we average each dataset's metric over its 30 resamples and three training seeds, so each dataset contributes one paired arm–control difference. The ablations and their interventions are summarized in Table 25. The classification-only and regression-only arms use the same number of optimization steps as the control and therefore see more tasks of their retained type; the unused prediction head receives no gradient.

Ablation	Intervention	Benchmarks
No dynamics graph	Replace the source–graph system with an independent, target-dependent source for each channel.	All
Source-only target dependence	Always make the source parameters target-dependent instead of choosing the source or the combiners with equal probability; graph parameters remain target-independent.	All
Combiner-only target dependence	Always make selected graph-combiner parameters target-dependent instead of choosing the source or the combiners with equal probability; source parameters remain target-independent.	All
Both target-dependent	Make source and selected graph-combiner parameters depend on the same label or regression coordinate in every task.	All
No shared template	Set $\rho = 0$ in Equation 2, so each source consists of its per-series draw alone.	All
Independent settings	Fix $\gamma = 0$ for every target-dependent parameter group instead of drawing $\gamma \sim \mathcal{U}[0, 1]$, so class settings and regression endpoints are drawn independently.	All
Uniform regression targets	Draw each regression target independently from $\mathcal{U}[0, 1]$ instead of from the trajectory of an independent single-output SCM; classification tasks are unchanged.	All
No statistics embedding	Remove the level and scale statistics embedding; patch tokens carry only the robust-normalized values.	All
No channel attention	Restrict channel attention to each token itself, so channels interact only when learned queries pool their `[CLS]` tokens.	All
Classification only	Draw only classification tasks during training.	UCR, UEA
Regression only	Draw only regression tasks during training.	TSER

Table 25: Definitions of the matched ablations. Each arm changes the listed factor relative to the 100k-step control while retaining the remaining training and evaluation settings.

We report the ablation effect as arm minus control: accuracy-point differences for UCR112 and UEA26, and the percentage change in geometric-mean RMSE for TSER62. Thus, negative classification effects and positive regression effects favor the control. Confidence intervals are 95% dataset-bootstrap intervals. We use two-sided Wilcoxon signed-rank tests across datasets and apply Holm correction within each benchmark over its ten comparisons at $\alpha = 0.05$, separately for each

scope. The comparison over each complete benchmark is primary. Because the prior and training recipe were developed on meta-validation datasets, we repeat each comparison on the meta-test partition alone. Tables 26–28 report both scopes; W/T/L counts datasets from the arm's perspective.

Ablation	Scope	Δ accuracy [95% CI]	W/T/L	Holm p
No dynamics graph	Full (112)	$\mathbf{-4.81\ [-5.79, -3.91]}$	3/0/109	4.1×10^{-18}
	Meta-test (94)	$\mathbf{-4.91\ [-5.96, -3.94]}$	3/0/91	3.5×10^{-15}
Source-only	Full (112)	$\mathbf{-1.37\ [-1.94, -0.90]}$	17/1/94	5.8×10^{-13}
	Meta-test (94)	$\mathbf{-1.25\ [-1.66, -0.84]}$	13/1/80	5.3×10^{-11}
Combiner-only	Full (112)	$\mathbf{-0.54\ [-0.86, -0.21]}$	36/3/73	4.4×10^{-5}
	Meta-test (94)	$\mathbf{-0.66\ [-0.95, -0.38]}$	29/2/63	3.8×10^{-5}
Both target-dependent	Full (112)	$\mathbf{-0.69\ [-1.03, -0.36]}$	22/2/88	8.6×10^{-9}
	Meta-test (94)	$\mathbf{-0.85\ [-1.22, -0.50]}$	15/2/77	1.9×10^{-8}
No shared template	Full (112)	$\mathbf{-6.66\ [-8.48, -4.98]}$	14/2/96	1.3×10^{-14}
	Meta-test (94)	$\mathbf{-7.59\ [-9.68, -5.73]}$	10/2/82	1.3×10^{-13}
Independent settings	Full (112)	$\mathbf{-0.71\ [-1.12, -0.37]}$	35/4/73	3.8×10^{-5}
	Meta-test (94)	$\mathbf{-0.71\ [-1.17, -0.33]}$	29/4/61	2.9×10^{-4}
Uniform regression targets	Full (112)	$\mathbf{-0.14\ [-0.30, 0.00]}$	43/4/65	0.025
	Meta-test (94)	$-0.12\ [-0.29, 0.05]$	37/4/53	0.10
No statistics embedding	Full (112)	$\mathbf{-0.99\ [-1.44, -0.57]}$	36/0/76	1.9×10^{-6}
	Meta-test (94)	$\mathbf{-0.96\ [-1.39, -0.57]}$	31/0/63	2.9×10^{-5}
No channel attention	Full (112)	$\mathbf{-0.16\ [-0.27, -0.04]}$	42/2/68	8.1×10^{-4}
	Meta-test (94)	$\mathbf{-0.20\ [-0.33, -0.07]}$	33/2/59	3.2×10^{-4}
Classification only	Full (112)	$\mathbf{-0.39\ [-0.72, -0.09]}$	40/3/69	0.003
	Meta-test (94)	$\mathbf{-0.51\ [-0.87, -0.18]}$	31/2/61	0.001

Table 26: Matched ablations on UCR112. Effects are absolute accuracy-point differences, arm minus control; negative values favor the control. Bold effects are significant after benchmark-wide Holm correction at $\alpha = 0.05$.

Ablation	Scope	Δ accuracy [95% CI]	W/T/L	Holm p
No dynamics graph	Full (26)	$\mathbf{-8.21\ [-12.24, -4.83]}$	2/1/23	4.7×10^{-5}
	Meta-test (20)	$\mathbf{-8.04\ [-12.67, -4.22]}$	2/1/17	0.001
Source-only	Full (26)	$\mathbf{-3.05\ [-5.85, -0.34]}$	3/0/23	0.003
	Meta-test (20)	$\mathbf{-3.00\ [-6.51, 0.29]}$	3/0/17	0.039
Combiner-only	Full (26)	$-1.09\ [-2.48, 0.01]$	11/1/14	0.30
	Meta-test (20)	$-1.12\ [-2.82, 0.23]$	8/1/11	0.49
Both target-dependent	Full (26)	$\mathbf{-2.31\ [-4.18, -0.91]}$	9/1/16	0.030
	Meta-test (20)	$-2.54\ [-5.01, -0.76]$	7/1/12	0.11
No shared template	Full (26)	$\mathbf{-7.18\ [-11.63, -3.46]}$	1/0/25	6.0×10^{-7}
	Meta-test (20)	$\mathbf{-8.84\ [-14.29, -4.22]}$	0/0/20	1.9×10^{-5}
Independent settings	Full (26)	$-1.07\ [-2.30, -0.05]$	9/2/15	0.28
	Meta-test (20)	$-1.14\ [-2.75, 0.18]$	8/2/10	0.49
Uniform regression targets	Full (26)	$-0.50\ [-1.17, 0.14]$	7/1/18	0.28
	Meta-test (20)	$-0.68\ [-1.46, 0.05]$	5/1/14	0.20
No statistics embedding	Full (26)	$-0.92\ [-2.19, 0.44]$	9/0/17	0.30
	Meta-test (20)	$-1.06\ [-2.58, 0.48]$	8/0/12	0.49
No channel attention	Full (26)	$\mathbf{-2.14\ [-4.21, -0.52]}$	3/1/22	0.006
	Meta-test (20)	$\mathbf{-2.24\ [-4.79, -0.28]}$	2/1/17	0.039
Classification only	Full (26)	$-1.88\ [-4.48, 0.15]$	9/1/16	0.30
	Meta-test (20)	$-2.30\ [-5.53, 0.26]$	6/1/13	0.32

Table 27: Matched ablations on UEA26, following the conventions of Table 26.

