TIMEE: END-TO-END TIME SERIES CLASSIFICATION VIA IN-CONTEXT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Time series classification (TSC) is dominated by a two-stage paradigm: train a feature encoder—either from scratch on the target dataset or via pretraining on large corpora—and then fit a task-specific classifier on top. While effective, this decoupling optimizes representation learning independently of the classification objective, requires per-dataset training, and prevents the model from exploiting label information during inference. We introduce TIMEE[1], a 4.5M-parameter foundation model for end-to-end TSC via in-context learning. Given a labeled support set and a query time series, TIMEE outputs a predicted class distribution directly, with no encoder trained or classifier fitted on the target dataset. Following the prior-data fitted network (PFN) framework, TIMEE is meta-trained exclusively on synthetic TSC tasks, where each task contains time series with distinct class identities arising from structured distributional shifts in the generative process. Despite seeing no real time series during pre-training, TIMEE achieves the best mean rank in ROC AUC on the 128-dataset UCR archive, statistically tied with the strongest supervised baselines and ahead of every foundation model. It is also the best-calibrated and fastest of all compared methods. To our knowledge, TIMEE is the first purely synthetic-pretrained model to reach state-of-the-art performance. These results establish end-to-end ICL with synthetic priors as a compelling, largely unexplored direction for TSC, with scaling, prior design, and richer generation mechanisms as natural avenues for improvement.

1 INTRODUCTION

The dominant approach to time series classification (TSC) decomposes the problem into two stages: first, a feature encoder maps each time series to a fixed-dimensional representation; second, a task-specific classifier is trained on these representations to produce predictions. Despite the surface-level differences of existing methods, this two-stage design is universal. Classical methods such as MiniRocket (Dempster et al., 2021) and Hydra (Dempster et al., 2023), per-dataset deep learning methods such as TS2Vec (Yue et al., 2022), and recent foundation models such as Chronos-2 (Ansari et al., 2025), TiRex (Auer et al., 2025b), and MantisV2 (Feofanov et al., 2026) all instantiate it.

We argue that this is not merely an implementation convention, but a structural bottleneck. The encoder is optimized independently of the classification objective, and a new classifier must be fitted for every new dataset. Most critically, at inference time, the encoder cannot directly access the labeled training samples and therefore cannot produce representations tailored to the dataset's class structure.

In-context learning (ICL) eliminates all three of these downsides simultaneously. Rather than fitting per-dataset weights, an ICL model observes the labeled training set at inference and directly outputs a predicted class distribution in a single forward pass. No weight updates are required. This paradigm has proven transformative for tabular classification: TabPFN (Hollmann et al., 2023; 2025; Grinsztajn et al., 2026) and TabICL (Qu et al., 2026; 2025) achieve state-of-the-art performance by meta-training over synthetic labeled tasks drawn from a structured prior, with no per-dataset training at test time.

Extending this paradigm to TSC raises two requirements. First, a synthetic data-generation process. The tabular ICL literature has largely resolved synthetic pre-training through structured Bayesian priors over causal graphs. However, for time series, existing synthetic generation approaches either

[1] TIMEE pronounced as "Timmy"

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.

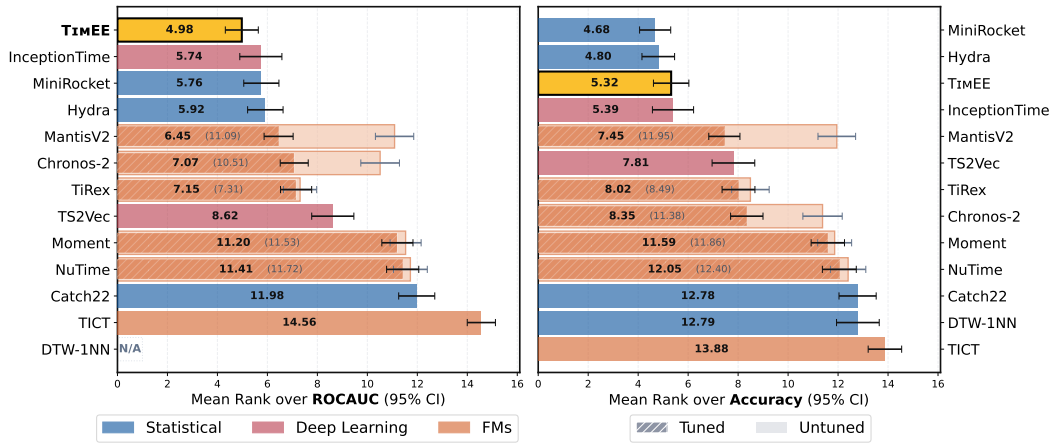


Figure 1: Mean rank across all 128 UCR datasets. TIMEE achieves the lowest mean rank on ROC AUC (left), statistically tied with INCEPTIONTIME, MINIROCKET, and HYDRA (Appendix Figure 14), and ahead of all foundation model baselines, while ranking third on accuracy (right). DTW-1NN is excluded from the ROC AUC comparison as it does not produce class scores.

produce unlabeled corpora (Xie et al., 2025) or assign class membership arbitrarily (Yeh et al., 2025). Neither approach provides the structured, label-meaningful datasets that ICL meta-training requires to learn a classification prior. Second, an architecture that builds class-aware representations by comparing series across the context. Existing time series encoders embed each series in isolation, blind to the other series in the dataset and their labels.

We introduce TIMEE, a 4.5M-parameter model for end-to-end TSC via ICL, built on a novel structured synthetic prior and an architecture with class-aware encoding. TIMEE is meta-trained exclusively on synthetic, labeled time series classification tasks and requires no per-dataset training at test time. Despite seeing no real time series during pre-training, it achieves the best mean ROC AUC rank across all 128 UCR datasets, statistically tied with state-of-the-art baselines (InceptionTime, MiniRocket, and Hydra) and ahead of every foundation model. It also delivers the best-calibrated predictions by a clear margin and lies on the Pareto front of inference speed versus predictive performance.

Our contributions are: **(I)** a new end-to-end ICL paradigm for TSC requiring no per-dataset training; **(II)** a VARX-based synthetic prior that generates labeled TSC tasks grounded in the data-generating process; **(III)** a transformer architecture that encodes and compares series with awareness of the labeled context; **(IV)** results matching the state-of-the-art on UCR, with the best probabilistic calibration and the fastest inference of all compared methods on datasets with $\leq$10 classes (25$\times$ faster than MiniRocket, over 1000$\times$ faster than InceptionTime).

2 BACKGROUND

Consider a time series classification problem with labeled training data $\mathcal{D}_{train} = (\mathbf{x}_i, y_i)_{i=1}^{n}$, where $\mathbf{x}_i \in \mathbb{R}^{T \times d}$ is a d-dimensional time series of length T and $y_i \in \mathcal{Y}$ is the corresponding class label. The goal is to estimate the predictive distribution $p(\cdot \mid \mathbf{x}_{test}, \mathcal{D}_{train})$ for a held-out test sample $\mathbf{x}_{test}$.

2.1 THE TWO-STAGE PARADIGM OF TIME SERIES CLASSIFICATION

The dominant approach to TSC decomposes the problem into two stages. First, a feature encoder $F_\theta : \mathbb{R}^{T \times d} \to \mathbb{R}^m$ maps each time series to a fixed-dimensional representation. Second, a task-specific classifier G_ϕ is trained on these representations to produce predictions $\hat{y} = G_\phi(F_\theta(x))$. We describe three general classes of methods below. See Appendix Table 1 for a detail comparison.

Classical and Time Series Features Engineering Methods define $F_\theta(x)$ without gradient-based learning, either through hand-crafted time series features or random convolutional projections. DTW-1NN uses dynamic time warping distance, while Catch22 (Lubba et al., 2019) extracts a curated set of 22 time series features. Rocket (Dempster et al., 2020) and MiniRocket (Dempster et al., 2021)

transform each series using a large bank of random convolutional kernels, aggregating features via global pooling. Hydra (Dempster et al., 2023) extends this by organizing kernels into competing groups to bridge random projections and dictionary-based pattern counting. Despite requiring no representation learning, these methods remain surprisingly competitive on standard benchmarks (Chen et al., 2015).

Per-Dataset Deep Learning Methods learns F_θ on each target dataset. InceptionTime (Ismail Fawaz et al., 2020) trains a convolutional encoder jointly with G_ϕ. TS2Vec (Yue et al., 2022) instead learns representations via hierarchical contrastive objectives over time and instance dimensions on the unlabeled target dataset, before fitting a downstream classifier head. These methods can achieve strong per-dataset performance but require training from scratch on every new dataset with no cross-task transfer.

Time Series Foundation Models (TSFM) amortize representation learning through large-scale pre-training. TSC-focused models, namely NuTime (Lin et al., 2024), MOMENT (Goswami et al., 2024), Mantis (Feofanov et al., 2025), and MantisV2 (Feofanov et al., 2026), pre-train encoders via contrastive or reconstruction objectives on large corpora of real and/or synthetic time series data, producing general-purpose embeddings that can be passed to lightweight downstream classifiers. As a complementary line of work, Auer et al. (2025a) studied transferring forecasting foundation models (notably Chronos-2 (Ansari et al., 2025) and TiRex (Auer et al., 2025b)) to classification using the extracted representations. In practice, however, competitive performance across all these models requires non-trivial layer selection: intermediate representations systematically outperform final-layer embeddings (Auer et al., 2025a).

Shared Structural Limitation Despite their differences, all of the above share a fundamental constraint: F_θ is optimized independently of the target classification task, and a new G_ϕ must be fitted per dataset. This decoupling has also been identified as a key source of sub-optimality in Shi et al. (2023).[2] Furthermore, F_θ encodes each series in isolation, without any awareness of the other labeled samples in the dataset, and therefore cannot exploit discriminative structure across classes during encoding. Most critically, at inference time, F_θ does not exploit the labeled training samples, and therefore cannot produce representations tailored to the class structure of the dataset at hand. TIMEE addresses these three constraints simultaneously via end-to-end in-context learning.

2.2 IN-CONTEXT LEARNING FOR SUPERVISED PREDICTION

In-context learning (ICL) offers an alternative to task-specific classifiers: rather than updating model weights, the model observes $\mathcal{D}_{train}$ at inference and approximates the conditional predictive distribution $p(\cdot \mid x_{test}, \mathcal{D}_{train})$ in a single forward pass.

This idea was first introduced in Prior-data Fitted Networks (PFN) (Müller et al., 2022). TabPFN (Hollmann et al., 2023) extended this paradigm for tabular classification by defining a synthetic Bayesian prior over tasks, while TabPFNv2 (Hollmann et al., 2025) scaled this substantially by introducing advancement in the synthetic priors and architecture. Parallel works that adopt the same paradigm include TabICL (Qu et al., 2025) and TabICLv2 (Qu et al., 2026) which introduce different architectures and synthetic priors.

For TSC, Fang et al. (2026) introduce ICL by pre-training an ICL head atop a fixed feature encoder. While this yields strong results, it remains structurally two-stage: the encoder is frozen and cannot be shaped by the classification objective. TiCT (Yeh et al., 2025) addresses the problem more directly by pre-training an end-to-end model on binary tasks, using threshold-based mixing of KernelSynth examples as synthetic prior. TIMEE extends this line of work to multi-class tasks and grounds the synthetic prior in a structured data-generating process that reflects meaningful class differences in the underlying dynamics. For a more in-depth discussion on the differences in the two approaches, we refer to Appendix A.2.

2.3 SYNTHETIC DATA GENERATION FOR TIME SERIES FOUNDATION MODELS

Due to the scarcity of high-quality labeled real-world time series data, synthetic generation has become a key enabler of zero-shot generalization in many time series foundation models. KernelSynth,

[2] InceptionTime is an exception to this limitation, as F_θ and G_ϕ are trained jointly on the target dataset.

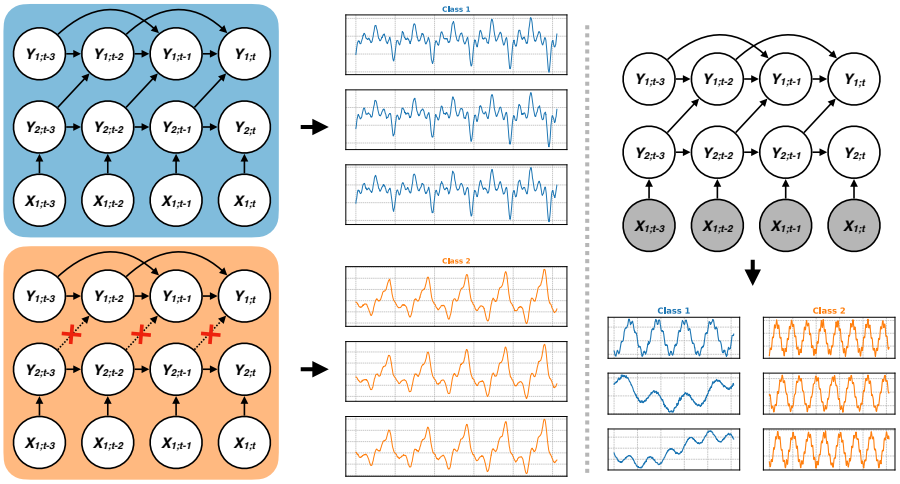


Figure 2: Class generation mechanisms in the TIMEE prior: ***Left – Structural Variation***: samples from Class 1 (blue) are generated under the full graph, while samples from Class 2 (orange) are generated under a graph with a subset of edges dropped (×); and ***Right – Signal Variation***: Class 1 (blue) and Class 2 (orange) share the same underlying signal pattern, and class identity is determined by the warping speed of the exogenous signal (grey nodes).

originally introduced in Chronos (Ansari et al., 2024), generates diverse synthetic time series for forecasting through compositions of Gaussian process kernels. Chronos-2 further introduces using temporal causal models in order to generate synthetic univariate time series.

In a separate direction, several works have proposed data mixing strategies to further diversify synthetic pre-training data. TiRex (Auer et al., 2025b) applies augmentation techniques including spike injection and convex mixing of base sequences, as well as an extended version of KernelSynth. CauKer (Xie et al., 2025) uses Structural Causal Models to produce clustering structure suitable for contrastive pre-training of time series encoders.

ICL pre-training for classification imposes a strictly harder requirement: synthetic data must consist of labeled classification tasks, not merely diverse time series. No prior work generates classification tasks in which class identity reflects differences in the data-generating process itself.

3 METHOD

3.1 PROBLEM FORMULATION AND TRAINING OBJECTIVE

While the two-stage paradigm estimates $p(\cdot \mid \mathbf{x}_{test}, \mathcal{D}_{train})$ anew for every dataset by fitting a classifier on encoded features, TIMEE learns an amortized predictor f_θ that approximates it in a single forward pass, with no per-dataset weight updates (see Section 2.2). Following the PFN (Müller et al., 2022) training objective, we train f_θ by meta-learning over synthetic tasks drawn from a prior p_{synth}. At each iteration, a synthetic dataset $\mathcal{D}_k = \{(x_i, y_i)\}_{i=1}^{N}$ is sampled and partitioned into:

$$\mathcal{D}_{k,train} = \{(x_j, y_j)\}_{j=1}^{n_{train}}, \quad \mathcal{D}_{k,test} = \{(x_l, y_l)\}_{l=1}^{n_{test}},$$

and the model minimizes the negative log-likelihood over query positions:

$$\mathcal{L}(\theta) = -\mathbb{E}_{\mathcal{D}\sim p_{synth}} \left[\sum_{(x,y)\in\mathcal{D}_{k,test}} \log p_\theta(y \mid x, D_{k,train}) \right]. \tag{1}$$

3.2 SYNTHETIC DATA GENERATION

The design of the prior p_{synth} is central to the performance of TIMEE. While prior work (Section 2.3) has established synthetic data generation as a promising data source for pre-training time series foundation models, these approaches target either forecasting or produce classification tasks in which

class identity does not reflect meaningful differences in the underlying generative process. We address this gap by pre-training TIMEE on synthetic labeled TSC datasets in which class structure arises from controlled variation in the data-generating process itself.

Data-Generating Process. The foundation of our data-generating process is a temporal structural model over d endogenous variables. Let $\mathbf{y}_t \in \mathbb{R}^d$ denote a multivariate[3] random variable at time t. We model its evolution as a Vector Autoregressive process with exogenous inputs (VARX) (Lütkepohl, 2005):

$$\mathbf{y}_t = \sum_{k=1}^{p} \mathbf{A}_k \mathbf{y}_{t-k} + \sum_{j=0}^{s} \mathbf{B}_j \mathbf{x}_{t-j} + \boldsymbol{\epsilon}_t, \qquad \boldsymbol{\epsilon}_t \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}) \tag{2}$$

where $\mathbf{A}_k \in \mathbb{R}^{d \times d}$ are the lag-k autoregressive coefficient matrices, encoding inter-variable temporal relationships, $\mathbf{x}_t \in \mathbb{R}^q$ is a q-dimensional exogenous input signal and $\mathbf{B}_j \in \mathbb{R}^{d \times q}$ maps lag-j exogenous inputs to endogenous variables.