Ablation	Scope	Δ gmean RMSE [95% CI]	W/T/L	Holm p
No dynamics graph	Full (62)	**+19.0% [+14.1, +24.4]**	3/0/59	2.7×10^{-9}
	Meta-test (50)	**+18.7% [+13.3, +24.6]**	3/0/47	1.8×10^{-9}
Source-only	Full (62)	**+8.1% [+4.6, +11.6]**	14/0/48	2.5×10^{-5}
	Meta-test (50)	**+7.5% [+3.7, +11.3]**	11/0/39	9.3×10^{-5}
Combiner-only	Full (62)	+0.0% [−1.6, +1.7]	27/0/35	0.95
	Meta-test (50)	−0.8% [−2.4, +0.8]	23/0/27	0.98
Both target-dependent	Full (62)	+1.6% [−0.0, +3.4]	23/0/39	0.25
	Meta-test (50)	+0.8% [−0.6, +2.5]	21/0/29	0.98
No shared template	Full (62)	**+12.4% [+6.9, +19.2]**	14/0/48	1.4×10^{-5}
	Meta-test (50)	**+10.2% [+5.2, +17.1]**	12/0/38	9.3×10^{-5}
Independent settings	Full (62)	+0.8% [−0.5, +2.1]	19/0/43	0.15
	Meta-test (50)	+1.2% [−0.1, +2.7]	15/0/35	0.098
Uniform regression targets	Full (62)	**+6.0% [+1.4, +11.8]**	16/0/46	0.015
	Meta-test (50)	+3.4% [−0.1, +6.8]	13/0/37	0.058
No statistics embedding	Full (62)	**+15.5% [+9.5, +22.5]**	12/0/50	1.5×10^{-5}
	Meta-test (50)	**+16.3% [+9.5, +24.5]**	10/0/40	2.7×10^{-5}
No channel attention	Full (62)	+0.6% [−0.5, +1.7]	22/0/40	0.25
	Meta-test (50)	+0.5% [−0.6, +1.6]	16/0/34	0.30
Regression only	Full (62)	−0.9% [−2.3, +0.6]	33/0/29	0.39
	Meta-test (50)	−0.4% [−1.9, +1.3]	27/0/23	0.98

Table 28: Matched ablations on TSER62. Effects are percentage changes in geometric-mean RMSE, arm relative to control; positive values favor the control. Bold effects are significant after benchmark-wide Holm correction at $\alpha = 0.05$.

Dynamics graph. Removing the dynamics graph costs 4.81 and 8.21 accuracy points on UCR112 and UEA26 and raises geometric-mean RMSE on TSER62 by 19.0%: the largest loss on UEA26 and TSER62 and the second-largest on UCR112. Without the graph, each channel is generated by an independent target-dependent source. This removes cross-channel dependence and excludes tasks in which target dependence enters through graph combiners. The large loss on univariate UCR112 suggests that the graph matters beyond multivariate structure. We interpret it as the prior's main source of series diversity: random convolutions, combiners, and activations turn one source into several nonlinear or filtered views. When target dependence enters through the source, each view carries the same target in a different form, acting like a random augmentation that preserves the target. The source-only ablation also excludes combiner-dependent tasks but loses only 1.37 points on UCR112, suggesting that their absence alone does not explain the no-graph loss. The larger UEA26 loss is consistent with an additional contribution from dependence between channels.

Location of target dependence. Restricting target dependence to the source in every task costs 1.37 and 3.05 accuracy points and raises geometric-mean RMSE by 8.1%, whereas restricting it to the combiner parameters costs only 0.54 points on UCR112 and has no detectable effect on UEA26 or TSER62. The two forms are therefore not interchangeable. When target dependence enters through the source, it changes kernel properties such as periods and length scales, and every observed channel inherits that change unless the graph's nonlinearities obscure it. When it enters through the combiners, it changes how selected nodes transform a shared source, so targets differ in waveform shape and in relations between channels while the driving dynamics stay fixed, and only some channels are informative. A model trained only on source-dependent tasks never sees target variation localized to selected graph nodes or channels; the larger UEA26 loss is therefore consistent with it not learning to identify informative channels. Conversely, combiner-only training produces a small but significant UCR112 loss, so source-dependent tasks cannot be removed without cost.

Both forms of target dependence. Making both parameter groups target-dependent in every task costs 0.69 and 2.31 accuracy points on UCR112 and UEA26, with no detectable TSER62 difference. In this arm the source and combiner parameters vary with the target along independently sampled parameter paths indexed by the same class label or regression coordinate, so every training task carries two separate cues to the same target. We hypothesize that the model learns to rely on whichever cue is easier and gets less practice detecting either cue alone, as real tasks may require. A construction in which the two groups vary in a coupled way remains untested.

Shared template. Removing the shared template costs 6.66 and 7.18 accuracy points on UCR112 and UEA26, the largest UCR112 loss of any ablation, and raises geometric-mean RMSE on TSER62 by 12.4%. With $\rho = 0$, the source contains only independent per-example draws, so examples no longer share a task-level waveform component. The resulting losses are consistent with the benchmarks containing tasks in which examples with similar targets share common temporal structure.

Correlation between settings. Fixing $\gamma = 0$ costs 0.71 accuracy points on UCR112; UEA26 (-1.07) and TSER62 ($+0.8\%$) move in the same direction but are not significant after correction. At $\gamma = 0$, class settings and regression endpoints are drawn independently, so the prior loses tasks with similar class settings or regression endpoints that large values of γ produce. The UCR112 loss is consistent with the model benefiting from practice on classes that differ only slightly. The experiments do not explain why the effect is detectable only on UCR112.

Regression target distribution. Drawing regression targets uniformly raises geometric-mean RMSE on TSER62 by 6.0% and costs a small but significant 0.14 accuracy points on UCR112. The single-output SCM makes the empirical target distribution differ between tasks, so targets can be skewed, clustered, or leave gaps, whereas independently drawn uniform targets impose the same uniform sampling distribution in every task. A model trained only on this fixed target distribution may rely on it rather than infer the target distribution from the context, which real regression targets need not follow. Only regression tasks change in this arm, so the UCR112 loss, like the classification-only result, suggests that classification reuses features learned from regression tasks.

Statistics embedding. Removing the statistics embedding raises geometric-mean RMSE on TSER62 by 15.5% and costs 0.99 accuracy points on UCR112. Because each channel is normalized by its own median and interquartile range, the patch tokens then carry no explicit information about a series' level or scale relative to the other series in the task. The large TSER62 loss is consistent with relative level and scale providing useful information about continuous targets. The smaller UCR112 loss suggests that classification depends less strongly on this information.

Channel attention. Removing channel attention costs 0.16 and 2.14 accuracy points on UCR112 and UEA26, with no detectable TSER62 difference. Channels are then encoded independently and exchange information only when the learned queries pool their `[CLS]` tokens, so the model can combine summaries of each channel but cannot compare channels at matching time positions, for example to detect lags or co-movement. We interpret the UEA26 loss as evidence that such relations carry class information in multivariate classification. For univariate inputs, the channel-attention operation has no other channel to mix information with. The UCR112 loss therefore cannot result from removing cross-channel interactions at inference and instead reflects how the ablation changes the shared weights during training.

Classification-only training. Although it sees more classification tasks than the control, the classification-only model loses 0.39 points on UCR112; UEA26 moves in the same direction (-1.88) but is not significant after correction. This suggests that regression tasks may teach features that classification reuses: a regression target moves the generating parameters continuously, so the model must resolve graded parameter differences rather than only discrete contrasts.

Regression-only training. Although it sees more regression tasks than the control, regression-only training changes geometric-mean RMSE by only -0.9%, which is not significant. Sharing training with classification therefore has no detectable cost for regression.

Scope. These experiments measure component effects under the matched 100k-step budget; they do not establish that the effect sizes are unchanged at the 500k-step budget of the evaluated SeriesPFN checkpoint. A nonsignificant comparison should not be read as evidence of equivalence, and the interpretations above are hypotheses consistent with the measured effects, not separately tested mechanisms.

F PER-DATASET RESULTS

Tables 29–34 report the dataset-level means underlying the comparisons over the complete benchmarks. Classification accuracies are shown as proportions. Each cell reuses the dataset mean from the corresponding benchmark-wide comparison; Table 8 identifies published scores and local predictions.

Table 29: Per-dataset UCR112 accuracy (part 1 of 2). Across both parts, bold marks the best mean at displayed precision. MR-H is MultiROCKET–Hydra; H-Inc. is H-InceptionTime.

Dataset	SeriesPFN	MASHT	TimEE	ChorusTIC	HC2	MR-H	H-Inc.	RDST
ACSF1	0.808	0.859	0.709	0.659	0.848	0.834	0.844	0.833
Adiac	0.848	**0.866**	0.815	0.790	0.794	0.824	0.852	0.740
ArrowHead	0.873	0.858	0.890	0.788	**0.899**	0.897	0.892	0.878
Beef	0.834	0.770	**0.852**	0.588	0.721	0.763	0.690	0.742
BeetleFly	0.902	0.922	0.888	0.837	0.923	0.912	0.915	0.927
BirdChicken	0.938	0.913	0.885	0.928	0.948	0.910	0.942	0.913
BME	**1.000**	**1.000**	0.992	0.991	0.999	**1.000**	0.998	0.982
Car	**0.943**	0.899	0.889	0.759	0.921	0.928	0.911	0.938
CBF	0.995	0.998	0.991	0.997	0.998	0.993	0.995	0.991
Chinatown	0.971	0.972	**0.977**	0.933	0.969	0.966	0.954	0.968
ChlorineConcentration	0.760	0.714	0.639	0.647	0.763	0.783	**0.867**	0.760
CinCECGTorso	**1.000**	0.995	0.978	0.719	0.993	0.977	0.820	0.982
Coffee	**1.000**	0.996	0.999	0.957	**1.000**	**1.000**	0.999	0.995
Computers	0.860	0.866	0.760	0.836	0.862	0.867	0.868	0.822
CricketX	0.829	0.819	0.827	0.769	0.841	0.826	**0.857**	0.835
CricketY	0.860	0.844	0.837	0.793	0.844	0.844	**0.861**	0.847
CricketZ	0.836	0.838	0.843	0.788	0.867	0.847	**0.874**	0.856
Crop	**0.802**	0.775	0.775	0.741	0.762	0.778	0.793	0.768
DiatomSizeReduction	0.958	0.960	0.957	0.938	0.929	0.950	0.953	0.945
DistalPhalanxOutlineAgeGroup	**0.825**	0.815	0.805	0.812	0.813	0.807	0.766	0.804
DistalPhalanxOutlineCorrect	**0.849**	0.839	0.827	0.831	0.828	0.840	0.816	0.816
DistalPhalanxTW	**0.731**	0.712	0.687	0.696	0.709	0.696	0.676	0.705
Earthquakes	0.747	0.748	0.747	0.748	0.747	0.748	0.733	0.739
ECG200	0.880	0.878	0.881	0.822	0.896	0.905	0.903	0.889
ECG5000	**0.951**	0.947	0.948	0.939	0.948	0.948	0.943	0.947
ECGFiveDays	0.978	0.999	0.944	0.855	0.992	0.991	0.980	0.993
ElectricDevices	0.895	0.904	0.799	0.883	0.902	0.903	0.892	0.897
EOGHorizontalSignal	0.862	0.883	0.819	0.727	0.818	0.872	**0.895**	0.779
EOGVerticalSignal	0.813	**0.852**	0.788	0.653	0.781	0.808	0.820	0.727
EthanolLevel	**0.883**	0.672	0.840	0.485	0.816	0.721	0.828	0.738
FaceAll	0.976	0.964	0.978	0.896	**0.987**	0.984	0.984	0.986
FaceFour	0.978	0.906	0.954	0.805	0.979	0.903	0.921	**0.991**
FacesUCR	0.949	0.927	0.957	0.830	0.974	0.971	**0.976**	**0.976**
FiftyWords	0.790	0.828	0.812	0.703	0.822	**0.846**	0.834	0.838
Fish	0.972	0.980	0.920	0.892	0.982	0.982	0.972	**0.984**
FordA	0.959	0.956	0.942	0.872	0.955	0.955	0.958	0.949
FordB	0.934	0.930	0.918	0.864	0.934	0.931	**0.941**	0.929
FreezerRegularTrain	**0.999**	**0.999**	0.998	0.996	**0.999**	0.998	0.996	0.997
FreezerSmallTrain	0.990	**0.997**	0.966	0.957	0.994	0.988	0.945	0.980
GunPoint	**0.999**	0.998	0.984	0.980	0.998	0.996	0.993	0.995
GunPointAgeSpan	**0.999**	0.996	0.985	0.985	0.997	0.997	0.990	0.991
GunPointMaleVersusFemale	0.999	**1.000**	0.999	0.996	**1.000**	**1.000**	0.999	**1.000**
GunPointOldVersusYoung	**1.000**	**1.000**	**1.000**	0.998	**1.000**	**1.000**	**1.000**	**1.000**
Ham	**0.871**	0.822	0.844	0.745	0.860	0.851	0.857	0.848
HandOutlines	0.943	0.939	0.924	0.895	0.938	0.952	**0.959**	0.947
Haptics	**0.590**	0.540	0.545	0.507	0.553	0.536	0.547	0.571
Herring	0.597	0.616	0.576	0.572	0.619	0.631	**0.639**	0.638
HouseTwenty	0.982	0.979	0.923	0.957	**0.984**	0.981	0.968	0.974
InlineSkate	0.590	0.637	0.449	0.432	0.526	0.508	0.574	0.459
InsectEPGRegularTrain	**1.000**	**1.000**	0.976	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
InsectEPGSmallTrain	**1.000**	**1.000**	0.903	0.998	**1.000**	0.985	**1.000**	0.965
InsectWingbeatSound	**0.671**	0.654	0.656	0.510	0.657	0.667	0.624	0.660
ItalyPowerDemand	0.963	0.961	0.963	0.884	0.963	**0.965**	0.959	0.951
LargeKitchenAppliances	0.929	0.945	0.783	0.854	0.939	0.941	**0.955**	0.918
Lightning2	0.787	0.786	0.830	0.749	0.770	0.756	0.794	0.761
Lightning7	0.747	0.817	0.794	0.772	0.793	0.804	**0.822**	0.736
Mallat	**0.984**	0.972	0.934	0.952	0.977	0.953	0.960	0.955
Meat	0.988	0.984	**0.990**	0.976	0.984	0.984	0.979	0.982
MedicalImages	0.813	**0.828**	0.783	0.799	0.811	0.810	0.798	0.774
MiddlePhalanxOutlineAgeGroup	0.715	**0.733**	0.729	0.719	0.689	0.657	0.592	0.672
MiddlePhalanxOutlineCorrect	**0.855**	0.853	0.851	0.823	0.831	0.850	0.841	0.826
MiddlePhalanxTW	0.576	**0.592**	0.573	0.570	0.570	0.547	0.521	0.555
MixedShapesRegularTrain	**0.987**	0.984	0.969	0.952	0.975	0.982	0.977	0.979
MixedShapesSmallTrain	**0.964**	0.960	0.929	0.909	0.954	0.959	0.927	0.956
MoteStrain	**0.953**	0.924	0.883	0.877	0.928	0.904	0.872	0.932