Class Identity Mechanism. Classification structure is introduced through two complementary and independently controllable mechanisms:

- ***Structural Variation:*** Classes are defined by structural variation in the dependency structure encoded in $\mathbf{A}_k$ and $\mathbf{B}_j$ as shown in Figure 2 (left). Concretely, class identity is determined by the set of edges that are present (or absent) in the dependency graph. This reflects well-established observations that many real-world classification tasks correspond to qualitatively different regimes of inter-variable interaction (Qian et al., 2025).
- ***Signal Variation.*** Classes are defined by the statistical properties of the exogenous input $\mathbf{x}_t$ while the underlying dependency structure $\mathbf{A}_k$, $\mathbf{B}_j$ remains fixed. In particular, we consider warping of $\mathbf{x}_t$ as class-discriminative feature, as shown in Figure 2 (right). The same underlying signal is hereby expressed at different speeds across classes, inducing systematic distributional differences in the endogenous $\mathbf{y}_t$ without any change to the inter-variable structure (Kamycki et al., 2019).

Augmentations. To further enhance the diversity of our synthetic pre-training corpora, we apply a set of augmentations at the individual time series level. Concretely, we consider three transformation types: (1) *smoothing* via a randomized Gaussian filters; (2) *spike injection*: by sampling spike s_t and augmenting the series by $\mathbf{y}'_t = \mathbf{y}_t + s_t$ (Auer et al., 2025b); and (3) *nonlinear transformations*: by applying non-linearity on the time series, i.e. $\mathbf{y}'_t = g(\mathbf{y}_t)$ where g is a randomly sampled nonlinearity. Additionally, we apply a mixup scheme (Ansari et al., 2024; Auer et al., 2025b), to generate convex combinations of synthetic time series across the dataset level.

We provide details about sampling procedure from the prior in Appendix C.1 and details about the augmentations in Appendix C.2 and C.3. We ablate individual prior components in Appendix C.5.

3.3 MODEL ARCHITECTURE

TIMEE is designed to encode and compare all series in the context simultaneously. It processes the context in four sequential stages (Figure 3): **Tokenization**, an **Encoder Stack**, **In-Context Reasoning**, and a **Decoder**. A separate **Label Conditioning** path injects support labels at two points, before the encoder and before in-context reasoning. Throughout, $\mathcal{S}$ and $\mathcal{Q}$ denote the support and query indices among the N series in the context, and y_i the label of support series $i \in \mathcal{S}$. Each component is ablated in Appendix B.4.

Tokenization. Like many TSFMs (Section 2), we first normalize each series to remove the individual scale, in order to encourage the model to focus on temporal shape rather than magnitude. Like Chronos-2 (Ansari et al., 2025), we apply $sinh^{-1}$ transformation to better handle the outliers. A subsequent cross-normalization step, computed using only the training-series statistics, further standardizes the representation across the context. Each series is then divided into non-overlapping patches of size $p = 16$ and projected to $d = 128$ dimensions via a shared linear layer, giving patch embeddings $z_{i,p} \in \mathbb{R}^d$.

[3]The univariate case $d = 1$ is also covered by this model.

Label Conditioning. A key advantage of the end-to-end paradigm is that class labels are available throughout the forward pass. We exploit this by conditioning the encoder directly on label identity. Concretely, following TabICLv2 (Qu et al., 2026), a learned class embedding table $E_{\text{patch}} \in \mathbb{R}^{C_{\max} \times d}$ is added to every patch of each support series,

$$z_{i,p} \leftarrow z_{i,p} + E_{\text{patch}}[y_i], \qquad i \in \mathcal{S}, \tag{3}$$

while query series receive no such signal. This allows subsequent attention layers to build class-aware contextual representations that are more discriminative and separable across classes, while preserving the label ambiguity of query patches that the model must resolve through in-context reasoning. A second table $E_{\text{icl}} \in \mathbb{R}^{C_{\max} \times d_{\text{cls}}}$, with its own parameters, plays the same role for the compressed series summaries (below).

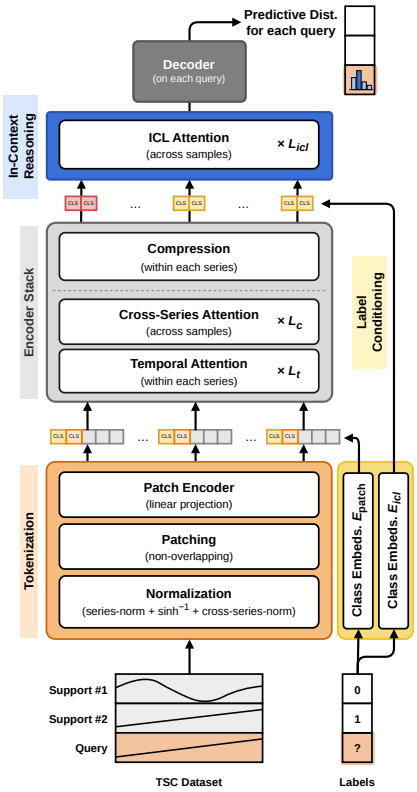


Figure 3: Architecture of TIMEE.

Encoder Stack. The encoder turns each series into a compact, class-aware summary that reflects both its own temporal structure and its relation to the rest of the context. $K = 4$ learnable `CLS` tokens are prepended to each series' patch sequence, yielding representations $\mathbf{Z} \in \mathbb{R}^{N \times (P+K) \times d}$. $L_t = 4$ *temporal* attention layers then operate within each series using RoPE (Su et al., 2021), and $L_c = 4$ *cross-series* attention layers operate across the N series at each patch position. Cross-series attention is masked so that support series attend only to other support series, and queries attend to support series but not to one another (Appendix B.2). Finally, a compression layer collapses each series into its K `CLS` vectors by attending only over patch positions, producing a compact series summary $\mathbf{r}_i \in \mathbb{R}^{K \times d}$, which is flattened to $\tilde{\mathbf{r}}_i \in \mathbb{R}^{d_{\text{cls}}}$ with $d_{\text{cls}} = K \cdot d = 512$.

In-Context Reasoning. The classification decision is made by comparing each query summary against the labeled support summaries. Support summaries are first re-conditioned on their labels,

$$\tilde{\mathbf{r}}_i \leftarrow \tilde{\mathbf{r}}_i + E_{\text{icl}}[y_i], \qquad i \in \mathcal{S}, \tag{4}$$

and all N summaries are processed by $L_{\text{icl}} = 2$ ICL attention layers of width d_{cls} under the same masking as the encoder.

Decoder. Each query summary passes through layer norm and a two-layer MLP producing $C_{\max}$ logits, of which the first C are used for a C-class dataset. The model is trained with cross-entropy loss over query positions (Equation 1).

Many-Class Extension. TIMEE is trained with at most $C_{\max} = 10$ classes. For $C > C_{\max}$, we use a one-vs-rest (OvR) decomposition strategy, reducing any C-class problem to C binary classification contexts at the cost of C forward passes. While more sophisticated strategies are possible, we find OvR to be a simple and effective solution; we leave richer many-class handling as future work.

3.4 PRETRAINING AND INFERENCE

Pretraining Setup. TIMEE is pretrained on 7M synthetic TSC tasks, derived from 1.5M unique VARX-generated datasets via augmentation (Section 3.2). Each task is constructed by uniformly sampling a support-query split and cropping all series to a common length sampled uniformly from $[16, 1024]$. We train for $30K$ steps on a single NVIDIA H200 GPU with `bf16` automatic mixed precision, taking approximately 40 hours. We use Muon (Jordan et al., 2024) for matrix parameters and AdamW (Loshchilov & Hutter, 2019) for all remaining weights. Details of pre-training setup are included in Appendix D.1.

Inference. Once pretrained, TIMEE requires no per-dataset training: a single forward pass over the labeled context yields the class distribution for all queries of the dataset. By default, we run this pass

on four preprocessed versions of the dataset, {raw, first-differenced} × {interpolated to 256, 512}, and average the resulting predictions (Appendix D.2). This test-time augmentation mirrors common practice: TabPFNv2, TabICLv2 and MantisV2 average over preprocessed inputs, and InceptionTime averages over five trained networks. All results in Section 4.2, including timings, use this default; Appendix G isolates the effect of the ensemble by reporting results with a single view.

4 EXPERIMENTS

We evaluate TIMEE against a broad set of state-of-the-art baselines for time series classification, assessing predictive performance, calibration, and inference efficiency.

4.1 EVALUATION SETUP

Datasets. We evaluate TIMEE on the UCR archive (Chen et al., 2015), a standard benchmark comprising 128 univariate TSC datasets, using the data splits and loading pipeline provided by *aeon* (Middlehurst et al., 2024)[4].

Baselines. We compare against a broad set of baselines spanning three categories (Section 2): classical and feature-based methods (DTW-1NN, Catch22, MiniRocket, Hydra), deep learning methods (InceptionTime, TSVec), and time series foundation models (NuTime, MOMENT, MantisV2, TiCT, Chronos-2, and TiRex).

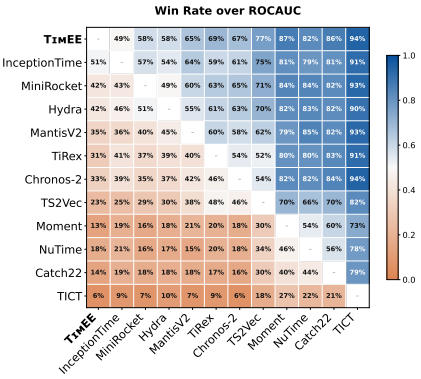


Figure 4: Win rate (ROC AUC) across 128 UCR datasets. TIMEE wins a majority of pairwise comparisons against every baseline except INCEPTIONTIME, with which it is evenly split (49%). Against the pre-trained foundation models, TIMEE wins 65–94% of datasets.

As discussed in Section 2.1, most foundation model baselines require tuning—such as selecting the optimal intermediate layer representation—to achieve competitive performance; we therefore report both default and tuned results for these models. Full tuning details are provided in Appendix E.

Regarding downstream classifier, for Catch22, MiniRocket, and Hydra, we use the *aeon* implementations with their canonical classifier pairings. For the remaining representation and foundation models, we select between Random Forest and SVM based on validation performance, with Random Forest generally preferred except for TS2Vec where SVM performs better. See Appendix E for more details.

4.2 MAIN RESULTS

Figure 1 illustrates the mean rank across 128 UCR datasets. TIMEE achieves the best mean rank on ROC AUC, statistically tied with INCEPTIONTIME, MINIROCKET, and HYDRA (Appendix Figure 14), and ahead of all foundation model baselines. While TIMEE ranks 3rd in Accuracy, its stronger ROC AUC suggests it is a more robust choice across varying deployment thresholds and class distributions (Provost et al., 1998). Notably, ridge-regression-based models like HYDRA and MINIROCKET rank slightly lower on ROC AUC than on accuracy: their scores are optimised for the argmax rather than for ranking, and thus carry less ordering information than the class distributions produced by TIMEE.[5] We present full per-dataset results in Appendix F.3.

Probabilistic Calibration. To evaluate whether these scores represent well-calibrated probabilities, we report mean rank over log-loss for datasets with ≤ 10 classes[6] ($n = 103$) in Figure 5. TIMEE achieves the top rank by a clear margin. As Log Loss is a strictly proper scoring rule, this indicates that TIMEE provides better-calibrated probabilistic estimates. The gap in Accuracy suggests that while ridge-based methods may force correct hard-label decisions on boundary samples via their objective function, TIMEE maintains a more reliable representation of uncertainty, yielding more robust inputs for downstream decision-making.

[5]Both use `RidgeClassifierCV`, which regresses onto ± 1 class targets and predicts via argmax; the scores are not trained as log-odds. We use the decision-function outputs for ROC AUC and log loss; see Appendix E.

[6]Restricted to ≤ 10 classes, since beyond that TIMEE requires OvR decomposition, whose normalization artifacts would confound comparison against native multi-class baselines; see Figure 13 for > 10 classes.

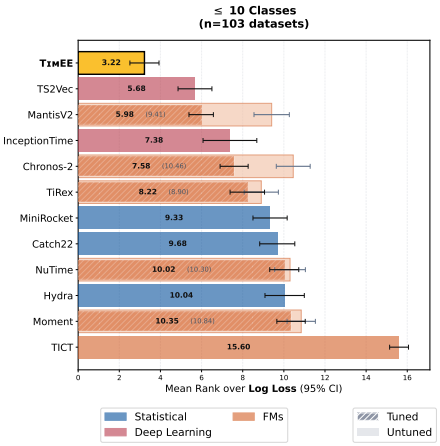


Figure 5: Mean rank on log loss across UCR datasets with at most 10 classes ($n = 103$). TIMEE achieves the lowest mean rank, indicating well-calibrated predictive distributions.

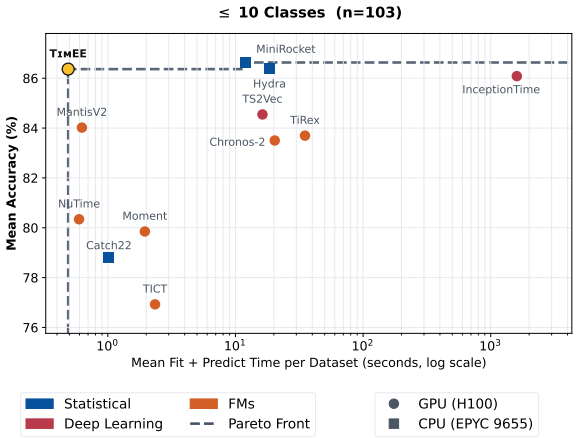


Figure 6: Pareto front of inference speed vs. accuracy on UCR datasets with at most 10 classes ($n = 103$). TIMEE lies on the Pareto front, matching the accuracy of MiniRocket and InceptionTime with $25\times$ and over $1000\times$ faster inference, respectively.

Inference Speed. TIMEE's end-to-end ICL mechanism enables fast inference without any per-dataset training. While most other methods require a separate, often costly, training phase, TIMEE collapses the traditional fit+predict stages into a single forward pass per view (Section 3.4). As a result, as shown in Figure 6, TIMEE lies on the Pareto front of inference speed vs. predictive performance; all reported timings include the full four-view ensemble. Note that this comparison is restricted to datasets with at most 10 classes, as OvR is required for larger class counts which introduces additional inference overhead (see Appendix Figure 13). This remains a limitation of TIMEE and is discussed in Section 5.

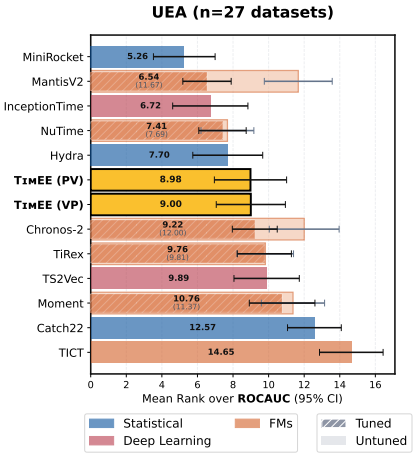


Figure 7: Mean rank over ROC AUC on 27 UEA datasets. We compare two variants of TIMEE multivariate extensions: (i) *PV* denotes *per-variate* variant; (ii) *VP* denotes *variate-pooling* variant. TIMEE achieves competitive results despite being primarily developed for the univariate setting.

4.3 ABLATIONS

We present ablation studies on: **(I)** examining how TIMEE can be extended to multivariate tasks; and **(II)** probing the internal mechanisms that drive its in-context classification performance.

Multivariate Extension. We consider two strategies for adapting TIMEE to multivariate settings: (i) *Per-variate*: predict independently on each variate and average the resulting class probabilities; and (ii) *Variate-pooling* (details in Appendix B.3): encode each variate independently and aggregate the M resulting `CLS` token sets via attention pooling before the ICL phase. The *variate pooling* variant is fine-tuned on synthetic multivariate data generated with our prior (Section 3.2). We evaluate both strategies against the baselines (Section 4) on 27 of the 30 UEA datasets[7]. As shown in Figure 7, the *per-variate* variant achieves competitive performance despite treating variates independently. Fine-tuning the *variate-pooling* variant, however, only yields a marginal gain. We attribute this to a limitation of our synthetic data generation: although the prior is natively extensible to multivariate inputs, the per-variate discriminative signal appears sufficient in practice, leaving little room for cross-variate aggregation to help. We discuss this further in Appendix C.1.2.