Continued on next page

Table 29 continued

Dataset	SeriesPFN	MASHT	TimEE	ChorusTIC	HC2	MR-H	H-Inc.	RDST
NonInvasiveFetalECGThorax1	0.964	0.953	0.927	0.905	0.955	0.960	0.962	0.927
NonInvasiveFetalECGThorax2	0.966	0.964	0.930	0.913	0.964	0.968	0.967	0.942
OliveOil	0.899	0.913	0.913	0.822	0.888	0.898	0.844	0.900
OSULeaf	0.957	**0.979**	0.889	0.884	0.969	0.969	0.962	0.971
PhalangesOutlinesCorrect	0.855	0.859	0.815	0.812	0.843	0.859	0.855	0.829
Phoneme	0.375	**0.405**	0.332	0.286	0.397	0.365	0.337	0.341
PigAirwayPressure	0.813	**0.963**	0.147	0.472	0.957	0.728	0.897	0.781
PigArtPressure	0.972	0.967	0.744	0.832	**0.982**	0.927	0.935	0.932
PigCVP	0.924	0.909	0.780	0.740	0.930	0.853	0.907	0.907
Plane	**1.000**	**1.000**	0.992	**1.000**	**1.000**	**1.000**	0.998	**1.000**
PowerCons	0.987	0.972	0.977	0.952	0.976	0.980	0.987	0.980
ProximalPhalanxOutlineAgeGroup	0.856	0.851	0.845	0.842	0.857	0.845	0.824	0.846
ProximalPhalanxOutlineCorrect	0.897	0.898	0.886	0.859	0.892	**0.910**	0.906	0.879
ProximalPhalanxTW	0.814	0.813	0.804	0.800	0.813	0.801	0.800	0.797
RefrigerationDevices	0.763	0.801	0.690	0.656	**0.817**	0.749	0.765	0.750
Rock	**0.902**	0.871	0.836	0.788	0.877	0.874	0.667	0.872
ScreenType	0.711	0.719	0.581	0.634	**0.754**	0.681	0.715	0.646
SemgHandGenderCh2	**0.980**	0.972	0.947	0.919	0.973	0.946	0.890	0.921
SemgHandMovementCh2	0.904	0.901	0.817	0.800	0.903	0.770	0.587	0.695
SemgHandSubjectCh2	**0.980**	0.957	0.929	0.861	0.955	0.925	0.748	0.887
ShapeletSim	0.996	**1.000**	0.934	0.915	**1.000**	**1.000**	0.896	0.997
ShapesAll	0.916	0.930	0.915	0.883	0.931	**0.945**	0.938	0.940
SmallKitchenAppliances	0.840	0.851	0.771	0.824	0.842	0.833	0.786	0.824
SmoothSubspace	0.993	0.977	0.985	0.980	0.980	0.984	0.989	0.992
SonyAIBORobotSurface1	**0.972**	0.964	0.935	0.889	0.950	0.944	0.929	0.937
SonyAIBORobotSurface2	0.945	0.949	0.931	0.856	0.940	0.945	0.942	0.915
StarLightCurves	0.982	0.981	0.981	0.980	0.982	0.981	0.977	0.981
Strawberry	**0.982**	0.976	0.978	0.973	0.980	0.980	0.979	0.979
SwedishLeaf	**0.981**	0.974	0.972	0.947	0.962	0.972	0.969	0.963
Symbols	0.970	**0.975**	0.945	0.928	0.973	0.972	0.965	0.966
SyntheticControl	0.995	0.994	0.992	0.991	**0.999**	0.992	0.997	0.992
ToeSegmentation1	0.935	0.943	0.928	0.900	**0.955**	0.931	0.947	0.949
ToeSegmentation2	0.940	0.942	0.917	0.887	**0.964**	0.940	0.960	0.947
Trace	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
TwoLeadECG	0.992	**0.998**	0.950	0.960	**0.998**	0.997	0.988	0.995
TwoPatterns	0.998	**1.000**	0.997	0.981	**1.000**	**1.000**	**1.000**	**1.000**
UMD	0.973	0.987	0.946	**0.992**	0.986	0.985	0.978	0.975
UWaveGestureLibraryAll	0.981	0.976	0.981	0.873	0.975	0.983	0.950	0.982
UWaveGestureLibraryX	**0.886**	0.870	0.857	0.840	0.863	0.878	0.839	0.858
UWaveGestureLibraryY	**0.835**	0.811	0.800	0.759	0.790	0.814	0.783	0.774
UWaveGestureLibraryZ	0.834	0.827	0.795	0.785	0.801	0.822	0.784	0.797
Wafer	**1.000**	**1.000**	0.999	0.996	**1.000**	**1.000**	0.999	**1.000**
Wine	0.806	0.897	0.921	0.884	0.941	0.936	0.888	0.948
WordSynonyms	0.725	0.774	0.771	0.626	0.767	**0.787**	0.760	0.771
Worms	**0.791**	0.788	0.771	0.745	0.748	0.739	**0.791**	0.723
WormsTwoClass	**0.820**	0.795	0.806	0.768	0.803	0.789	0.797	0.778
Yoga	0.927	0.926	0.896	0.859	0.929	**0.936**	0.928	0.930

Table 30: Per-dataset UCR112 accuracy (part 2 of 2). Across both parts, bold marks the best mean at displayed precision. FP is FreshPRINCE; PF is Proximity Forest.

Dataset	SeriesPFN	TS2TabPFN	TiCT	LiteTime	WEASEL 2.0	QUANT	FP	PF	1NN-DTW
ACSF1	0.808	**0.867**	0.370	0.834	0.829	0.810	0.800	0.638	0.554
Adiac	0.848	0.833	0.476	0.833	0.818	0.818	0.808	0.722	0.603
ArrowHead	0.873	0.870	0.767	0.888	0.885	0.838	0.788	0.884	0.722
Beef	0.834	0.778	0.547	0.649	0.777	0.776	0.791	0.594	0.504
BeetleFly	0.902	0.900	0.863	0.923	**0.947**	0.872	0.882	0.860	0.775
BirdChicken	0.938	0.917	0.938	0.950	0.933	**0.955**	0.923	0.903	0.840
BME	**1.000**	**1.000**	0.666	0.992	0.992	**1.000**	0.999	0.999	0.881
Car	**0.943**	0.927	0.673	0.939	0.908	0.894	0.796	0.806	0.675
CBF	0.995	**0.999**	0.931	0.996	0.981	0.991	0.988	0.994	0.993
Chinatown	0.971	0.968	0.924	0.970	0.972	0.975	0.973	0.948	0.935
ChlorineConcentration	0.760	0.737	0.590	0.846	0.768	0.750	0.775	0.631	0.644
CinCECGTorso	**1.000**	0.981	0.692	0.877	0.926	0.973	0.949	0.938	0.669
Coffee	**1.000**	0.999	0.943	0.999	0.998	**1.000**	0.989	0.992	0.986
Computers	0.860	**0.889**	0.722	0.869	0.780	0.839	0.823	0.714	0.703
CricketX	0.829	0.835	0.637	0.845	0.815	0.765	0.724	0.800	0.767
CricketY	0.860	0.849	0.600	0.854	0.815	0.772	0.727	0.800	0.737
CricketZ	0.836	0.849	0.652	0.869	0.828	0.790	0.735	0.803	0.772
Crop	**0.802**	0.787	0.656	0.785	0.759	0.776	0.768	0.753	0.711
DiatomSizeReduction	0.958	0.955	0.831	0.945	0.949	0.912	0.917	0.957	**0.962**
DistalPhalanxOutlineAgeGroup	**0.825**	0.798	0.756	0.765	0.817	0.811	0.804	0.802	0.749
DistalPhalanxOutlineCorrect	**0.849**	0.844	0.781	0.820	0.835	0.835	0.833	0.823	0.758
DistalPhalanxTW	**0.731**	0.706	0.674	0.683	0.705	0.701	0.699	0.692	0.606
Earthquakes	0.747	0.741	0.744	0.724	0.745	**0.756**	0.751	0.750	0.702
ECG200	0.880	**0.908**	0.775	0.903	0.888	0.882	0.853	0.873	0.793
ECG5000	**0.951**	0.947	0.929	0.942	0.948	0.945	0.945	0.940	0.925
ECGFiveDays	0.978	0.992	0.820	0.990	0.983	0.987	**1.000**	0.883	0.775
ElectricDevices	0.895	**0.913**	0.777	0.875	0.883	0.878	0.908	0.842	0.754
EOGHorizontalSignal	0.862	0.888	0.583	0.862	0.750	0.839	0.769	0.824	0.591
EOGVerticalSignal	0.813	0.849	0.566	0.769	0.692	0.781	0.696	0.778	0.535
EthanolLevel	**0.883**	0.852	0.362	0.849	0.686	0.827	0.573	0.324	0.267
FaceAll	0.976	0.977	0.801	0.985	0.980	0.954	0.943	0.977	0.940
FaceFour	0.978	0.907	0.749	0.928	0.905	0.914	0.831	0.945	0.850
FacesUCR	0.949	0.959	0.665	0.972	0.955	0.902	0.889	0.956	0.899
FiftyWords	0.790	0.802	0.561	0.794	0.815	0.772	0.718	0.826	0.686
Fish	0.972	0.983	0.752	0.979	0.980	0.945	0.886	0.934	0.759
FordA	0.959	0.958	0.884	0.955	0.960	0.968	**1.000**	0.850	0.578
FordB	0.934	0.937	0.859	0.938	0.938	0.917	0.921	0.839	0.646
FreezerRegularTrain	**0.999**	**0.999**	0.886	0.998	0.996	**0.999**	0.994	0.942	0.901
FreezerSmallTrain	0.990	0.992	0.885	0.978	0.945	0.995	0.965	0.823	0.802
GunPoint	**0.999**	0.994	0.940	0.994	0.996	0.992	0.956	0.991	0.876
GunPointAgeSpan	**0.999**	0.996	0.972	0.987	0.993	0.990	0.978	0.997	0.989
GunPointMaleVersusFemale	0.999	**1.000**	0.995	**1.000**	0.998	0.999	0.994	0.999	0.974
GunPointOldVersusYoung	**1.000**	**1.000**	0.983	**1.000**	0.996	**1.000**	**1.000**	**1.000**	**1.000**
Ham	**0.871**	0.834	0.657	0.861	0.862	0.819	0.777	0.783	0.687
HandOutlines	0.943	0.953	0.888	0.944	0.941	0.923	0.915	0.912	0.857
Haptics	**0.590**	0.551	0.447	0.552	0.523	0.519	0.504	0.458	0.391
Herring	0.597	0.626	0.613	0.629	0.603	0.610	0.602	0.574	0.567
HouseTwenty	0.982	0.972	0.924	0.969	0.957	0.968	0.949	0.937	0.920
InlineSkate	0.590	**0.646**	0.363	0.557	0.454	0.581	0.534	0.561	0.394
InsectEPGRegularTrain	**1.000**	**1.000**	0.925	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
InsectEPGSmallTrain	**1.000**	**1.000**	0.863	**1.000**	0.980	**1.000**	**1.000**	**1.000**	**1.000**
InsectWingbeatSound	**0.671**	0.668	0.499	0.626	0.653	0.656	0.648	0.607	0.346
ItalyPowerDemand	0.963	0.963	0.861	0.960	0.953	0.962	0.961	0.956	0.922
LargeKitchenAppliances	0.929	0.951	0.746	0.945	0.860	0.879	0.893	0.810	0.796
Lightning2	0.787	0.790	0.764	0.800	0.708	0.781	0.768	**0.849**	0.821
Lightning7	0.747	0.812	0.641	0.820	0.760	0.747	0.733	0.792	0.706
Mallat	**0.984**	0.975	0.543	0.964	0.952	0.961	0.964	0.972	0.937
Meat	0.988	0.986	0.755	0.979	0.983	0.984	0.981	0.987	0.971
MedicalImages	0.813	0.827	0.743	0.776	0.782	0.812	0.805	0.771	0.737
MiddlePhalanxOutlineAgeGroup	0.715	0.651	0.656	0.619	0.717	0.704	0.676	0.659	0.562
MiddlePhalanxOutlineCorrect	**0.855**	0.853	0.764	0.842	0.834	0.825	0.846	0.824	0.713
MiddlePhalanxTW	0.576	0.574	0.548	0.522	0.586	0.583	0.580	0.549	0.501
MixedShapesRegularTrain	**0.987**	0.985	0.911	0.976	0.972	0.968	0.950	0.964	0.853
MixedShapesSmallTrain	**0.964**	0.959	0.808	0.946	0.949	0.934	0.898	0.928	0.794
MoteStrain	**0.953**	0.903	0.872	0.869	0.945	0.923	0.872	0.915	0.830
NonInvasiveFetalECGThorax1	0.964	**0.965**	0.773	0.959	0.932	0.938	0.932	0.907	0.789
NonInvasiveFetalECGThorax2	0.966	**0.970**	0.803	0.965	0.957	0.955	0.945	0.938	0.861
OliveOil	0.899	0.910	0.397	0.791	**0.931**	0.902	0.879	0.879	0.868
OSULeaf	0.957	0.972	0.721	0.965	0.973	0.880	0.870	0.859	0.620
PhalangesOutlinesCorrect	0.855	**0.865**	0.783	0.856	0.844	0.850	0.848	0.829	0.752
Phoneme	0.375	0.355	0.179	0.330	0.338	0.373	0.356	0.320	0.233
PigAirwayPressure	0.813	0.478	0.139	0.813	0.863	0.453	0.454	0.275	0.182
PigArtPressure	0.972	0.908	0.341	0.929	0.958	0.889	0.871	0.716	0.457