[7]EigenWorms and PenDigits are excluded because some baselines cannot run on them; InsectWingbeat is excluded because neither several baselines nor TIMEE can run on it (Appendix F.4).

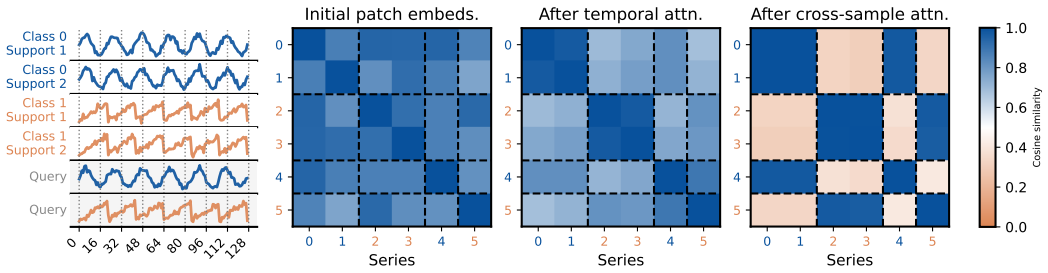


Figure 8: Representation alignment across encoder stages. The leftmost panel shows six time series from a 2-shot binary classification task on synthetic data: class *0* (blue, sinusoidal) and class *1* (orange, sawtooth). The three heatmaps show pairwise cosine similarity of series representations at successive stages of the encoder, comparing the patch embeddings at (i) initial state; (ii) after temporal attention; (iii) after cross-series attention.

How the Encoder Stack Builds Class-Discriminative Representations. To understand how the encoder progressively builds class-discriminative representations, Figure 8 traces pairwise cosine similarities across its successive stages. Initially, patch embeddings are largely undifferentiated – similarities are uniformly high across all series regardless of class membership. Temporal attention begins to differentiate series by refining within-series structure, and same-class representations grow slightly more similar to one another. Cross-series attention then sharply accentuates this emerging structure akin to a contrastive separation in representation space. Crucially, query series — which receive no class-label signal during the forward pass — nonetheless align to their correct class cluster, confirming that the encoder builds class-discriminative representations that prime the subsequent in-context reasoning stage.

5 Conclusion

We introduced TimEE, a foundation model for time series classification that departs from the two-stage paradigm by framing TSC as an in-context learning problem. The key enabler is a synthetic prior grounded in a structured data-generating process, which produces labeled classification tasks where class identity reflects meaningful differences in the underlying dynamics. To our knowledge, no prior work addresses this for TSC. Our strong empirical results on the 128 UCR datasets suggest that the gap between the two-stage paradigm and end-to-end ICL is not merely from architecture, but also from the right synthetic prior and the ability to leverage class structure at inference time. We hope this work encourages further exploration of prior design, model scale, and richer class generation mechanisms, which is likely to lead to further substantial performance improvements.

Limitation 1: Many-class handling. TimEE is pre-trained with at most 10 classes and relies on a one-vs-rest decomposition for larger label spaces, which introduces significant inference overhead and diminishes the speed advantage of ICL-based classification (Figure 13). Extending TimEE to natively support larger class counts is an important direction for future work.

Limitation 2: Multivariate classification. Extending TimEE to multivariate inputs via variate pooling yields only marginal gains over the simpler per-variate baseline, which we attribute to a limitation of the VARX prior: class-discriminative information tends to concentrate in individual variates, leaving little signal for cross-variate aggregation to exploit. Designing priors that better leverage inter-variate structure remains an open problem. Beyond the prior, the architecture itself was designed for univariate TSC: variate pooling is added post hoc on top of a frozen univariate encoder, and both variants require one encoder pass per variate, so inference cost grows linearly with the number of channels. Effective and efficient multivariate TSC will therefore likely require jointly extending the prior and the architecture.

Future Direction: Prior design and scale. TimEE is a small model trained on a single family of synthetic priors. The strong UCR results suggest that prior quality is a primary lever for performance in this paradigm — richer class generation mechanisms and larger model scale are therefore natural and promising avenues for future work.

AI USE STATEMENT

In this work, we used generative AI tools (such as Gemini and Claude) for text refinement, copyediting, and assisting with writing code for method development, evaluation, and plotting scripts. We did not any AI tools for generating core scientific claims, deriving the methodology, or producing the primary synthetic datasets. All AI-assisted work was thoroughly reviewed and verified by the authors. We take full responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work introduces a foundation model for time series classification trained entirely on synthetic data. As TIMEE relies solely on procedurally generated data (via a VARX-based prior) rather than real-world datasets, it mitigates common privacy, bias, and consent concerns associated with human-centric time series data (such as medical records or user telemetry). We do not foresee any immediate negative societal impacts from this methodology. However, as with all general-purpose classification models, practitioners deploying TIMEE in sensitive domains (e.g., healthcare diagnostics, finance) should ensure thorough evaluation of its predictions, fairness, and calibration on their specific target distributions to prevent unintended harms.

REPRODUCIBILITY STATEMENT

We have made significant efforts to ensure the reproducibility of our work. The complete source code for TIMEE, including the synthetic data generation pipeline (VARX prior) and the architecture, is available in our anonymous repository at `https://anonymous.4open.science/r/timee-research-2712`. Detailed hyperparameters for the model architecture and pre-training are provided in Appendix B.1 and Appendix D.1. Our evaluation relies on the open-source *aeon* library and the standard UCR/UEA archives, ensuring standardized data splits. Furthermore, all baseline implementation and tuning procedures are documented in Appendix E.

REFERENCES

Abdul Fatir Ansari, Lorenzo Stella, Ali Caner Turkmen, Xiyuan Zhang, Pedro Mercado, Huibin Shen, Oleksandr Shchur, Syama Sundar Rangapuram, Sebastian Pineda Arango, Shubham Kapoor, Jasper Zschiegner, Danielle C. Maddix, Hao Wang, Michael W. Mahoney, Kari Torkkola, Andrew Gordon Wilson, Michael Bohlke-Schneider, and Bernie Wang. Chronos: Learning the Language of Time Series. *Transactions on Machine Learning Research*, May 2024. ISSN 2835-8856. 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, October 2025. URL `http://arxiv.org/abs/2510.15821`. arXiv:2510.15821 [cs].

Andreas Auer, Daniel Klotz, Sebastinan Böck, and Sepp Hochreiter. Pre-trained forecasting models: Strong zero-shot feature extractors for time series classification, 2025a. URL `https://arxiv.org/abs/2510.26777`.

Andreas Auer, Patrick Podest, Daniel Klotz, Sebastian Böck, Günter Klambauer, and Sepp Hochreiter. Tirex: Zero-shot forecasting across long and short horizons with enhanced in-context learning, 2025b. URL `https://arxiv.org/abs/2505.23719`.

Fouad Bahrpeyma, Mark Roantree, Paolo Cappellari, Michael Scriney, and Andrew McCarren. A methodology for validating diversity in synthetic time series generation. *MethodsX*, 8:101459, 2021. ISSN 2215-0161. doi: 10.1016/j.mex.2021.101459.

Yanping Chen, Eamonn Keogh, Bing Hu, Nurjahan Begum, Anthony Bagnall, Abdullah Mueen, and Gustavo Batista. The UCR Time Series Classification Archive, July 2015.

Angus Dempster, François Petitjean, and Geoffrey I Webb. Rocket: exceptionally fast and accurate time series classification using random convolutional kernels: A. dempster et al. *Data Mining and Knowledge Discovery*, 34(5):1454–1495, 2020.

Angus Dempster, Daniel F Schmidt, and Geoffrey I Webb. Minirocket: A very fast (almost) deterministic transform for time series classification. In *Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining*, pp. 248–257, 2021.

Angus Dempster, Daniel F Schmidt, and Geoffrey I Webb. Hydra: competing convolutional kernels for fast and accurate time series classification: A. dempster et al. *Data Mining and Knowledge Discovery*, 37(5):1779–1805, 2023.

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, January 2026. URL http://arxiv.org/abs/2602.00620. arXiv:2602.00620 [cs].

Vasilii Feofanov, Songkang Wen, Marius Alonso, Romain Ilbert, Hongbo Guo, Malik Tiomoko, Lujia Pan, Jianfeng Zhang, and Ievgen Redko. Mantis: Lightweight Calibrated Foundation Model for User-Friendly Time Series Classification, February 2025. URL http://arxiv.org/abs/2502.15637. arXiv:2502.15637 [cs].

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, 2026. URL https://arxiv.org/abs/2602.17868.

Mononito Goswami, Konrad Szafer, Arjun Choudhry, Yifu Cai, Shuo Li, and Artur Dubrawski. MOMENT: a family of open time-series foundation models. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *ICML'24*, pp. 16115–16152, Vienna, Austria, July 2024. JMLR.org.

Léo Grinsztajn, Klemens Flöge, Oscar Key, Felix Birkel, Philipp Jund, Brendan Roof, Benjamin Jäger, Dominik Safaric, Simone Alessi, Adrian Hayler, Mihir Manium, Rosen Yu, Felix Jablonski, Shi Bin Hoo, Anurag Garg, Jake Robertson, Magnus Bühler, Vladyslav Moroshan, Lennart Purucker, Clara Cornu, Lilly Charlotte Wehrhahn, Alessandro Bonetto, Bernhard Schölkopf, Sauraj Gambhir, Noah Hollmann, and Frank Hutter. Tabpfn-2.5: Advancing the state of the art in tabular foundation models, 2026. URL https://arxiv.org/abs/2511.08667.

N. Hollmann, S. Müller, K. Eggensperger, and F. Hutter. TabPFN: A transformer that solves small tabular classification problems in a second. In *The Eleventh International Conference on Learning Representations (ICLR'23)*. ICLR, 2023.

N. Hollmann, S. Müller, L. Purucker, A. Krishnakumar, M. Körfer, Shi Bin Hoo, Robin Tibor Schirrmeister, and Frank Hutter. Accurate predictions on small data with a tabular foundation model. *Nature*, 637(8045):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. ISSN 1573-756X. doi: 10.1007/s10618-020-00710-y. URL http://dx.doi.org/10.1007/s10618-020-00710-y.

Keller Jordan, Yuchen Jin, Vlado Boza, You Jiacheng, Franz Cesista, Laker Newhouse, and Jeremy Bernstein. Muon: An optimizer for hidden layers in neural networks, 2024. *URL https://kellerjordan. github. io/posts/muon*, 6(3):4, 2024.

Krzysztof Kamycki, Tomasz Kapuscinski, and Mariusz Oszust. Data augmentation with suboptimal warping for time-series classification. *Sensors*, 20(1):98, 2019.

Chenguo Lin, Xumeng Wen, Wei Cao, Congrui Huang, Jiang Bian, Stephen Lin, and Zhirong Wu. Nutime: Numerically multi-scaled embedding for large-scale time-series pretraining, 2024. URL https://arxiv.org/abs/2310.07402.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization, 2019. URL https://arxiv.org/abs/1711.05101.

C. Lubba, S. Sethi, P. Knaute, S. Schultz, B. Fulcher, and N. Jones. catch22: Canonical time-series characteristics. pp. 1821–1852, 2019.

Helmut Lütkepohl. *New Introduction to Multiple Time Series Analysis*. Springer, Berlin, Heidelberg, 2005. ISBN 978-3-540-40172-8 978-3-540-27752-1. doi: 10.1007/978-3-540-27752-1. URL http://link.springer.com/10.1007/978-3-540-27752-1.

Matthew Middlehurst, Ali Ismail-Fawaz, Antoine Guillaume, Christopher Holder, David Guijo-Rubio, Guzal Bulatova, Leonidas Tsaprounis, Lukasz Mentel, Martin Walter, Patrick Schäfer, and Anthony Bagnall. aeon: a python toolkit for learning from time series. *Journal of Machine Learning Research*, 25(289):1–10, 2024. URL http://jmlr.org/papers/v25/23-1444.html.

S. Müller, N. Hollmann, S. Arango, J. Grabocka, and F. Hutter. Transformers can do Bayesian inference. In *The Tenth International Conference on Learning Representations (ICLR'22)*. ICLR, 2022.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. 12:2825–2830, 2011.

Foster J. Provost, Tom Fawcett, and Ron Kohavi. The case against accuracy estimation for comparing induction algorithms. In *Proceedings of the Fifteenth International Conference on Machine Learning*, ICML '98, pp. 445–453, San Francisco, CA, USA, 1998. Morgan Kaufmann Publishers Inc. ISBN 1558605568.

Chen Qian, Xiucai Ding, and Lexin Li. Structural classification of locally stationary time series based on second-order characteristics. *arXiv preprint arXiv:2507.04237*, 2025.

Jingang Qu, David Holzmüller, Gaël Varoquaux, and Marine Le Morvan. Tabicl: A tabular foundation model for in-context learning on large data, 2025. URL https://arxiv.org/abs/2502.05564.

Jingang Qu, David Holzmüller, Gaël Varoquaux, and Marine Le Morvan. Tabiclv2: A better, faster, scalable, and open tabular foundation model, 2026. URL https://arxiv.org/abs/2602.11139.

Hiroaki Sakoe and Seibi Chiba. Dynamic programming algorithm optimization for spoken word recognition. *IEEE transactions on acoustics, speech, and signal processing*, 26(1):43–49, 1978.

Zhenmei Shi, Jiefeng Chen, Kunyang Li, Jayaram Raghuram, Xi Wu, Yingyu Liang, and Somesh Jha. The trade-off between universality and label efficiency of representations from contrastive learning. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=rvsbw2YthH_.

Jianlin Su, Yu Lu, Shengfeng Pan, Bo Wen, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding, 2021.

Floris Takens. Detecting strange attractors in turbulence. In *Dynamical Systems and Turbulence, Warwick 1980: proceedings of a symposium held at the University of Warwick 1979/80*, pp. 366–381. Springer, 2006.

Shifeng Xie, Vasilii Feofanov, Marius Alonso, Ambroise Odonnat, Jianfeng Zhang, Themis Palpanas, and Ievgen Redko. CauKer: classification time series foundation models can be pre-trained on synthetic data only, August 2025. URL http://arxiv.org/abs/2508.02879. arXiv:2508.02879 [cs].

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, 2025. URL https://arxiv.org/abs/2511.19694.

Zhihan Yue, Yujing Wang, Juanyong Duan, Tianmeng Yang, Congrui Huang, Yunhai Tong, and Bixiong Xu. Ts2vec: Towards universal representation of time series. In *Proceedings of the AAAI conference on artificial intelligence*, volume 36, pp. 8980–8987, 2022.

A Related Work Details

A.1 Comparison to related methods for Time Series Classification

Table 1: Structural decomposition of compared methods into encoder F_θ and classifier G_ϕ. *New* G_ϕ indicates whether a task-specific classifier must be re-fitted for each new dataset. *Class-aware* indicates whether F_θ has access to the labeled support set at inference time. †Competitive performance requires tuning the intermediate layer used for feature extraction (Auer et al., 2025a)).

Method	**Encoder** F_θ	F_θ **objective**	F_θ **data**	**Classifier** G_ϕ	**New** G_ϕ **per dataset**	**Class-aware at inference**
Classical & Time Series Features Engineering Methods						
DTW-1NN	Dynamic warping distance	—	—	1-NN	—	×
Catch22	22 hand-crafted statistics	—	—	Linear / SVM	✓	×
Rocket / MiniRocket	Random conv. kernels; PPV & max pooling	—	—	Ridge regression	✓	×
Hydra	Competing random conv. kernels; per-group argmax counts	—	—	Ridge regression	✓	×
Per-dataset deep learning						
InceptionTime	Multi-scale conv. encoder	Cross-entropy (joint)	Target dataset	Softmax head (joint)	✓	×
TS2Vec	Hierarchical contrastive encoder	Contrastive (unsupervised)	Target dataset (unlabeled)	Linear / SVM	✓	×
Foundation models — TSC-focused						
NuTime	Transformer; multi-scaled window embedding	BYOL contrastive	>1M seq.	Linear / SVM	✓	×
MOMENT	Masked autoencoder transformer	Masked reconstruction	Large real-TS corpus	Linear / SVM†	✓	×
Mantis	ViT encoder	Contrastive	Large real-TS corpus	Linear / SVM†	✓	×
MantisV2	ViT encoder + test-time strategies	Contrastive	CauKer synthetic	Linear / SVM†	✓	×
Foundation models — forecasting, repurposed						
Chronos-2	Transformer encoder	Forecasting (quantile)	Large TS corpus	Linear / SVM†	✓	×
TiRex	Transformer encoder	Forecasting	Large TS corpus	Linear / SVM†	✓	×
End-to-end in-context learning						
TiCT	ResNet encoder	ICL cross-entropy (binary)	KernelSynth mixup	ICL head (bit-encoded labels)	×	✓
TimEE (ours)	Transformer encoder	ICL cross-entropy	VARX structured synthetic	ICL head	×	✓

A.2 Comparison TimEE vs TiCT (Yeh et al., 2025)

Concurrent with this work, TiCT proposes end-to-end ICL for TSC with synthetic pre-training — the paradigm most directly related to ours. TiCT pre-trains a ResNet-based encoder on binary synthetic tasks via KernelSynth mixup, uses bit-based label encoding for arbitrary class counts, and selects context via 64-nearest-neighbor retrieval at inference. TimEE departs from TiCT in three key respects: our class generation mechanism — structural breaks in a VARX process – produces multi-class tasks where class identity reflects differences in the underlying dynamics; our transformer architecture explicitly models both temporal within-series structure and cross-series ICL; and we use the full training set as context without a retrieval step. We evaluate against substantially stronger baselines on both UCR and UEA. That TiCT and TimEE independently converge on the same paradigm strengthens the case that end-to-end ICL with synthetic pre-training is a promising direction for TSC.