Continued on next page

Table 30 continued

Dataset	SeriesPFN	TS2TabPFN	TiCT	LiteTime	WEASEL 2.0	QUANT	FP	PF	1NN-DTW
PigCVP	0.924	0.625	0.251	0.881	**0.944**	0.641	0.716	0.500	0.306
Plane	**1.000**	**1.000**	0.995	**1.000**	**1.000**	**1.000**	0.995	**1.000**	0.999
PowerCons	0.987	0.984	0.921	0.989	0.926	**0.992**	0.958	0.987	0.889
ProximalPhalanxOutlineAgeGroup	0.856	0.845	0.810	0.824	**0.859**	0.850	0.841	0.840	0.771
ProximalPhalanxOutlineCorrect	0.897	0.908	0.827	0.905	0.897	0.901	0.896	0.866	0.810
ProximalPhalanxTW	0.814	0.807	0.776	0.803	0.802	**0.816**	0.813	0.791	0.740
RefrigerationDevices	0.763	0.800	0.607	0.735	0.771	0.744	0.747	0.672	0.571
Rock	**0.902**	0.821	0.609	0.770	0.877	0.791	0.847	0.775	0.655
ScreenType	0.711	0.725	0.600	0.711	0.562	0.674	0.672	0.572	0.468
SemgHandGenderCh2	**0.980**	0.971	0.853	0.897	0.943	0.955	0.960	0.963	0.926
SemgHandMovementCh2	0.904	**0.927**	0.602	0.575	0.667	0.853	0.886	0.891	0.799
SemgHandSubjectCh2	**0.980**	0.969	0.832	0.740	0.911	0.925	0.941	0.938	0.876
ShapeletSim	0.996	**1.000**	0.904	0.718	**1.000**	0.984	**1.000**	0.789	0.656
ShapesAll	0.916	0.918	0.699	0.933	0.941	0.887	0.891	0.890	0.766
SmallKitchenAppliances	0.840	**0.852**	0.695	0.783	0.806	0.833	0.829	0.738	0.635
SmoothSubspace	0.993	0.984	0.827	0.980	0.976	0.992	0.972	**0.998**	0.823
SonyAIBORobotSurface1	**0.972**	0.946	0.889	0.939	0.954	0.908	0.942	0.920	0.808
SonyAIBORobotSurface2	0.945	0.944	0.842	0.938	**0.954**	0.922	0.933	0.899	0.848
StarLightCurves	0.982	**0.983**	0.976	0.977	0.981	0.981	0.979	0.980	0.915
Strawberry	**0.982**	0.981	0.919	0.977	0.978	0.976	0.966	0.960	0.954
SwedishLeaf	**0.981**	0.976	0.818	0.967	0.967	0.959	0.948	0.953	0.785
Symbols	0.970	0.974	0.889	0.972	0.970	0.949	0.943	0.967	0.940
SyntheticControl	0.995	0.995	0.974	0.997	0.984	0.993	0.996	0.998	0.992
ToeSegmentation1	0.935	0.932	0.878	0.953	0.954	0.852	0.854	0.836	0.736
ToeSegmentation2	0.940	0.941	0.876	0.958	0.953	0.884	0.898	0.886	0.841
Trace	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
TwoLeadECG	0.992	0.993	0.894	0.991	0.997	0.985	0.967	0.982	0.918
TwoPatterns	0.998	**1.000**	0.702	**1.000**	0.995	**1.000**	0.997	**1.000**	**1.000**
UMD	0.973	0.987	0.980	0.964	0.971	0.971	0.982	0.954	0.843
UWaveGestureLibraryAll	0.981	**0.984**	0.911	0.916	0.961	0.968	0.962	0.973	0.896
UWaveGestureLibraryX	**0.886**	**0.886**	0.775	0.840	0.841	0.852	0.837	0.831	0.728
UWaveGestureLibraryY	**0.835**	0.824	0.711	0.764	0.775	0.775	0.769	0.767	0.625
UWaveGestureLibraryZ	0.834	**0.839**	0.712	0.783	0.788	0.790	0.778	0.767	0.656
Wafer	**1.000**	**1.000**	0.988	0.999	**1.000**	**1.000**	**1.000**	0.996	0.984
Wine	0.806	0.916	0.727	0.641	**0.957**	0.913	0.897	0.856	0.859
WordSynonyms	0.725	0.757	0.515	0.723	0.751	0.701	0.629	0.778	0.646
Worms	**0.791**	0.776	0.635	0.778	0.718	0.728	0.773	0.693	0.574
WormsTwoClass	**0.820**	0.786	0.771	0.805	0.779	0.802	0.794	0.769	0.675
Yoga	0.927	0.934	0.823	0.920	0.925	0.894	0.924	0.887	0.847

Table 31: Per-dataset UEA26 accuracy (part 1 of 2). Across both parts, bold marks the best mean at displayed precision. HC2 denotes HIVE-COTE 2.0.

Dataset	SeriesPFN	MASHT	TimEE-PV	ChorusTIC	HC2	Arsenal	ROCKET	DrCIF
ArticularyWordRecognition	0.994	0.990	0.979	0.985	**0.996**	**0.996**	**0.996**	0.979
AtrialFibrillation	**0.389**	0.220	0.289	0.382	0.282	0.260	0.269	0.233
BasicMotions	**1.000**	**1.000**	0.989	**1.000**	0.989	0.989	0.989	**1.000**
Cricket	**1.000**	0.995	0.987	0.988	**1.000**	**1.000**	**1.000**	0.993
DuckDuckGeese	0.658	**0.719**	0.448	0.522	0.499	0.467	0.455	0.576
EigenWorms	**0.953**	0.947	0.766	0.868	0.894	0.848	0.852	0.906
Epilepsy	0.998	0.997	0.998	0.989	0.998	0.991	0.991	0.987
ERing	0.984	0.978	0.949	0.910	**0.985**	0.980	0.980	0.978
EthanolConcentration	0.783	0.557	0.745	0.402	0.791	0.477	0.474	0.671
FaceDetection	0.590	0.537	0.503	0.573	**0.714**	0.708	0.694	0.669
FingerMovements	0.545	0.547	0.561	0.548	0.552	0.559	0.547	0.557
HandMovementDirection	**0.624**	0.332	0.352	0.269	0.398	0.446	0.445	0.463
Handwriting	0.351	0.359	0.455	0.347	0.563	0.551	0.548	0.344
Heartbeat	**0.791**	0.753	0.724	0.763	0.729	0.725	0.725	0.764
Libras	0.950	**0.953**	0.870	0.923	0.927	0.895	0.893	0.911
LSST	0.692	0.690	0.622	0.677	0.637	0.633	0.619	0.553
MotorImagery	0.465	0.520	0.523	0.522	0.532	0.534	0.533	0.496
NATOPS	**0.933**	0.858	0.812	0.864	0.892	0.889	0.887	0.845
PEMS-SF	0.931	0.933	0.963	0.860	**0.998**	0.854	0.855	0.997
PenDigits	**0.997**	0.995	0.949	0.991	0.996	0.996	0.996	0.994
PhonemeSpectra	**0.393**	0.355	0.334	0.322	0.294	0.283	0.270	0.306
RacketSports	**0.938**	0.896	0.901	0.875	0.930	0.928	0.927	0.898
SelfRegulationSCP1	**0.920**	0.905	0.883	0.830	0.879	0.867	0.866	0.870
SelfRegulationSCP2	0.514	0.520	0.515	0.505	0.505	0.488	0.494	0.508
StandWalkJump	0.338	0.569	0.447	0.313	0.438	0.464	0.460	0.409
UWaveGestureLibrary	**0.958**	0.942	0.885	0.895	0.949	0.946	0.947	0.923

Table 32: Per-dataset UEA26 accuracy (part 2 of 2). Across both parts, bold marks the best mean at displayed precision. HC1 denotes HIVE-COTE 1.0; CIF is Canonical Interval Forest; STC is the Shapelet Transform Classifier; and TDE is the Temporal Dictionary Ensemble.

Dataset	SeriesPFN	TS2TabPFN	TimEE-VP	TiCT-PV	HC1	CIF	TDE	STC	DTW-D
ArticularyWordRecognition	0.994	0.992	0.990	0.966	0.980	0.979	0.977	0.975	0.989
AtrialFibrillation	**0.389**	0.278	0.269	0.298	0.293	0.251	0.302	0.318	0.236
BasicMotions	**1.000**	**1.000**	0.987	0.953	**1.000**	0.998	0.992	0.979	0.953
Cricket	**1.000**	0.999	0.998	0.952	0.993	0.984	0.993	0.989	**1.000**
DuckDuckGeese	0.658	0.653	0.405	0.395	0.476	0.560	0.322	0.435	0.492
EigenWorms	**0.953**	0.949	0.806	0.692	0.782	0.903	0.863	0.747	0.646
Epilepsy	0.998	0.995	0.998	0.954	**1.000**	0.984	0.997	0.987	0.963
ERing	0.984	0.976	0.964	0.812	0.943	0.957	0.939	0.843	0.929
EthanolConcentration	0.783	0.663	0.717	0.286	0.807	0.729	0.526	**0.824**	0.302
FaceDetection	0.590	0.649	0.503	0.618	0.692	0.689	0.606	0.698	0.533
FingerMovements	0.545	0.559	**0.563**	0.512	0.538	0.539	0.538	0.534	0.542
HandMovementDirection	**0.624**	0.353	0.335	0.339	0.378	0.522	0.377	0.350	0.303
Handwriting	0.351	0.398	0.431	0.209	0.504	0.351	0.559	0.288	**0.612**
Heartbeat	**0.791**	0.758	0.738	0.723	0.722	0.765	0.722	0.721	0.689
Libras	0.950	0.944	0.888	0.674	0.903	0.917	0.876	0.845	0.880
LSST	0.692	**0.698**	0.590	0.545	0.538	0.562	0.563	0.578	0.548
MotorImagery	0.465	0.512	0.511	0.536	0.522	0.518	**0.540**	0.508	0.521
NATOPS	**0.933**	0.897	0.815	0.774	0.829	0.844	0.816	0.844	0.820
PEMS-SF	0.931	0.952	0.950	0.890	0.980	**0.998**	**0.998**	0.984	0.771
PenDigits	**0.997**	0.996	0.977	0.957	0.972	0.990	0.972	0.977	0.993
PhonemeSpectra	**0.393**	0.301	0.292	0.132	0.329	0.266	0.233	0.306	0.154
RacketSports	**0.938**	0.896	0.902	0.814	0.906	0.893	0.891	0.881	0.856
SelfRegulationSCP1	**0.920**	0.911	0.882	0.811	0.860	0.859	0.835	0.847	0.818
SelfRegulationSCP2	0.514	0.511	0.501	0.496	0.517	0.489	0.527	0.516	**0.537**
StandWalkJump	0.338	**0.653**	0.389	0.427	0.407	0.451	0.362	0.440	0.220
UWaveGestureLibrary	**0.958**	0.940	0.906	0.754	0.913	0.924	0.927	0.870	0.923

Table 33: Per-dataset TSER62 RMSE (part 1 of 2). Across both parts, bold marks the lowest mean at displayed precision. TS2Tab is TS2TabPFN; FP is FreshPRINCE; IncE is the InceptionTime ensemble; and MR is MultiROCKET.

Dataset	SeriesPFN	TS2Tab	DrCIF	FP	IncE	MR
AcousticContaminationMadrid	3.730	**3.134**	4.120	3.281	4.077	4.263
AluminiumConcentration	**202.3**	215.8	245.2	270.0	247.2	248.0
AppliancesEnergy	2.051	**1.904**	2.405	2.054	4.234	2.903
BarCrawl6min	**0.04425**	0.04629	0.04480	0.04565	0.06786	3988
BeijingIntAirportPM25Quality	47.19	**46.17**	52.96	50.37	50.83	56.59
BeijingPM10Quality	70.54	**55.00**	70.57	66.51	63.85	89.99
BeijingPM25Quality	46.16	**33.47**	47.13	43.42	40.36	57.01
BenzeneConcentration	1.861	1.284	3.666	2.132	**0.2292**	1.349
BIDMC32HR	0.8507	0.7094	1.270	1.040	**0.5784**	0.7283
BIDMC32RR	1.167	**0.4203**	0.9154	0.6883	0.4994	0.4914
BIDMC32SpO2	0.3686	**0.2504**	0.4198	0.3408	0.2869	0.3373
BinanceCoinSentiment	0.3502	**0.3398**	0.3448	0.3461	0.3777	0.3907
BitcoinSentiment	0.1996	**0.1939**	0.1971	0.2012	0.2185	0.2040
BoronConcentration	2.045	**1.632**	1.763	1.880	1.710	2.027
CalciumConcentration	2147	**1633**	1939	2190	2472	2050
CardanoSentiment	0.2911	**0.2909**	0.2974	0.3017	0.3494	0.3296
ChilledWaterPredictor	2224	**2109**	2241	2479	2147	2420
CopperConcentration	1.784	**1.719**	1.826	1.846	1.841	1.839
Covid19Andalusia	0.01243	**0.01224**	0.01282	0.01261	0.01745	0.01302
Covid3Month	0.03934	0.03928	0.03977	**0.03830**	0.05472	0.1880
DailyOilGasPrices	**1.353**	1.366	1.594	1.490	2.030	1.774
DailyTemperatureLatitude	1.695	**1.125**	1.950	1.783	1.616	2.903
DhakaHourlyAirQuality	13.66	**0.5094**	3.171	1.784	4.053	4.256
ElectricityPredictor	517.5	**395.3**	587.4	582.7	589.5	596.4
ElectricMotorTemperature	6.161	**3.334**	4.656	4.650	4.728	7.379
EthereumSentiment	0.2313	**0.2231**	0.2234	0.2259	0.2520	0.2490
FloodModeling1	0.01523	**0.00378**	0.00946	0.00875	0.00643	0.01162
FloodModeling2	0.00841	0.00689	0.00518	0.00503	0.00548	**0.00405**
FloodModeling3	0.01206	**0.00343**	0.00644	0.00558	0.00636	0.00513
GasSensorArrayAcetone	0.1647	0.1695	0.1845	0.2011	0.2728	**0.1435**
GasSensorArrayEthanol	0.1300	**0.1271**	0.1518	0.1860	0.3293	0.1567
HotwaterPredictor	969.0	**895.1**	1246	1240	1156	1253
HouseholdPowerConsumption1	136.2	**47.98**	165.8	104.0	104.4	148.0
HouseholdPowerConsumption2	**25.86**	26.63	30.64	29.39	30.61	28.24
IEEEPPG	3.742	4.836	12.47	10.12	**3.698**	8.772
IronConcentration	**54.76**	55.83	62.13	65.35	64.44	59.62
LiveFuelMoistureContent	40.41	**40.15**	40.61	41.17	44.80	47.89
LPGasMonitoringHomeActivity	1.240	**1.220**	1.262	1.296	1.295	1.357
MadridPM10Quality	7.125	7.008	8.646	8.213	**6.951**	7.934
MagnesiumConcentration	169.6	**108.5**	160.8	168.0	127.5	133.2
ManganeseConcentration	**72.96**	78.47	81.28	84.63	85.30	87.77
MethaneMonitoringHomeActivity	0.5261	**0.5019**	0.5052	0.5305	0.5572	0.5761
MetroInterstateTrafficVolume	552.7	**550.2**	552.1	559.3	4739	3576
NaturalGasPricesSentiment	0.06351	0.05553	0.05968	0.05924	0.1508	0.07662
NewsHeadlineSentiment	0.1417	**0.1416**	0.1456	0.1456	0.1435	0.1565
NewsTitleSentiment	**0.1367**	**0.1367**	0.1405	0.1406	0.1385	0.1499
OccupancyDetectionLight	59.89	**51.11**	80.55	67.11	85.66	89.44
ParkingBirmingham	0.1195	**0.09470**	0.1170	0.1146	0.1299	0.1413
PhosphorusConcentration	18.86	**17.26**	19.06	19.15	17.77	18.98
PotassiumConcentration	175.0	**149.2**	167.7	174.9	172.4	168.9
PPGDalia	8.254	9.067	11.67	10.62	**6.681**	11.79
PrecipitationAndalusia	**0.3086**	0.3105	0.3851	0.3613	0.3794	0.3617
SierraNevadaMountainsSnow	37.09	33.90	43.00	36.92	**33.83**	47.81
SodiumConcentration	390.3	355.0	**312.8**	407.9	382.2	532.4
SolarRadiationAndalusia	0.6224	0.6197	0.6216	**0.6195**	0.6543	0.6315
SteamPredictor	**27582**	28616	31132	31015	29704	30577
SulphurConcentration	**176.9**	193.6	206.1	222.4	206.5	213.8
TetuanEnergyConsumption	1233	**1192**	1222	1283	10217	1529
VentilatorPressure	0.6369	0.4212	0.6166	0.6015	**0.3762**	0.7506
WaveDataTension	14.67	**14.62**	15.18	14.91	16.87	15.77
WindTurbinePower	145.4	**21.17**	97.85	44.66	283.5	113.0
ZincConcentration	**1.324**	1.433	1.429	1.589	1.543	1.382