B Model Architecture

B.1 Model Hyperparameters

We report the model hyperparameters for TimEE in Table 2.

Table 2: Architecture hyperparameters of TimEE.

Hyperparameter	**Value**
Tokenization	
Patch size	16
Patch stride	16
Architecture	
Model dimension d_{model}	128
Temporal layers, L_t	5
Cross-series layers, L_c	5
ICL layers, L_{icl}	2
Attention heads	4
d_{kv}	32
$d_{ICL_{kv}}$	64
MLP hidden dimension	512
Decoder MLP hidden dimension	512
Dropout	0.1
Classification	
CLS tokens	4
Maximum classes	10

B.2 Attention Masking Strategy

Let $\mathcal{S} = \{1, \ldots, n\}$ and $\mathcal{Q} = \{n+1, \ldots, n+m\}$ denote the indices of training and query series respectively, with $N = n + m$. The binary attention mask $\mathbf{M} \in \{0,1\}^{N \times N}$, applied consistently across all cross-series attention layers and the in-context reasoning layers, is defined as:

$$\mathbf{M}_{ij} = \begin{cases} 1 & \text{if } i \in \mathcal{S} \text{ and } j \in \mathcal{S} \\ 1 & \text{if } i \in \mathcal{Q} \text{ and } j \in \mathcal{S} \cup \{i\} \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

where $\mathbf{M}_{ij} = 1$ indicates that series i is permitted to attend to series j, and masked positions are set to $-\infty$ in the pre-softmax attention logits. The resulting block structure is:

$$\mathbf{M} = \begin{array}{c} \\ \mathcal{S} \\ \mathcal{Q} \end{array} \begin{array}{c} \begin{array}{cc} \mathcal{S} & \mathcal{Q} \end{array} \\ \begin{pmatrix} \checkmark & \mathbf{0} \\ \checkmark & \mathbf{I} \end{pmatrix} \end{array} \tag{6}$$

Training series attend only to other training series; query series attend to all training series and themselves, but not to other query series. Consequently, no information flows from query series to training series at any layer, and each query is classified independently given the support set: the prediction for a query does not depend on which other queries are present in the context. Temporal attention layers and the compression layer operate within each series independently and require no masking.

B.3 Extension to Multivariate

Architecture. TimEE is designed and trained to operate on univariate series only. To handle multivariate inputs $\mathbf{X} \in \mathbb{R}^{N \times S \times M}$, we encode each of the M variates independently through the shared encoder, producing M sets of `CLS` token vectors per series. These are aggregated via *slot-wise attention pooling*: each of the K `CLS` slots has a learnable query that attends over the M variate embeddings via a softmax, allowing each slot to specialize on the most informative variates. The resulting pooled summary $\mathbf{r}_i \in \mathbb{R}^{K \cdot d}$ is then passed to the ICL phase.

Fine-tuning Details. Queries are zero-initialized so that pooling starts as a uniform average, providing a stable initialization from the pretrained univariate checkpoint. The encoder weights are frozen and only the pooling queries, ICL block, and decoder are updated during multivariate fine-tuning.

B.4 Ablation of Model Components.

We conduct the following ablation studies in the following training setup while we modify the architecture configuration by:

- scaling down the model configuration to $d_{kv} = 16$, $n_{heads} = 2$;
- fixing the sequence length at 512 with batch size 64;
- training on our synthetic data for 25K steps;
- evaluating on the train split of 86 UCR datasets (defined by: at least 100 samples, the least frequent classes must at least have 2 examples) with stratified 3-fold cross-validation.

Results reported are averaged over 3 seeds.

Contribution of Temporal and Cross-Series Attention. We ablate the contribution of cross-series (vertical) attention within the encoder by varying the allocation of encoder layers between temporal (L_t) and cross-series (L_c) attention, keeping the total fixed at $L_t + L_c = 8$ with $L_{\text{icl}} = 2$.

Table 3: Ablation of encoder layer allocation between temporal and cross-series attention.

Configuration	L_t	L_c	ROC AUC	Accuracy (%)
Temporal only	8	0	78.21 ± 0.39	70.70 ± 0.08
	6	2	79.96 ± 0.29	71.06 ± 0.10
Base config	4	4	80.32 ± 0.57	71.26 ± 0.09
	2	6	80.45 ± 0.36	71.20 ± 0.05
Cross-series only	0	8	79.65 ± 0.15	71.25 ± 0.02

Removing cross-series attention entirely ($L_c = 0$) causes a clear 2pp drop, confirming that encoder-level cross-series context provides complementary signal to the ICL phase and cannot be recovered from `CLS`-level comparison alone.

Removing temporal attention ($L_t = 0$) causes a smaller but still notable drop of approximately 1pp; local patch features provide a partial substitute for temporal structure, while cross-series comparison at the patch level is harder to replicate from compressed representations.

Performance is stable across all mixed configurations, indicating that the model is insensitive to the exact allocation as long as both components are present.

We note that all configurations retain a single compression attention layer (Section 3.3), so the $L_t = 0$ condition is not strictly temporal-attention-free.

Contribution of conditioning on Class Label. We ablate the two class embedding injection sites: patch-level conditioning (added to training patches before the encoder) and ICL-level conditioning (added to `CLS` tokens before the ICL block).

Without any class conditioning, accuracy collapses to 50%: the model has no access to label information during the forward pass and cannot associate support series with their classes. Patch-level conditioning is the dominant contributor—and in isolation sufficient: removing ICL-level conditioning while retaining patch-level conditioning leaves ROC AUC essentially unchanged ($80.32 \rightarrow 80.35$,

Table 4: Ablation of label conditioning sites.

Patch conditioning	ICL conditioning	ROC AUC	Accuracy (%)
✓	✓	80.32 ± 0.57	71.26 ± 0.09
✓	–	80.35 ± 0.13	71.16 ± 0.08
–	✓	78.55 ± 0.18	71.01 ± 0.09
–	–	22.27 ± 0.37	50.13 ± 0.43

within run-to-run variance), whereas retaining only ICL-level conditioning drops ROC AUC by $\sim$1.8pp ($80.32 \rightarrow 78.55$), because the encoder's cross-series attention operates on class-unaware patch representations and produces less discriminative `CLS` summaries.

Although ICL-level conditioning provides no measurable benefit over patch-level conditioning alone, we retain it in our final model: it was included in the large-scale pretraining run, and the negligible overhead does not justify re-training without it. Together, the two sites interact non-linearly: their individual contributions do not account for the full $\sim$50pp collapse when both are removed, confirming that the model requires at least one explicit class signal to function.

C Synthetic Data

C.1 Details of Sampling Procedure

We generate every synthetic dataset from a VARX-based prior (Lütkepohl, 2005) under one of two class-generation mechanisms (Section 3.2). For each dataset, one mechanism is selected uniformly at random. Under both mechanisms, the base VARX parameters (dimensions d and q, lag orders p and s, coefficient matrices $\mathbf{A}_k$ and $\mathbf{B}_j$, and the base exogenous signal) are sampled once per dataset and shared across all classes. Classes differ only in the controlled modification that defines the respective mechanism.

Under the *structural variation* mechanism, class identity is determined by class-specific edge dropout applied to the shared base dependency graph, i.e., each class drops a different subset of non-zero entries in $\mathbf{A}_k$ or $\mathbf{B}_j$. The full sampling procedure is given in Algorithm 1.

Under the *signal variation* mechanism, the dependency structure $\mathbf{A}_k, \mathbf{B}_j$ remains fixed across all classes. Class identity is instead determined by statistical properties of the exogenous input signal, namely the speed at which the shared base signal is warped. The full sampling procedure is given in Algorithm 2.

Sampling distributions. All parameters of our prior are sampled from fixed meta-distributions per dataset, so that each dataset follows a different data-generating process. Table 5 lists the distributions for the parameters of Algorithms 1 and 2. For univariate pretraining, we fix the endogenous dimension to $d = 1$. The remaining components of the prior are described below.

Innovation noise. The innovation term of the VARX process is Gaussian, $\boldsymbol{\epsilon}_t \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma})$, drawn independently at each time step. To obtain a valid covariance matrix, we construct $\boldsymbol{\Sigma} = \mathbf{M}\mathbf{M}^\top$, where the entries of $\mathbf{M} \in \mathbb{R}^{d\times d}$ are sampled i.i.d. from $\mathcal{U}(0, 1)$. This construction guarantees that $\boldsymbol{\Sigma}$ is symmetric positive semi-definite.

Exogenous signals. We build the exogenous inputs $\mathbf{x}_t$ from well-known time series patterns, following the classical decomposition of a time series into trend, seasonal, and irregular components. We largely follow Bahrpeyma et al. (2021) and construct each exogenous signal as a mixture of different trends, seasonal patterns, and irregularities.

Choice of sampling ranges. Beyond the distributions in Table 5, we impose only two hard constraints. First, the number of classes is capped at $C_{\max} = 10$, matching the output dimension of the model. Second, the maximum lag orders $p_{\max}$ and $s_{\max}$ are bounded, since the cost of unrolling a VARX process grows with the lag order. All remaining ranges are chosen deliberately broad.

Algorithm 1 Structural Variation Prior

Require: Number of classes C, number of samples per class n, max lag orders $p_{\max}$, $s_{\max}$, min/max endogenous dimension $d_{\min}$, $d_{\max}$, max exogenous dimension $q_{\max}$

Sample dimensions $d \sim \mathcal{U}\{d_{\min}, d_{\max}\}$, $q \sim \mathcal{U}\{1, q_{\max}\}$
Sample lag orders $p \sim \mathcal{U}\{1, p_{\max}\}$ and $s \sim \mathcal{U}\{0, s_{\max}\}$
Sample sparse base coefficient matrices $\mathbf{A}_k \in \mathbb{R}^{d\times d}$ for $k = 1, \ldots, p$
Sample $\mathbf{B}_j \in \mathbb{R}^{d\times q}$ for $j = 0, \ldots, s$
Sample exogenous signal $\mathbf{x}_t \in \mathbb{R}^q$ for $t = 1, \ldots, T$
for $c = 1, \ldots, C$ **do**
    Select a subset of non-zero weights in $\mathbf{A}_k$ or $\mathbf{B}_j$ to drop, yielding class-specific matrices $\mathbf{A}_k^{(c)}, \mathbf{B}_j^{(c)}$
    for $i = 1, \ldots, n$ **do**
        Unroll VARX using $\mathbf{A}_k^{(c)}, \mathbf{B}_j^{(c)}$, $\mathbf{x}_t$ with innovations $\epsilon \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma})$ to obtain $\mathbf{y}_1, \ldots, \mathbf{y}_T$
        Assign class label $y_i \leftarrow c$
    end for
end for
return $\{(\mathbf{y}_i^{(1:T)}, y_i)\}$

Algorithm 2 Signal Variation Prior

Require: Number of classes C, number of samples per class n, max lag orders $p_{\max}$, $s_{\max}$, signal variation range $[\omega_{\min}, \omega_{\max}]$, min/max dimension $d_{\min}$, $d_{\max}$

Sample dimensions $d \sim \mathcal{U}\{d_{\min}, d_{\max}\}$, $q \sim \mathcal{U}\{1, q_{\max}\}$
Sample lag orders $p \sim \mathcal{U}\{1, p_{\max}\}$ and $s \sim \mathcal{U}\{0, s_{\max}\}$
Sample sparse base coefficient matrices $\mathbf{A}_k \in \mathbb{R}^{d \times d}$ for $k = 1, \ldots, p$
Sample $\mathbf{B}_j \in \mathbb{R}^{d \times q}$ for $j = 0, \ldots, s$
Sample base exogenous signal $\tilde{\mathbf{x}}_t \in \mathbb{R}^q$ for $t = 1, \ldots, T$
Partition $[\omega_{\min}, \omega_{\max}]$ into C buckets $\{\mathcal{W}_c\}_{c=1}^{C}$
for $c = 1, \ldots, C$ **do**
    for $i = 1, \ldots, n$ **do**
        Sample signal variation $\omega_i \sim \mathcal{U}(\mathcal{W}_c)$
        Apply signal variation ω_i to base exogenous signal $\tilde{\mathbf{x}}_t$ to obtain $\mathbf{x}_t^{(i)}$
        Unroll VARX using $\mathbf{A}_k, \mathbf{B}_j, \mathbf{x}_t^{(i)}$ with innovations $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{\Sigma})$ to obtain $\mathbf{y}_1, \ldots, \mathbf{y}_T$
        Assign class label $y_i \leftarrow c$
    end for
end for
return $\{(\mathbf{y}_i^{(1:T)}, y_i)\}$

Table 5: Prior hyperparameters and their sampling distributions.

Parameter	Distribution	Description
$\mathbf{A}_k$	$\mathcal{U}(-1, 1)$	Endogenous (autoregressive) coefficients
$\mathbf{B}_j$	$\mathcal{U}(-3, 3)$	Exogenous coefficients
C	$\mathcal{U}(2, 10)$	Number of classes
$p_{\max}$	$\mathcal{U}(10, 20)$	Max endogenous lag order
$s_{\max}$	$\mathcal{U}(10, 20)$	Max exogenous lag order
$q_{\max}$	$\mathcal{U}(1, 10)$	Max exogenous input dimension
`support_query_split`	$\mathcal{U}(0.1, 0.9)$	Fraction of series used as support
`series_len`	$\mathcal{U}(16, 1024)$	Time series length

C.1.1 Relation to ARX

The VARX process described in Section 3.2 naturally subsumes the univariate setting as a special case. For $d = 1$, the system reduces to a scalar AutoRegressive process with Exogenous inputs (ARX):

$$y_t = \sum_{k=1}^{p} a_k y_{t-k} + \sum_{j=0}^{s} \mathbf{b}_j^\top \mathbf{x}_{t-j} + \varepsilon_t, \qquad \varepsilon_t \sim \mathcal{N}(0, \sigma^2)$$

where $a_k \in \mathbb{R}$ and $\mathbf{b}_j \in \mathbb{R}^q$ are the scalar and exogenous coefficient vectors respectively. All structural properties of our original VARX prior, the class-discriminative structural variations as well as the signal variations, carry over directly to this special case without modification. This ensures that the same DGP underlies both the univariate and multivariate synthetic data, and that results on univariate benchmarks such as UCR are fully consistent with the general VARX formulation. We note, however, that the limited transfer from multivariate synthetic pre-training data to real-world multivariate classification tasks remains a limitation of TimEE and identify it as an active avenue for future research.

C.1.2 Multivariate Synthetic Data

Our VARX-based prior natively supports multivariate data generation for arbitrary $d > 1$, and using the extension discussed in Section 4.3 TimEE can in principle be applied to multivariate time series classification tasks directly. However, we observe that in synthetically generated VARX datasets, the information necessary for class discrimination is typically well-concentrated in individual variates

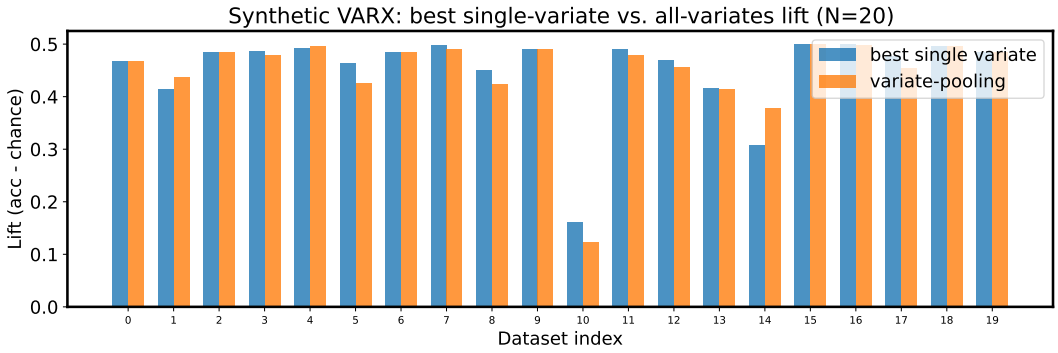


Figure 9: Dataset generated using structural variation mechanism.

already. As a consequence, averaging predictions across all d variates rarely improves over the best single-variate prediction, and often even degrades performance due to the inclusion of uninformative channels. This behavior is consistent with observations in the forecasting literature (Ansari et al., 2025) and finds motivation in Takens' embedding theorem (Takens, 2006), which states that under mild conditions, an observation function of a dynamical system suffices to reconstruct the full state space of that system. While Takens' theorem applies strictly to deterministic smooth dynamical systems, the underlying intuition is suggestive in our setting: in the VARX process, each observed $y_t^{(i)}$ is a linear combination of the full history of all endogenous and exogenous variables, meaning inter-variable information naturally propagates into individual channels through the autoregressive dynamics. We highlight this through an empirical study on 20 synthetic multivariate datasets generated by the TIMEE prior, shown in Figure 9. For each dataset, we compare the predictive performance of the single best univariate prediction - that is, the best performance achieved by any individual variate, against predictions obtained via *variate pooling* as described in Section 4.3. In nearly all datasets, variate pooling fails to improve over the single best variate prediction, and in several cases actively degrades performance. This confirms that under our VARX prior, class-discriminative information is already captured by individual variates, and aggregating across channels provides little to no benefit.

C.2 AUGMENTATIONS

Smoothing. We apply a Gaussian filter with bandwidth $\sigma \sim \mathcal{U}(\sigma_{\min}, \sigma_{\max})$, independently sampled per time series.

Spike Injection. The number of spikes $n_s \sim \mathcal{U}\{n_{\min}, n_{\max}\}$ is sampled per series. Spike positions are drawn uniformly from $\{1, \ldots, T\}$, and each spike amplitude is sampled independently as $s \sim \mathcal{U}(s_{\min}, s_{\max})$, giving $\mathbf{y}'_t = \mathbf{y}_t + s \cdot \mathbf{1}$ for a randomly drawn position t.

Nonlinear Transformation. A nonlinearity is sampled uniformly from $\{\tanh, \sin, \text{ReLU}, |\cdot|\}$ and applied pointwise as $\mathbf{y}'_t = g(\mathbf{y}_t)$.

C.3 MIXUP

To further diversify our synthetic pre-training datasets, we follow the mixup scheme used to augment forecasting data (Ansari et al., 2025; Auer et al., 2025b). We adapt this mechanism to the classification setting by forming convex combinations of samples drawn from within the same dataset, thereby preserving label structure while increasing intra-class diversity. Concretely, given k time series samples $\tilde{\mathbf{y}}^{(1)}, \ldots, \tilde{\mathbf{y}}^{(k)}$ drawn from arbitrary classes, the mixed sample is:

$$\mathbf{y}^{\text{mix}} = \sum_{i=1}^{k} \lambda_i \tilde{\mathbf{y}}^{(i)}, \qquad \lambda \sim \text{Dir}(\alpha)$$

where $\sim \mathcal{U}\{1, K_{\max}\}$ and mixing weights λ are drawn from a Dirichlet distribution. The full procedure is described in Algorithm 3.

Algorithm 3 Time Series Mixup

Require: Dataset $D := \{\mathbf{y}^{(1:t)}, y\}$, Target Length T
1: $D_{\text{out}} \leftarrow \emptyset$
2: Sample subset of time series $\{(\mathbf{y}_i^{(1:t)}, y_i)\} \subset D$
3: **Preprocessing:**
4: $\quad \mathbf{y_i}^{(1:t)} \leftarrow \text{Normalize}(\mathbf{y_i}^{(1:t)})$
5: $\quad \mathbf{y_i}^{(1:T)} \leftarrow \text{Resize}(\mathbf{y_i}^{(1:t)}, T)$ ▷ Interpolate or truncate
6: **Mixture Generation:**
7: $\quad$ Select k samples $\{\mathbf{y}_j^{(1:T)}\}_{j=1}^k$ with labels $\{y_j\}_{j=1}^k$
8: $\quad$ Sample weights $\lambda \sim \text{Dirichlet}(\alpha)$
9: $\quad \mathbf{y}_{mix}^{(1:T)} \leftarrow \sum_{j=1}^k (\lambda_j \cdot \mathbf{y}_j^{(1:T)}) + \mathcal{N}(\mathbf{0}, \mathbf{\Sigma})$
10: **Pseudo-Label Assignment:**
11: $\quad$ Assign y_{mix} based on λ and $\{y_j\}$ using threshold
12: $\quad$ Append $(\mathbf{y}_{mix}^{(1:T)}, y_{mix})$ to D_{out}
13: **return** D_{out}

C.4 Examples of synthetic data

We present example datasets generated from our synthetic prior, for different class generation mechanisms. Figure 10 present samples generated by structural variation. Figures 11 presents a dataset generated by signal level variation. Further we showcase the impact of the mixup scheme described in Section C.3 on a generated synthetic dataset, where Figure 12a shows the originally sampled dataset before applying mixup while Figure 12b shows the same dataset after applying mixup.

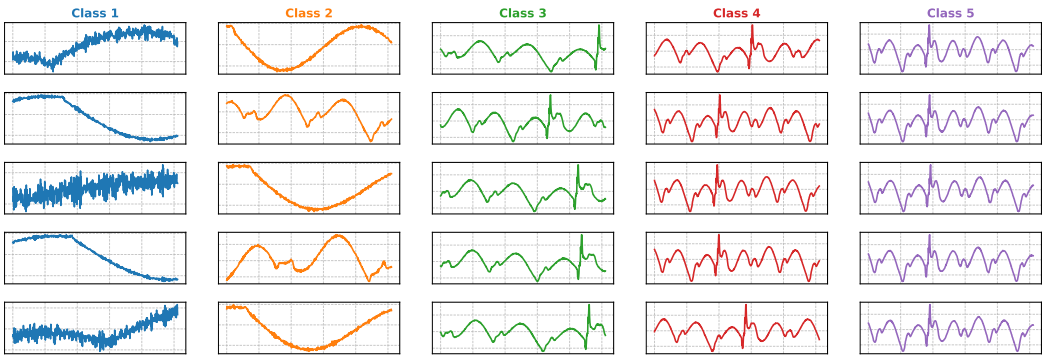


Figure 10: Dataset generated using structural variation mechanism.

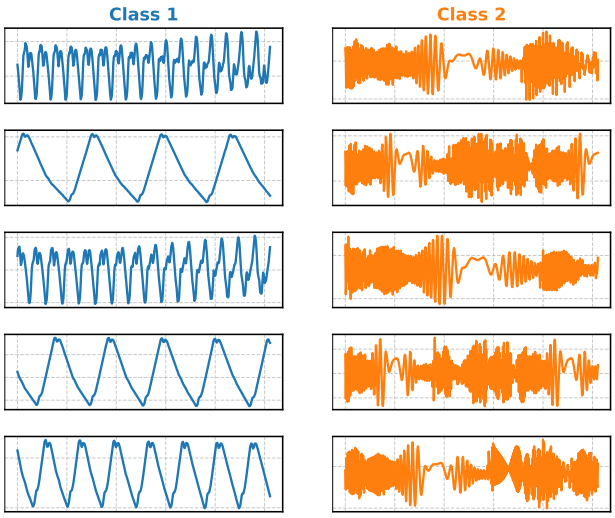
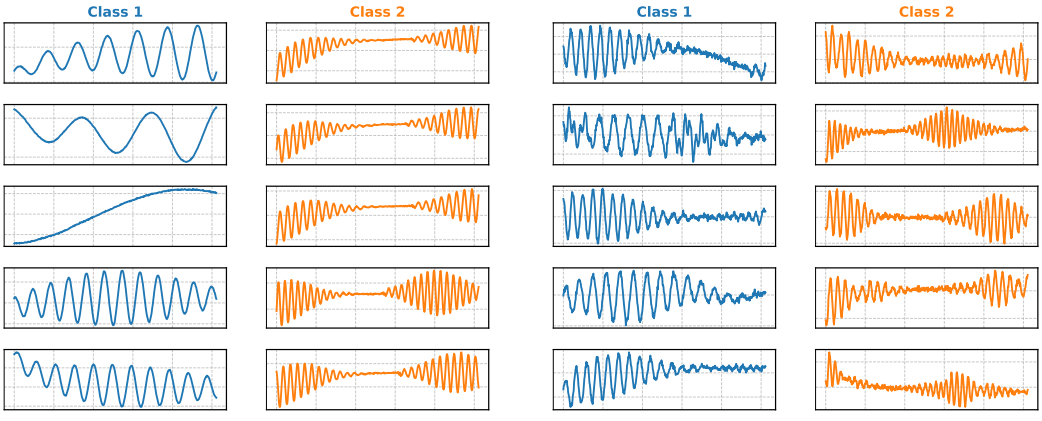


Figure 11: Dataset generated using signal variation mechanism.



(a) Before mixup.

(b) After mixup.

Figure 12: Dataset generated using signal variation. (a) before applying the mixup scheme; (b) after applying the mixture scheme.

C.5 Ablation on Synthetic Data Generation

Setup. We ablate the class-identity mechanisms, mixup, and the time-series-level augmentations (Section 3.2) in the reduced training setup described in Appendix B.4. Table 6 pretrains on each mechanism alone and on the 50/50 mixture used in our final prior, each with and without mixup, with all augmentations enabled. Table 7 fixes the mechanism to signal variation with mixup and enables one augmentation at a time. We report mean $\pm$ std over three seeds.

Table 6: Ablation of prior components (class-generation mechanism and mixup).

Class mechanism	Mixup	ROC AUC	Accuracy (%)
Structural variation	–	88.35 ± 0.33	70.86 ± 0.52
Structural variation	✓	91.82 ± 0.18	76.59 ± 0.43
Signal variation	–	90.46 ± 0.17	75.47 ± 0.33
Signal variation	✓	91.75 ± 0.13	77.08 ± 0.15
Both (50/50)	–	90.76 ± 0.03	75.27 ± 0.26
Both (50/50)	✓	$\mathbf{91.92 \pm 0.25}$	$\mathbf{77.30 \pm 0.13}$

Table 7: Ablation of time-series-level augmentations (signal variation, with mixup).

Augmentations	ROC AUC	Accuracy (%)
All (Smoothing, Spikes, Nonlinear)	91.75 ± 0.13	77.08 ± 0.15
Smoothing only	91.66 ± 0.16	76.91 ± 0.20
Spikes only	91.52 ± 0.16	76.55 ± 0.11
Nonlinear only	91.68 ± 0.41	77.33 ± 0.32

Contribution of Class Mechanisms. Without mixup, signal variation alone clearly outperforms structural variation alone (+4.6 pp accuracy, +2.1 ROC AUC). We attribute this to edge dropout by itself producing classes that differ in subtle ways and exhibit limited intra-class diversity. With mixup, both mechanisms reach comparable performance, with overlapping ranges across seeds. Each mechanism alone already yields a competitive prior, and combining both produces the strongest configuration overall. Our final prior therefore combines both mechanisms, which diversifies the pretraining distribution.

Contribution of Mixup. Mixup improves performance under both mechanisms, but its effect is stronger for structural variation (+5.7 pp accuracy, +3.5 ROC AUC). Mixup compensates for the limited intra-class diversity of structural variation by increasing intra-class diversity and producing less sharply separated class boundaries.

Contribution of Augmentations. We ablate the three per-time-series augmentations in Table 7. No single augmentation consistently improves on the combination, so we keep all three by default.

D PRETRAINING AND INFERENCE

D.1 PRETRAINING CONFIGURATIONS

Table 8 summarizes the full set of hyperparameters used during pretraining.

Table 8: Pretraining hyperparameters.

Hyperparameter	**Value**
Training steps	30,000
Warmup steps	1,500 (5% of total)
Learning rate	3×10^{-4}
LR schedule	Cosine decay with linear warmup
Optimizer (matrix weights)	Muon (Jordan et al., 2024)
Optimizer (non-matrix weights)	AdamW (Loshchilov & Hutter, 2019)
Batch size	240
Mixed precision	bf16 AMP
Number of classes C	Uniform $\sim \mathcal{U}\{2, 10\}$ per task
Support-query split	Uniform $\sim \mathcal{U}[0.1, 0.9]$ per task
Series length	Uniform $\sim \mathcal{U}[16, 1024]$, shared across context
Hardware	Single NVIDIA H200 GPU
Training time	≈ 40 hours

D.2 TEST-TIME AUGMENTATION: PREPROCESSING VIEWS

TIMEE averages predictions from a small set of views of the input dataset, each obtained by applying a deterministic preprocessing transformation to all series in the context (both support and queries) before tokenization. We use two elementary transformations:

Length interpolation. Series shorter than a target length $T^\star$ are linearly interpolated to length $T^\star$; longer series are left unchanged. This lifts very short series into the length regime seen during pretraining. We use $T^\star \in \{256, 512\}$.

First-order differencing. Each series $\mathbf{y}$ is replaced by $\Delta\mathbf{y} = (y_2 - y_1, \ldots, y_T - y_{T-1})$, removing level and slow trend and emphasising local dynamics. The differenced series then passes through the standard normalization in Section 3.3.

A view is defined by a choice of differencing (on/off) and target length, giving the four default views $\{\text{raw},\ \text{differenced}\} \times \{256,\ 512\}$. Each view is processed by an independent forward pass, and the final prediction is the mean of the four predicted class distributions.

E Implementation of Baselines

E.1 Statistical Methods

MiniRocket (Dempster et al., 2021): We apply MiniRocket feature extraction followed by a `StandardScaler` and `RidgeClassifierCV`, using the `aeon` implementation.

Hydra (Dempster et al., 2023): We use the `aeon` implementation with default hyperparameters, which likewise classifies with `RidgeClassifierCV`.

Scoring of ridge-based classifiers. `RidgeClassifierCV` regresses onto ± 1 class-indicator targets and predicts via argmax; it exposes no `predict_proba`, and the `aeon` wrappers return one-hot vectors in its place. Computing ROC AUC or log loss from these would reduce both metrics to accuracy. For MiniRocket and Hydra we therefore use the `decision_function` outputs, mapped to the probability simplex via softmax. This leaves the argmax, and hence accuracy, unchanged, and ROC AUC is invariant to the choice of monotone mapping. Log loss is not: the scale of ridge scores is set by the regularisation strength rather than by log-odds, so their log-loss values should be read as indicative rather than as a calibrated estimate.

Catch22 (Lubba et al., 2019): We extract 22 canonical time-series summary statistics via `pycatch22` and classify with a `RandomForestClassifier` (`n_estimators=300`).

DTW-1NN (Sakoe & Chiba, 1978): 1-nearest-neighbour classifier with Dynamic Time Warping distance, using the `aeon` implementation; as it produces no scores, it is excluded from ROC AUC and log-loss comparisons.

E.2 Deep Learning Methods

TS2Vec (Yue et al., 2022): A contrastive self-supervised encoder trained independently per dataset with output dimension 320. We extract full-series embeddings and classify with an SVM whose regularisation parameter C is selected via 5-fold cross-validation on the training set.

InceptionTime (Ismail Fawaz et al., 2020): An ensemble of 5 InceptionTime networks trained for 1500 epochs per dataset, using the `aeon` implementation.

E.3 Foundation Models

For all foundation models we use a `RandomForestClassifier` from `scikit-learn` (Pedregosa et al., 2011) with `n_estimators=300` consistent with Feofanov et al. (2026). All remaining hyperparameters take their default values. Where a tuned variant is reported, we select the encoder layer whose embeddings yield the highest mean ROCAUC across the evaluation suite and apply it uniformly across all datasets.

E.3.1 TSC Foundation Models

Mantis-V2 (Feofanov et al., 2026): We use the official implementation. [8] As Mantis-V2 requires input sequences of length divisible by `num_patches`, we interpolate to next higher num patches. For the tuned version, embeddings from the 3rd layer yielded the strongest downstream performance. We also interpolate each time series to a minimum length of 512.

Moment (Goswami et al., 2024): We use the *large* size model for all experiments with Moment. We follow all defaults from the official implementation. [9] Using later encoder layers performs comparably; we use the 13th layer for the univariate setting and the 11th for the multivariate setting.