Table 34: Per-dataset TSER62 RMSE (part 2 of 2). Across both parts, bold marks the lowest mean at displayed precision. RF is Random Forest; RotF is Rotation Forest; TSF is Time Series Forest; and 1NN is 1NN-DTW.

Dataset	SeriesPFN	ROCKET	RF	RotF	TSF	1NN
AcousticContaminationMadrid	3.730	4.633	5.072	5.126	4.602	5.440
AluminiumConcentration	**202.3**	247.1	293.3	234.1	245.1	382.6
AppliancesEnergy	2.051	2.769	3.986	2.558	3.753	5.739
BarCrawl6min	**0.04425**	0.05328	0.04807	0.04815	0.04651	0.06266
BeijingIntAirportPM25Quality	47.19	60.85	54.17	50.27	66.01	77.36
BeijingPM10Quality	70.54	94.56	71.71	66.32	84.66	118.2
BeijingPM25Quality	46.16	67.01	46.71	42.84	61.67	71.07
BenzeneConcentration	1.861	2.217	0.2832	0.7287	4.190	4.402
BIDMC32HR	0.8507	0.6563	5.901	4.885	3.987	1.945
BIDMC32RR	1.167	0.5265	1.943	1.711	1.417	1.234
BIDMC32SpO2	0.3686	0.3003	1.852	1.644	1.275	0.5257
BinanceCoinSentiment	0.3502	0.3872	0.3501	0.3538	0.3479	0.4844
BitcoinSentiment	0.1996	0.2688	0.1993	0.2014	0.2097	0.2603
BoronConcentration	2.045	2.131	2.316	1.885	2.039	1.994
CalciumConcentration	2147	2488	2963	1784	2344	4236
CardanoSentiment	0.2911	0.3065	0.2959	0.3053	0.2999	0.3931
ChilledWaterPredictor	2224	2518	2450	2437	2287	3350
CopperConcentration	1.784	2.042	1.951	1.863	1.902	2.481
Covid19Andalusia	0.01243	0.01420	0.01382	0.01326	0.01264	0.01541
Covid3Month	0.03934	0.04282	0.04018	0.04170	0.03895	0.05455
DailyOilGasPrices	**1.353**	2.275	1.708	1.559	1.685	2.742
DailyTemperatureLatitude	1.695	4.734	3.116	2.678	2.152	2.274
DhakaHourlyAirQuality	13.66	46.90	5.572	2.702	4.814	7.448
ElectricityPredictor	517.5	575.7	527.3	606.7	604.2	666.8
ElectricMotorTemperature	6.161	12.27	5.242	4.698	8.869	8.840
EthereumSentiment	0.2313	0.2871	0.2330	0.2294	0.2345	0.3188
FloodModeling1	0.01523	0.01730	0.01900	0.01799	0.01103	0.01382
FloodModeling2	0.00841	0.00927	0.00948	0.00790	0.00538	0.01109
FloodModeling3	0.01206	0.01729	0.01668	0.01568	0.00855	0.01307
GasSensorArrayAcetone	0.1647	0.1943	0.2594	0.1931	0.2118	0.3174
GasSensorArrayEthanol	0.1300	0.2136	0.2549	0.1499	0.1941	0.3425
HotwaterPredictor	969.0	1236	1310	1383	1401	1819
HouseholdPowerConsumption1	136.2	214.8	239.5	199.2	276.2	417.5
HouseholdPowerConsumption2	**25.86**	30.68	39.07	37.59	36.02	53.43
IEEEPPG	3.742	7.160	21.66	20.21	20.47	23.99
IronConcentration	**54.76**	59.65	72.91	61.92	63.82	88.67
LiveFuelMoistureContent	40.41	44.51	41.09	41.34	40.74	56.15
LPGasMonitoringHomeActivity	1.240	1.349	1.303	1.289	1.281	1.700
MadridPM10Quality	7.125	8.458	8.357	8.117	9.062	10.44
MagnesiumConcentration	169.6	144.9	244.7	166.5	193.0	298.2
ManganeseConcentration	**72.96**	87.29	93.77	79.91	84.11	122.0
MethaneMonitoringHomeActivity	0.5261	0.6239	0.5254	0.5244	0.5171	0.6934
MetroInterstateTrafficVolume	552.7	613.5	565.3	567.5	552.4	778.7
NaturalGasPricesSentiment	0.06351	0.09088	0.05598	0.05859	**0.05480**	0.07428
NewsHeadlineSentiment	0.1417	0.1468	0.1462	0.1495	0.1422	0.1979
NewsTitleSentiment	**0.1367**	0.1418	0.1411	0.1442	0.1371	0.1925
OccupancyDetectionLight	59.89	111.4	100.2	75.29	102.4	185.1
ParkingBirmingham	0.1195	0.1757	0.1199	0.1114	0.1206	0.1437
PhosphorusConcentration	18.86	23.40	23.61	20.64	20.35	22.15
PotassiumConcentration	175.0	209.2	228.0	186.2	183.9	242.0
PPGDalia	8.254	12.24	14.95	14.68	15.38	22.09
PrecipitationAndalusia	**0.3086**	0.3452	0.3945	0.3690	0.4023	0.4909
SierraNevadaMountainsSnow	37.09	68.55	52.22	48.69	51.81	63.58
SodiumConcentration	390.3	1010	470.0	652.7	397.0	659.4
SolarRadiationAndalusia	0.6224	0.6446	0.6304	0.6239	0.6256	0.8236
SteamPredictor	**27582**	30814	32219	31525	31358	42925
SulphurConcentration	**176.9**	272.0	223.1	215.1	213.1	251.9
TetuanEnergyConsumption	1233	1790	1269	1232	1402	2102
VentilatorPressure	0.6369	0.8529	0.6717	0.5685	1.848	1.136
WaveDataTension	14.67	15.14	15.36	15.21	15.44	21.34
WindTurbinePower	145.4	611.4	103.8	74.25	117.1	276.4
ZincConcentration	**1.324**	1.490	1.534	1.510	1.508	1.829