NuTime (Lin et al., 2024): A window-based ViT pretrained via BYOL on UCR datasets. We extract the frozen CLS-token embedding and classify with the above classifier. Input series are resampled to length 512. For the tuned variant, embeddings from the 6th layer yielded the strongest downstream performance.

[8] `https://github.com/vfeofanov/mantis`

[9] `https://github.com/moment-timeseries-foundation-model/moment`

TiCT (Yeh et al., 2025): A 47M-parameter ResNet pretrained for in-context time series classification. For each test sample, we retrieve its 64 nearest neighbours from the training set as context and perform a single forward pass; no separate classifier is required.

E.3.2 Forecasting Models

For both forecasting models, we extract patch-level embeddings per time series, apply mean pooling over patches to obtain a fixed-size representation, and classify using the above mentioned classifier.

Chronos-2 (Ansari et al., 2025): We use the *base* model throughout. By default, embeddings are extracted from the final encoder layer. For the tuned variant, we take embeddings from the 5th layer, which yielded the strongest downstream performance.

TiRex (Auer et al., 2025b): We apply the same extraction and classification strategy. For the tuned variant, we likewise use the 6th layer, following the same selection procedure.

E.4 Multivariate Setting

For models where we implement the multivariate adaptation ourselves, we reuse the best layer settings identified in the univariate search. Per-channel embeddings are concatenated before classification; mean pooling consistently degraded performance across models. We use a `RandomForestClassifier` throughout, as SVMs struggled with the extended embedding dimension produced by concatenation. **TiCT** classifies each channel independently and averages the per-channel softmax probabilities. DTW-1NN, MiniRocket, Hydra, and InceptionTime use their native multivariate implementations via `aeon` without modification. For TiRex, reusing the univariate tuned settings did not transfer to the multivariate setting and the default outperforms the tuned version.

F Experiment Results

F.1 Inference Time vs. Performance on Datasets

To evaluate the trade-off between the inference efficiency and the predictive performance, we measure the inference time of each method.

For GPU baselines, we evaluate all GPU-accelerated candidates on a NVIDIA H100 GPU; for CPU baselines, we evaluate them on AMD EPYC 9655.

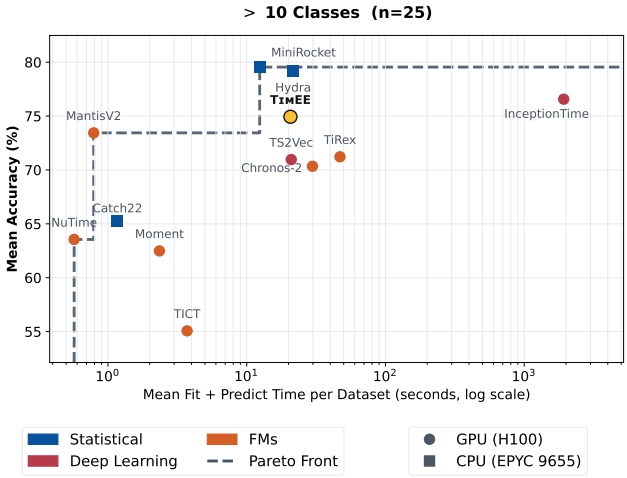


Figure 13: Pareto front of inference speed vs. accuracy on UCR datasets with more than 10 classes. OneVsRest many-class strategy of TimEE imposes significant inference overhead. This remains as a limitation of TimEE.

F.2 Critical Difference Diagram

In the following critical difference diagrams, each method enters with a single entry (the tuned variant where one exists), whereas Figure 1 ranks tuned and untuned variants jointly; mean ranks are therefore lower here and the relative order of closely ranked methods may differ slightly.

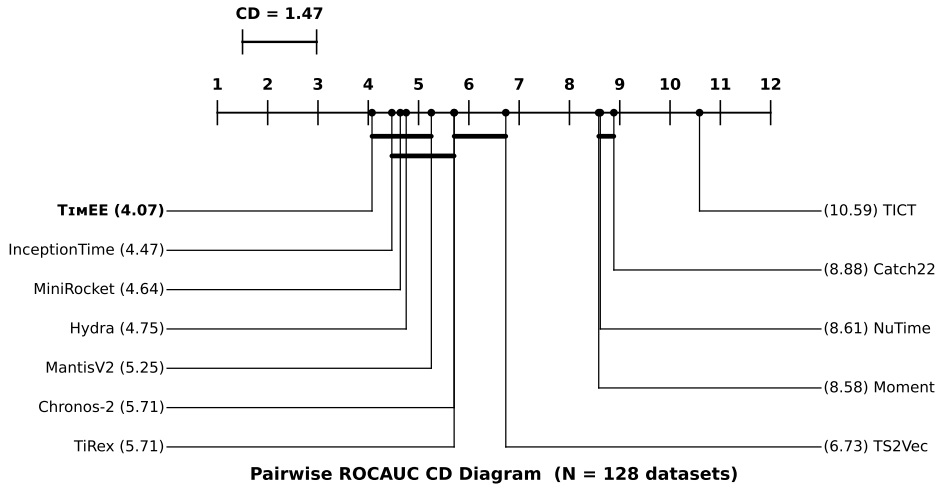


Figure 14: Critical difference diagram over ROC AUC across 128 UCR datasets (Friedman test with Nemenyi post-hoc test, $\alpha = 0.05$). Methods connected by a horizontal bar are not significantly different.

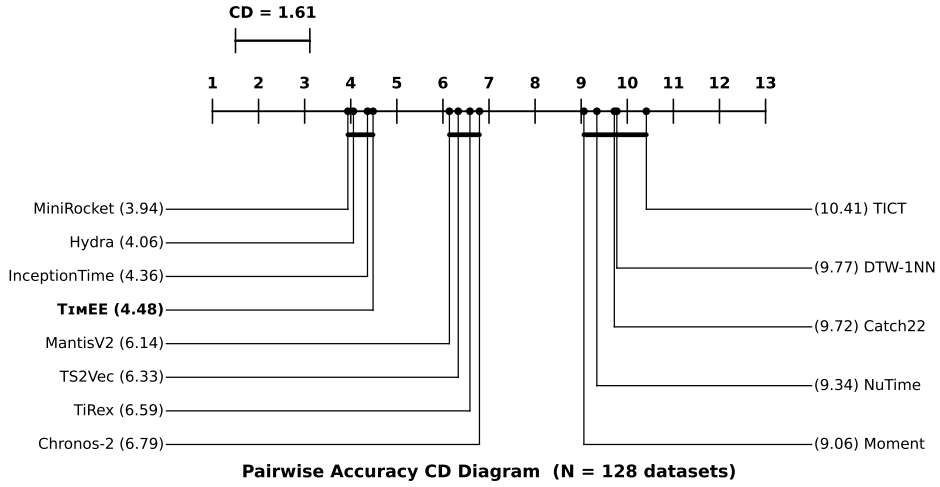


Figure 15: Critical difference diagram over accuracy across 128 UCR datasets (Friedman test with Nemenyi post-hoc test, $\alpha = 0.05$). Methods connected by a horizontal bar are not significantly different.

F.3 PER-DATASET RESULTS

Table 9: Per-dataset ROC AUC on the UCR archive (128 datasets; higher is better). Best per dataset in bold. Mean and mean rank are computed over the 128 datasets with results for all models. Models with a tuned variant also show their untuned variant; ranks are computed over all columns jointly.

Dataset	Catch22	Hydra	MiniRocket	InceptionTime	TS2Vec	Chronos-2	Chronos-2 (untuned)	MantisV2	MantisV2 (untuned)	Moment	Moment (untuned)	NuTime	NuTime (untuned)	TiCT	TiRex	TiRex (untuned)	TimeEE
ACSF1	.986	.992	**.996**	**.996**	.989	.988	.975	.977	.973	.976	.970	.972	.965	.777	.990	.980	.983
Adiac	.985	.985	.979	.992	.981	**.994**	.986	**.994**	.989	.988	.989	.988	.988	.922	.992	.990	.992
AllGestureWiimoteX	.886	.925	.941	**.962**	.949	.912	.855	.945	.903	.902	.893	.905	.907	.876	.922	.903	.947
AllGestureWiimoteY	.901	.944	.957	**.976**	.946	.922	.913	.947	.861	.924	.922	.883	.880	.874	.944	.925	.960
AllGestureWiimoteZ	.899	.935	.952	**.972**	.954	.934	.885	.956	.905	.911	.917	.911	.911	.854	.935	.910	.954
ArrowHead	.892	.948	**.976**	**.976**	.937	.936	.930	.940	.936	.945	.932	.912	.903	.895	.899	.964	.975
Beef	.801	.986	.950	.942	.889	.928	.885	.915	.834	.924	.918	.934	.942	.858	.969	.992	**.997**
BeetleFly	.850	.980	**1.000**	**1.000**	.990	.970	.965	**1.000**	.950	**1.000**	**1.000**	.950	.950	.880	.980	**1.000**	**1.000**
BirdChicken	.980	.970	.900	**1.000**	.800	.970	.970	.990	.960	.980	.940	**1.000**	**1.000**	.990	.940	.835	.960
BME	.983	**1.000**	**1.000**	**1.000**	**1.000**	.999	**1.000**	**1.000**	.964	.991	.987	.957	.960	.848	**1.000**	**1.000**	**1.000**
Car	.904	.970	**.971**	.955	.909	.934	.917	.930	.941	.911	.910	.932	.918	.836	.939	.940	.956
CBF	.997	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	.995	.996	.999	.999	.992	**1.000**	.999	**1.000**
Chinatown	.987	.994	.994	**.996**	.989	.992	**.996**	.989	.500	.995	.995	.982	.979	.977	.995	.995	.995
ChlorineConcentration	.677	.887	.876	**.947**	.919	.851	.832	.831	.756	.832	.828	.782	.789	.711	.854	.874	.803
CinCECGTorso	.949	**1.000**	.971	.971	.950	.971	**1.000**	.959	.908	.900	.867	.945	.947	.862	**1.000**	**1.000**	**1.000**
Coffee	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	.995	**1.000**	**1.000**	**1.000**	.985	**1.000**	.995	.995	.995	**1.000**	**1.000**	**1.000**
Computers	.806	.769	.812	**.880**	.744	.828	.824	.772	.794	.733	.722	.820	.832	.744	.808	.805	.722
CricketX	.906	.968	.974	**.984**	.976	.960	.912	.977	.969	.954	.950	.946	.942	.894	.952	.953	.981
CricketY	.928	.978	.981	**.989**	.971	.966	.939	.978	.970	.958	.960	.959	.958	.905	.965	.967	.988
CricketZ	.914	.970	.975	**.988**	.973	.958	.925	.981	.976	.955	.956	.946	.950	.917	.957	.955	.986
Crop	.961	.966	.973	.981	.980	.977	.972	.979	.975	.972	.972	.970	.969	.942	.980	.977	**.984**
DiatomSizeReduction	.995	.999	.972	.997	**1.000**	.999	.998	.986	.984	.995	.997	.993	.992	.969	.988	.991	**1.000**
DistalPhalanxOutlineAgeGroup	.865	.903	.890	.858	.881	.904	.885	**.906**	.880	.881	.879	.877	.871	.856	.899	.897	.886
DistalPhalanxOutlineCorrect	.871	.861	.876	.851	.787	**.878**	.876	.869	.845	.877	.874	.842	.838	.825	.864	.866	.876
DistalPhalanxTW	.845	**.896**	.878	.854	.866	.874	.831	.873	.885	.878	.862	.858	.850	.818	.875	.872	.891
DodgerLoopDay	.829	.846	.899	.852	.862	.876	.879	.863	.796	.803	.808	.804	.819	.850	.859	.905	**.923**
DodgerLoopGame	.765	.915	.939	.917	**.980**	.941	.942	.886	.632	.880	.866	.848	.834	.916	.832	.921	.972
DodgerLoopWeekend	**.997**	.981	.984	.992	.991	**.997**	.989	.991	.979	.984	.988	.983	.985	.991	.988	.992	.991
Earthquakes	.692	.719	.692	.662	.544	.673	.639	.613	.595	.604	.635	.699	.709	.637	.640	.719	**.730**
ECG200	.908	.953	.963	**.967**	.957	.927	.903	.928	.893	.946	.940	.905	.900	.882	.923	.934	.957
ECG5000	.911	.905	.886	.927	.936	.929	.942	.916	.907	.936	.927	.911	.894	.884	**.953**	.950	.952
ECGFiveDays	.956	**1.000**	**1.000**	**1.000**	**1.000**	.992	.996	.959	.984	.954	.967	.914	.888	.921	.996	**1.000**	**1.000**
ElectricDevices	.911	.905	.894	.894	.916	**.918**	.906	.912	.881	.893	.893	.898	.900	.840	.902	.902	.897
EOGHorizontalSignal	.905	.908	.895	.930	.903	.909	.889	**.937**	.930	.886	.883	.865	.864	.805	.915	.899	.929
EOGVerticalSignal	.870	.832	.840	.879	.862	.837	.871	.857	.842	.816	.831	.778	.765	.787	.839	**.907**	.891
EthanolLevel	.617	.798	.811	**.958**	.695	.684	.686	.614	.539	.638	.620	.630	.631	.597	.617	.852	.878
FaceAll	.966	.998	.997	.995	.994	.980	.988	.963	.954	.972	.972	.949	.948	.966	.992	.997	**.999**
FaceFour	.942	.998	**1.000**	.992	.972	.982	.978	.993	.979	.941	.964	.985	.989	.940	.993	**1.000**	**1.000**
FacesUCR	.949	.996	.998	**.999**	.995	.974	.976	.982	.961	.958	.957	.958	.955	.931	.974	.989	.995
FiftyWords	.934	.973	.982	**.991**	.958	.973	.937	.971	.959	.972	.969	.953	.938	.905	.965	.964	.985
Fish	.965	.999	.998	**1.000**	.992	.988	.982	.991	.986	.982	.989	.997	.995	.944	.994	.991	.996
FordA	.967	**.992**	.988	**.992**	.983	.977	.884	.981	.967	.949	.962	.961	.962	.958	.991	.986	.987
FordB	.845	.908	.893	**.937**	.875	.872	.737	.881	.862	.815	.846	.836	.842	.807	.924	.913	.901
FreezerRegularTrain	**1.000**	**1.000**	**1.000**	**1.000**	.998	.999	.992	**1.000**	.980	.968	.964	.999	.999	.971	.989	**1.000**	**1.000**
FreezerSmallTrain	.995	.977	**.996**	.900	.956	.973	.923	.987	.891	.861	.855	.995	.994	.851	.918	.983	.991
Fungi	.997	**1.000**	**1.000**	**1.000**	.008	.996	.993	.997	.990	.997	.999	.974	.963	.988	.992	.997	**1.000**
GestureMidAirD1	.981	**.985**	.975	.958	.920	.981	.980	.974	.945	.967	.972	.947	.945	.958	.978	.975	.980
GestureMidAirD2	.978	.980	.970	.938	.880	.983	.982	.979	.969	.982	.980	.955	.952	.946	**.984**	.981	.982
GestureMidAirD3	.928	.910	.895	.807	.740	.955	**.959**	.938	.890	.917	.908	.873	.855	.827	.935	.951	.938
GesturePebbleZ1	.968	.985	.989	.987	.949	.989	.964	**.990**	.971	.981	.970	.958	.961	.962	.989	.980	.979
GesturePebbleZ2	.958	.986	.984	.970	.965	.990	.982	**.993**	.990	.981	.975	.970	.971	.957	.988	.992	.970
GunPoint	.994	**1.000**	**1.000**	**1.000**	.998	**1.000**	**1.000**	.998	.997	.997	.997	.994	.994	.985	.991	.999	.996
GunPointAgeSpan	.995	**1.000**	**1.000**	**1.000**	.999	**1.000**	.999	.999	**1.000**	.992	.991	.997	.997	.999	.998	.997	**1.000**
GunPointMaleVersusFemale	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	.999	**1.000**	**1.000**	.999	.999	.998	.995	**1.000**	**1.000**	**1.000**	**1.000**
GunPointOldVersusYoung	.997	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	.991	**1.000**	**1.000**	.995	.993	**1.000**	**1.000**	.996	**1.000**	.984	**1.000**
Ham	.672	.805	.812	**.828**	.770	.727	.783	.733	.714	.792	.771	.807	.825	.718	.695	.825	.780
HandOutlines	.918	.973	.963	**.983**	.948	.953	.953	.952	.881	.942	.945	.939	.937	.929	.945	.947	.966
Haptics	.746	.808	.815	**.821**	.792	.807	.782	.806	.784	.772	.776	.773	.775	.733	.797	.795	.817
Herring	.511	**.797**	.768	.734	.641	.740	.611	.700	.766	.659	.643	.656	.642	.654	.720	.621	.526
HouseTwenty	.983	.988	.990	.972	.957	.982	.943	**.992**	.980	.970	.975	.954	.956	.989	.991	.988	.988
InlineSkate	.808	.810	.817	.841	.781	.804	.705	.767	.821	.712	.709	.735	.742	.738	**.843**	.738	.802
InsectEPGRegularTrain	.997	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	.956	**1.000**	**1.000**	.980	.968	**1.000**	**1.000**	.983	**1.000**	.997	.998
InsectEPGSmallTrain	.959	**1.000**	**1.000**	**1.000**	**1.000**	.988	.930	**1.000**	**1.000**	.972	.952	**1.000**	**1.000**	.899	.998	.960	.992
InsectWingbeatSound	.930	.946	.946	.938	.949	.947	.953	.942	.925	.946	.945	.900	.891	.866	.953	**.959**	**.959**
ItalyPowerDemand	.959	.991	.993	.990	.985	.991	.993	.982	.500	.987	.990	.950	.947	.934	.993	.992	**.994**
LargeKitchenAppliances	.953	.970	.968	**.979**	.951	.972	.896	.925	.856	.899	.900	.881	.891	.853	.932	.870	.867
Lightning2	.869	.824	.804	.905	**.937**	.801	.824	.847	.811	.861	.845	.772	.778	.789	.799	.815	.898
Lightning7	.935	.977	.973	.967	.976	.972	.955	.966	.933	.955	.952	.952	.956	.909	.971	.953	**.979**
Mallat	.997	.999	.998	.999	.989	.998	.995	.997	.997	.997	.998	.992	.992	.871	**1.000**	**1.000**	.999
Meat	.974	.991	**1.000**	.997	.998	**1.000**	.994	.994	**1.000**	.998	**1.000**	.996	.988	.805	.992	.994	**1.000**
MedicalImages	.964	.960	.969	.972	**.974**	.962	.952	.968	.947	.960	.963	.955	.952	.943	.965	.972	.972
MelbournePedestrian	.975	.995	.998	**.999**	.998	.998	.992	.998	.500	.987	.985	.996	.996	.972	.998	.993	.997
MiddlePhalanxOutlineAgeGroup	.650	.623	.614	.599	.649	.645	.637	.633	.645	.622	.627	.657	**.666**	.633	.648	.649	.642
MiddlePhalanxOutlineCorrect	.854	.919	.921	.905	.901	**.939**	.926	.931	.907	.921	.921	.867	.863	.842	.921	.918	.928
MiddlePhalanxTW	.753	.774	.786	.759	.768	.783	.772	.776	.779	.789	**.790**	.738	.728	.741	.759	.763	.762

continued on next page

(continued)

Dataset	Catch22	Hydra	MiniRocket	InceptionTime	TS2Vec	Chronos-2	Chronos-2 (untuned)	MantisV2	MantisV2 (untuned)	Moment	Moment (untuned)	NuTime	NuTime (untuned)	TICT	TiRex	TiRex (untuned)	TimEE
MixedShapesRegularTrain	.989	**.999**	.998	.998	.989	.995	.993	.996	.995	.983	.980	.992	.992	.986	.998	.996	**.999**
MixedShapesSmallTrain	.975	**.997**	.994	.989	.978	.989	.985	.990	.990	.962	.960	.986	.986	.948	.994	.990	**.997**
MoteStrain	.946	.989	.972	.955	.933	.981	.970	.987	.937	.971	.967	**.991**	.988	.956	.963	.979	.973
NonInvasiveFetalECGThorax1	.996	**.999**	**.999**	**.999**	.998	.994	.996	.996	.993	.997	.997	.991	.990	.984	.998	.998	**.999**
NonInvasiveFetalECGThorax2	.996	**.999**	.998	**.999**	.998	.995	.997	.997	.996	.997	.997	.994	.994	.988	.998	.998	**.999**
OliveOil	.917	.986	.990	.967	.903	**.991**	.983	**.991**	.982	.970	.973	.954	.958	.696	.980	.982	.982
OSULeaf	.945	**1.000**	.996	.997	.973	.991	.940	.996	.985	.964	.955	.960	.960	.925	.995	.985	.990
PhalangesOutlinesCorrect	.866	.888	.906	**.912**	.859	.904	.904	.889	.883	.898	.905	.844	.839	.852	.891	.905	.904
Phoneme	.778	.752	.713	.776	.632	**.841**	.736	.828	.815	.789	.776	.789	.774	.618	.831	.796	.823
PickupGestureWiimoteZ	.935	.978	.988	.985	.962	.969	.936	**.990**	.983	.942	.945	.967	.968	.946	.971	.954	.973
PigAirwayPressure	.842	.991	.997	**1.000**	.977	.845	.561	.936	.979	.578	.560	.925	.922	.687	.922	.593	.734
PigArtPressure	.992	**1.000**	.996	**1.000**	.674	.993	.596	.996	.982	.750	.772	.997	.998	.779	.998	.848	.986
PigCVP	.953	**.999**	.998	**.999**	.909	.978	.607	.992	.987	.676	.701	.993	.992	.734	.988	.906	.983
PLAID	.953	.973	.986	.768	.844	.977	.954	**.989**	.983	.958	.954	.967	.968	.907	.981	.954	.965
Plane	**1.000**	**1.000**	**1.000**	**1.000**	.999	**1.000**	**1.000**	**1.000**	**1.000**	.999	.999	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
PowerCons	.962	.999	**1.000**	**1.000**	.992	.987	.988	.991	.978	.967	.964	.992	.992	.979	.979	.995	.999
ProximalPhalanxOutlineAgeGroup	.924	.936	.913	.926	.938	.938	.935	.942	.927	.934	.933	.932	.926	.899	.945	**.948**	.940
ProximalPhalanxOutlineCorrect	.898	.952	**.965**	.960	.934	.936	.939	.927	.921	.928	.929	.905	.906	.897	.953	.948	.955
ProximalPhalanxTW	.902	.872	.874	.898	.918	.910	.916	.928	.898	.921	.920	.916	.904	.827	.923	.927	**.937**
RefrigerationDevices	.717	.693	.656	.682	.236	.759	.778	.726	.694	.705	.705	.713	.722	.695	.761	**.789**	.699
Rock	.941	.981	.950	.891	.763	.957	.984	.952	.878	.942	.903	.880	.886	.831	.976	.985	**.994**
ScreenType	.707	.677	.644	**.780**	.604	.712	.621	.688	.670	.609	.596	.690	.698	.704	.740	.672	.657
SemgHandGenderCh2	.941	.924	.968	.941	.990	.975	.943	.974	.977	.879	.869	.943	.947	.917	.955	.986	**.992**
SemgHandMovementCh2	.914	.825	.915	.869	**.985**	.939	.903	.946	.949	.855	.858	.927	.925	.866	.943	.900	.974
SemgHandSubjectCh2	.968	.946	.978	.930	.995	.970	.954	.973	.966	.920	.925	.939	.944	.959	.964	.974	**.996**
ShakeGestureWiimoteZ	.971	.999	.999	.966	.980	.984	.968	.995	.964	.974	.966	.977	.976	.936	.988	.987	**1.000**
ShapeletSim	.999	**1.000**	**1.000**	.999	**1.000**	**1.000**	.993	.997	.927	.988	.962	.978	.960	.968	**1.000**	.996	.996
ShapesAll	.984	.996	.990	**.997**	.992	.995	.973	.993	.990	.990	.989	.988	.985	.959	.991	.985	.996
SmallKitchenAppliances	.922	.946	.944	.902	.880	**.949**	.935	.948	.930	.865	.856	.933	.934	.885	.939	.941	.933
SmoothSubspace	.966	.987	.994	.999	.999	.995	.994	.995	.500	.948	.901	.986	.983	.948	.993	.992	**1.000**
SonyAIBORobotSurface1	.978	**.999**	.993	.994	.992	.984	.963	.982	.887	.989	.978	.963	.962	.977	.995	.990	.996
SonyAIBORobotSurface2	.973	**.994**	.985	.991	.968	.957	.926	.987	.893	.958	.964	.918	.899	.958	.959	.922	.976
StarLightCurves	.991	.993	**.995**	.993	.991	.994	.992	**.995**	.994	.988	.987	.994	.994	.992	.994	**.995**	.993
Strawberry	.973	.992	.993	**.996**	.993	.987	.990	.989	.989	.986	.985	.983	.982	.964	.987	.991	.994
SwedishLeaf	.996	**.999**	**.999**	**.999**	.997	.998	.991	**.999**	.996	.996	.996	.996	.996	.979	.998	.998	**.999**
Symbols	.997	.998	.997	**1.000**	.999	.998	.992	.998	.994	.998	.998	.995	.996	.990	.998	.997	.998
SyntheticControl	**1.000**	.999	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	.999	**1.000**	.999	.999	.999	**1.000**	**1.000**	**1.000**	**1.000**
ToeSegmentation1	.950	.989	.989	.990	.980	.978	.963	**.996**	**.996**	.975	.981	.927	.944	.941	.990	.995	.989
ToeSegmentation2	.856	.968	.978	**.993**	.970	.960	.857	.970	.963	.945	.946	.895	.894	.933	.978	.981	.986
Trace	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
TwoLeadECG	.929	**1.000**	**1.000**	**1.000**	.998	.991	.994	**1.000**	.988	.946	.914	.990	.988	.984	.987	.999	.986
TwoPatterns	.961	.999	**1.000**	**1.000**	**1.000**	.990	.986	.999	.996	.992	.995	.969	.967	.894	.996	**1.000**	**1.000**
UMD	.978	.998	.993	.999	**1.000**	.999	.999	.999	.989	.995	.993	.995	.995	.998	.995	.999	.999
UWaveGestureLibraryAll	.977	.998	.998	.995	.995	.994	.994	.989	.970	.991	.990	.987	.984	.991	.991	.995	**.999**
UWaveGestureLibraryX	.959	.980	.978	.971	.967	.971	.960	.975	.952	.968	.966	.973	.972	.948	.974	.971	**.981**
UWaveGestureLibraryY	.946	.964	.963	.958	.948	.953	.946	.961	.930	.950	.949	.956	.954	.926	.958	.953	**.968**
UWaveGestureLibraryZ	.950	.972	.971	.962	.962	.964	.953	.967	.950	.958	.957	.964	.964	.927	.964	.959	**.973**
Wafer	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	.999	**1.000**	**1.000**	.999	.999	.999	**1.000**	**1.000**	.993	**1.000**	**1.000**	**1.000**
Wine	.420	**.981**	.907	.805	.918	.913	.895	.936	.870	.877	.931	.837	.881	.944	.835	.807	.863
WordSynonyms	.892	.944	.937	.956	.921	.932	.870	.935	.909	.932	.933	.885	.876	.804	.916	.902	**.959**
Worms	.946	.945	.915	.936	.931	.953	.901	.958	.956	.928	.894	.943	.940	.859	.968	.965	**.970**
WormsTwoClass	**.936**	.882	.857	.888	.879	.888	.809	.893	.871	.795	.811	.852	.822	.858	.926	.879	.928
Yoga	.866	**.978**	.967	.973	.950	.895	.875	.929	.913	.919	.923	.908	.909	.899	.895	.915	.955
Mean	.915	**.948**	.946	.947	.917	.943	.918	.943	.909	.919	.917	.924	.923	.889	.941	.938	**.948**
Mean rank	11.98	5.92	5.76	5.74	8.62	7.07	10.51	6.45	11.09	11.20	11.53	11.41	11.72	14.56	7.15	7.31	**4.98**

Table 10: Per-dataset accuracy on the UCR archive (128 datasets; higher is better). Best per dataset in bold. Mean and mean rank are computed over the 128 datasets with results for all models. Models with a tuned variant also show their untuned variant; ranks are computed over all columns jointly.

Dataset	Catch22	DTW-1NN	Hydra	MiniRocket	InceptionTime	TS2Vec	Chronos-2	Chronos-2 (untuned)	MantisV2	MantisV2 (untuned)	Moment	Moment (untuned)	NuTime	NuTime (untuned)	TICT	TiRex	TiRex (untuned)	TimEE
ACSF1	.830	.620	.850	.910	**.930**	.890	.850	.820	.830	.790	.820	.760	.760	.720	.400	.890	.790	.830
Adiac	.719	.609	.818	.816	.831	.754	.831	.770	**.844**	.762	.767	.772	.749	.739	.481	.788	.783	.829
AllGestureWiimoteX	.529	.717	.703	.736	**.774**	.726	.586	.483	.633	.513	.590	.583	.566	.586	.570	.604	.581	.669
AllGestureWiimoteY	.613	.730	.731	.760	**.789**	.721	.639	.570	.677	.504	.647	.637	.540	.543	.583	.684	.637	.729
AllGestureWiimoteZ	.521	.651	.686	.729	**.806**	.709	.601	.496	.691	.527	.580	.587	.541	.536	.493	.590	.537	.697
ArrowHead	.760	.800	.834	**.863**	.846	.829	.817	.829	.817	.817	.811	.823	.777	.743	.766	.754	.829	.823
Beef	.600	.667	.833	.833	.667	.667	.733	.667	.700	.600	.733	.733	.700	.633	.600	.767	**.900**	**.900**
BeetleFly	.750	.700	.900	.850	.850	.850	.850	.800	.950	.900	.900	**1.000**	.850	.850	.850	.950	.850	.850
BirdChicken	.850	.700	.900	.900	**.950**	.800	.900	.900	.900	.800	.900	.800	**.950**	**.950**	.900	.900	.850	.900
BME	.933	.980	**1.000**	**1.000**	**1.000**	.987	.987	.987	**1.000**	.827	.967	.907	.840	.840	.687	**1.000**	.993	**1.000**
Car	.783	.767	**.933**	.917	.883	.683	.733	.767	.733	.767	.767	.783	.767	.750	.617	.800	.833	.900

continued on next page

(continued)

Dataset	Catch22	DTW-1NN	Hydra	MiniRocket	InceptionTime	TS2Vec	Chronos-2	Chronos-2 (untuned)	MantisV2	MantisV2 (untuned)	Moment	Moment (untuned)	NuTime	NuTime (untuned)	TICT	TiRex	TiRex (untuned)	TimeEE
CBF	.961	.996	.993	**.999**	**.999**	**.999**	.993	**.999**	.997	.992	.959	.967	.976	.974	.942	.998	.990	**.999**
Chinatown	.921	.953	.983	.983	**.985**	.959	.980	.974	.950	.726	.980	.974	.936	.924	.930	.974	.971	**.985**
ChlorineConcentration	.599	.650	.760	.755	**.855**	.825	.710	.714	.704	.645	.709	.700	.673	.676	.601	.724	.749	.609
CinCECGTorso	.808	.930	**.995**	.857	.845	.775	.822	.926	.812	.693	.604	.605	.736	.740	.643	.992	.989	.989
Coffee	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	.964	**1.000**	**1.000**	**1.000**	.929	**1.000**	.929	.964	.964	**1.000**	**1.000**	**1.000**
Computers	.724	.620	.716	.732	**.832**	.660	.744	.728	.736	.736	.668	.656	.772	.764	.688	.752	.712	.656
CricketX	.579	.772	.800	.821	**.841**	.797	.708	.569	.777	.736	.687	.674	.667	.664	.603	.687	.697	.810
CricketY	.546	.759	.841	.828	**.862**	.756	.721	.646	.790	.759	.662	.667	.692	.664	.579	.710	.754	.851
CricketZ	.636	.746	.805	.828	**.862**	.815	.751	.610	.818	.785	.710	.713	.672	.685	.610	.718	.726	.841
Crop	.651	.712	.734	.766	**.797**	.750	.736	.705	.736	.710	.708	.711	.672	.670	.656	.747	.737	.784
DiatomSizeReduction	.941	.935	.948	.928	.941	.967	.886	.899	.820	.843	.866	.866	.850	.817	.869	.869	.863	**.987**
DistalPhalanxOutlineAgeGroup	.712	.626	.763	.734	.719	.727	.763	.763	.763	.734	.734	.727	.734	.734	.719	.755	.748	**.791**
DistalPhalanxOutlineCorrect	.783	.725	.786	.790	.757	.732	**.812**	.779	.793	.772	.779	.786	.775	.761	.732	.779	.793	.786
DistalPhalanxTW	.669	.633	**.712**	.655	.683	.676	.698	.669	**.712**	.669	.691	.691	.676	.691	.683	.683	.676	.676
DodgerLoopDay	.519	.588	.519	.623	.571	**.662**	.532	.558	.558	.455	.390	.442	.545	.545	.532	.558	.623	.649
DodgerLoopGame	.693	**.928**	.819	.850	.858	.921	.866	.866	.795	.583	.764	.764	.780	.787	.858	.740	.843	.913
DodgerLoopWeekend	.944	.978	.976	**.984**	**.984**	**.984**	.937	.952	.929	.952	.921	.937	.960	.968	.944	.913	.952	**.984**
Earthquakes	.748	.727	.734	.727	.748	.748	.741	.727	.741	.748	**.755**	**.755**	.741	.741	**.755**	.748	.748	.748
ECG200	.820	.880	.870	.910	**.930**	.910	.860	.830	.830	.820	.880	.860	.830	.780	.790	.830	.860	.920
ECG5000	.938	.925	**.949**	.945	.941	.935	.935	.934	.938	.939	.940	.943	.934	.932	.927	.939	.940	**.949**
ECGFiveDays	.777	.797	**1.000**	**1.000**	**1.000**	**1.000**	.913	.941	.861	.930	.835	.846	.815	.779	.861	.923	.980	.994
ElectricDevices	.725	.619	**.752**	.742	.708	.731	.749	.716	.745	.639	.647	.644	.707	.706	.629	.713	.709	.609
EOGHorizontalSignal	.552	.475	.572	.613	.616	.506	.533	.470	.569	.608	.492	.486	.439	.417	.403	.561	.500	**.635**
EOGVerticalSignal	.456	.475	.492	.522	.486	.506	.401	.398	.483	.403	.392	.390	.290	.301	.390	.423	.470	**.547**
EthanolLevel	.352	.282	.588	.578	**.822**	.454	.420	.416	.360	.246	.400	.386	.352	.346	.340	.378	.672	.644
FaceAll	.764	.808	**.898**	.807	.796	.777	.706	.742	.719	.661	.668	.684	.652	.636	.739	.716	.881	.802
FaceFour	.648	.886	.886	.989	.955	.886	.716	.659	.909	.795	.773	.795	.830	.739	.784	.727	.852	**1.000**
FacesUCR	.695	.912	.956	.960	**.962**	.929	.761	.783	.844	.759	.727	.696	.713	.705	.684	.811	.849	.941
FiftyWords	.602	.758	**.842**	.837	.791	.765	.670	.633	.719	.609	.714	.692	.613	.591	.587	.637	.701	.811
Fish	.749	.846	**.989**	.977	.983	.920	.874	.874	.937	.914	.851	.869	.920	.926	.754	.920	.897	.926
FordA	.920	.691	**.961**	.948	**.961**	.927	.918	.783	.930	.904	.875	.898	.896	.900	.898	.958	.948	.949
FordB	.752	.607	.825	.820	**.859**	.798	.791	.633	.800	.767	.747	.772	.756	.754	.723	.838	.793	.802
FreezerRegularTrain	.998	.907	.998	**1.000**	.997	.984	.972	.944	.984	.918	.898	.892	.978	.977	.896	.939	.994	.998
FreezerSmallTrain	.950	.676	.926	**.967**	.840	.885	.925	.843	.930	.774	.755	.751	.944	.945	.807	.811	.919	.943
Fungi	.903	.823	**1.000**	**1.000**	.995	.962	.935	.892	.930	.855	.962	.968	.742	.661	.796	.882	.930	.941
GestureMidAirD1	.746	.639	.754	.754	.669	.485	.746	.754	.700	.685	.769	.754	.562	.562	.646	.746	.746	**.785**
GestureMidAirD2	.662	.600	.685	.662	.508	.392	.654	.608	**.708**	.662	.685	.700	.485	.462	.577	.685	**.708**	.685
GestureMidAirD3	.469	.377	.485	.423	.246	.200	**.554**	.523	.400	.246	.462	.438	.315	.269	.338	.469	.500	**.554**
GesturePebbleZ1	.779	.826	.890	**.907**	.855	.692	.890	.831	.884	.808	.872	.826	.767	.791	.767	.890	.872	.860
GesturePebbleZ2	.753	.778	.886	.892	.835	.722	.873	.848	.880	.785	.880	.823	.778	.785	.759	.835	**.905**	.816
GunPoint	.953	.913	**1.000**	.993	**1.000**	.980	.993	.987	.980	.967	.973	.973	.953	.960	.940	.927	.980	.987
GunPointAgeSpan	.956	.965	**1.000**	.997	.978	.965	.987	.987	.984	.994	.946	.943	.972	.972	.972	.975	.959	.991
GunPointMaleVersusFemale	.994	.975	**1.000**	**1.000**	.997	.997	.991	.978	**1.000**	.991	.984	.984	.972	.953	.997	.994	.987	**1.000**
GunPointOldVersusYoung	.975	.965	**1.000**	**1.000**	**1.000**	**1.000**	.997	.956	.997	.997	.975	.965	**1.000**	**1.000**	.987	**1.000**	.943	**1.000**
Ham	.600	.600	.743	.714	.705	.676	.657	.705	.676	.648	.695	.695	.733	.733	.667	.648	**.762**	.686
HandOutlines	.854	.862	.941	.935	**.957**	.916	.914	.897	.903	.805	.897	.905	.892	.892	.886	.900	.895	.932
Haptics	.445	.412	.513	.529	**.558**	.523	.526	.510	.497	.490	.464	.464	.445	.461	.451	.526	.500	.552
Herring	.578	.531	**.750**	.688	.703	.594	.688	.609	.609	.672	.594	.609	.609	.625	.656	.625	.531	.578
HouseTwenty	.958	.941	.958	.958	.924	.908	.950	.882	.941	.933	.916	.916	.866	.866	.899	**.966**	.958	.958
InlineSkate	.451	.387	**.491**	.476	.473	.424	.433	.264	.384	.467	.325	.338	.335	.336	.384	.442	.320	.442
InsectEPGRegularTrain	.960	.827	**1.000**	**1.000**	**1.000**	**1.000**	.996	.807	**1.000**	**1.000**	.928	.916	**1.000**	**1.000**	.900	**1.000**	.904	.964
InsectEPGSmallTrain	.827	.695	**1.000**	.992	**1.000**	**1.000**	.884	.779	**1.000**	.996	.851	.871	**1.000**	**1.000**	.795	.968	.787	.956
InsectWingbeatSound	.566	.585	.655	.672	.631	.617	.632	.653	.603	.540	.617	.627	.526	.508	.494	.657	**.675**	.655
ItalyPowerDemand	.881	.955	.966	.963	**.969**	.964	.952	.958	.927	.501	.949	.956	.882	.879	.867	.964	.964	.961
LargeKitchenAppliances	.827	.795	.883	.875	**.896**	.861	.888	.733	.763	.709	.757	.760	.757	.760	.717	.813	.699	.675
Lightning2	.705	**.869**	.721	.754	.787	**.869**	.721	.705	.721	.721	.754	.705	.738	.738	.754	.738	.738	.852
Lightning7	.671	.712	.808	.795	**.849**	.822	.658	.699	.685	.658	.685	.712	.699	.685	.712	.726	.699	**.849**
Mallat	.899	.914	.941	.946	.949	.887	.909	.838	.891	.900	.897	.890	.838	.850	.501	**.959**	**.959**	.924
Meat	.867	.933	.917	.967	.933	.933	.933	.900	.933	.917	.933	.950	.917	.883	.617	.883	.917	**.983**
MedicalImages	.762	.747	.778	.791	.795	**.812**	.729	.651	.759	.687	.708	.713	.714	.705	.758	.721	.763	.796
MelbournePedestrian	.804	.816	.931	.967	**.975**	.964	.945	.890	.953	.105	.865	.857	.925	.915	.805	.953	.898	.931
MiddlePhalanxOutlineAgeGroup	.617	.520	.597	.552	.513	.649	.552	.604	.591	.558	.584	.571	**.656**	.617	.578	.584	.604	.636
MiddlePhalanxOutlineCorrect	.766	.766	.835	.849	.842	.825	**.863**	.849	.835	.797	.835	.845	.780	.787	.784	.859	.852	.852
MiddlePhalanxTW	.532	.506	.539	.526	.552	.584	.552	.539	.526	.532	.584	.552	.552	.552	.545	.526	.539	**.591**
MixedShapesRegularTrain	.918	.909	**.981**	.971	.967	.920	.942	.925	.948	.938	.890	.873	.936	.938	.919	.964	.949	.979
MixedShapesSmallTrain	.864	.833	**.965**	.950	.916	.855	.904	.909	.921	.892	.819	.805	.917	.911	.797	.943	.921	.955
MoteStrain	.867	.866	.929	.927	.897	.860	.926	.891	.934	.858	.903	.898	**.942**	.941	.899	.899	.922	.900
NonInvasiveFetalECGThorax1	.852	.811	.943	.949	**.963**	.937	.834	.867	.862	.815	.873	.863	.782	.778	.769	.918	.914	.931
NonInvasiveFetalECGThorax2	.882	.871	.958	**.964**	.962	.940	.866	.892	.887	.867	.909	.907	.820	.818	.813	.927	.931	.942
OliveOil	.700	.867	.933	.933	.800	.900	.867	.933	.833	.833	.867	.900	.700	.733	.400	.800	.933	**.967**
OSULeaf	.707	.612	**.988**	.959	.950	.843	.901	.657	.934	.860	.748	.740	.798	.798	.727	.938	.802	.868
PhalangesOutlinesCorrect	.784	.761	.823	.840	.838	.772	.836	.832	.829	.821	.825	.838	.776	.784	.784	.823	**.841**	.824
Phoneme	.309	.227	.324	.294	.352	.310	.399	.222	.371	.345	.312	.310	.293	.294	.183	.389	.322	**.410**
PickupGestureWiimoteZ	.540	.660	.820	.880	**.920**	.740	.800	.700	.840	.840	.620	.660	.740	.720	.700	.820	.740	.780
PigAirwayPressure	.308	.096	.769	.865	**.947**	.644	.351	.101	.476	.625	.106	.101	.332	.389	.139	.399	.096	.130
PigArtPressure	.909	.197	.986	.981	**1.000**	.971	.812	.144	.933	.851	.173	.163	.942	.938	.332	.923	.317	.639
PigCVP	.538	.159	**.966**	.947	.952	.861	.745	.096	.861	.846	.120	.101	.832	.832	.250	.851	.389	.731
PLAID	.767	.834	.873	**.931**	.467	.540	.844	.780	.886	.851	.780	.777	.788	.793	.667	.838	.778	.844
Plane	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	.971	**1.000**	**1.000**	**1.000**	**1.000**	.981	.981	**1.000**	.990	**1.000**	**1.000**	**1.000**	**1.000**
PowerCons	.917	.922	.978	**.994**	**.994**	.961	.922	.906	.972	.911	.856	.850	.928	.933	.917	.917	.978	.989
ProximalPhalanxOutlineAgeGroup	.834	.785	**.868**	.834	.849	.834	.859	.859	.829	.834	.849	.844	.849	.849	.771	.854	.863	.863
ProximalPhalanxOutlineCorrect	.838	.790	.911	.904	**.914**	.890	.859	.852	.869	.863	.883	.880	.835	.838	.832	.883	.904	.907
ProximalPhalanxTW	.766	.756	.810	.800	.771	.795	.800	.805	**.824**	.776	.805	.800	.795	.780	.776	.815	.820	.805

continued on next page

(continued)

Dataset	Catch22	DTW-1NN	Hydra	MiniRocket	InceptionTime	TS2Vec	Chronos-2	Chronos-2 (untuned)	MantisV2	MantisV2 (untuned)	Moment	Moment (untuned)	NuTime	NuTime (untuned)	TICT	TiRex	TiRex (untuned)	TimeEE
RefrigerationDevices	.517	.440	.528	.480	.531	**.616**	.560	.560	.557	.501	.528	.520	.528	.533	.528	.557	.581	.501
Rock	.740	.840	.920	.820	.620	.760	.880	**.940**	.820	.580	.760	.720	.680	.720	.640	.840	**.940**	**.940**
ScreenType	.533	.411	.493	.461	**.589**	.413	.507	.445	.467	.485	.405	.429	.488	.507	.544	.541	.491	.448
SemgHandGenderCh2	.877	.845	.847	.888	.875	**.958**	.920	.882	.913	.920	.810	.798	.860	.865	.825	.888	.950	.957
SemgHandMovementCh2	.662	.638	.496	.676	.558	**.876**	.718	.627	.744	.753	.551	.553	.680	.671	.587	.709	.622	.813
SemgHandSubjectCh2	.807	.800	.771	.867	.682	**.951**	.844	.798	.831	.816	.724	.724	.733	.733	.820	.827	.867	.940
ShakeGestureWiimoteZ	.800	.840	.920	.920	.840	.900	.920	.860	.940	.900	.820	.780	.840	.860	.660	.860	.820	**.960**
ShapeletSim	.972	.700	.989	**1.000**	.983	.994	**1.000**	.961	.972	.833	.928	.911	.928	.861	.900	.967	.928	.967
ShapesAll	.777	.802	**.928**	.923	.925	.903	.883	.800	.875	.847	.830	.823	.842	.823	.698	.857	.835	.900
SmallKitchenAppliances	.805	.672	.808	.827	.792	.699	.827	.811	**.835**	.792	.683	.691	.813	.813	.720	.824	.832	.803
SmoothSubspace	.853	.947	.927	.953	.973	.980	.933	.973	.947	.333	.793	.740	.900	.907	.833	.953	.933	**.993**
SonyAIBORobotSurface1	.845	.696	.920	.887	.869	.913	.709	.609	.802	.710	.779	.744	.772	.752	.894	.902	.877	**.923**
SonyAIBORobotSurface2	.917	.859	.948	.921	**.951**	.908	.879	.811	.937	.801	.886	.888	.838	.807	.887	.852	.818	.916
StarLightCurves	.970	.905	**.982**	**.982**	.980	.971	.980	.966	.981	.980	.955	.950	.979	.979	.978	.979	.979	**.982**
Strawberry	.927	.946	.973	**.984**	**.984**	.968	.949	.959	.965	.957	.938	.946	.935	.935	.911	.949	.957	.976
SwedishLeaf	.890	.846	.978	.962	.965	.931	.944	.858	.955	.912	.907	.914	.928	.930	.819	.944	.946	**.979**
Symbols	.958	.938	.980	**.983**	**.983**	.979	.979	.920	.969	.878	.969	.955	.942	.940	.920	.970	.956	.966
SyntheticControl	.987	.983	.957	.987	.997	**1.000**	.997	.997	.990	.983	.990	.980	.970	.973	.977	.987	.987	.983
ToeSegmentation1	.860	.750	.961	.961	**.974**	.917	.921	.908	.961	.965	.934	.934	.855	.860	.860	.939	.947	.939
ToeSegmentation2	.777	.908	.908	.923	**.946**	.900	.854	.746	.869	.862	.892	.862	.746	.738	.862	.923	.923	.892
Trace	**1.000**	.990	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	.990	.990	**1.000**	**1.000**	.990	**1.000**
TwoLeadECG	.818	.868	**.999**	.998	.996	.977	.934	.955	.995	.926	.847	.829	.930	.920	.948	.934	.992	.946
TwoPatterns	.846	.998	.987	.997	**1.000**	**1.000**	.922	.912	.982	.952	.924	.946	.846	.835	.706	.954	.993	.996
UMD	.896	.972	**.993**	**.993**	**.993**	.986	.979	.986	.986	.917	.965	.965	.951	.944	.972	.965	.958	.965
UWaveGestureLibraryAll	.831	.966	.973	.971	.934	.936	.926	.944	.886	.797	.910	.905	.889	.878	.912	.908	.942	**.979**
UWaveGestureLibraryX	.761	.773	**.854**	.848	.822	.802	.807	.764	.809	.740	.803	.803	.808	.807	.774	.808	.806	.849
UWaveGestureLibraryY	.707	.699	.792	.776	.762	.720	.738	.703	.759	.658	.724	.725	.742	.733	.720	.748	.715	**.793**
UWaveGestureLibraryZ	.707	.678	**.803**	.801	.775	.762	.745	.728	.758	.706	.742	.737	.756	.750	.712	.740	.741	.784
Wafer	.997	.996	**1.000**	.999	.999	.998	.986	.991	.996	.981	.988	.990	.994	.994	.984	.997	.998	.998
Wine	.463	.611	.907	.852	.685	**.926**	.889	.815	.889	.722	.833	.815	.722	.796	.889	.759	.630	.778
WordSynonyms	.538	.738	.743	**.760**	.716	.693	.569	.545	.616	.549	.589	.596	.539	.527	.513	.524	.574	.757
Worms	.753	.532	.753	.740	.766	.753	.779	.545	.766	.792	.675	.636	.688	.740	.688	**.805**	.701	**.805**
WormsTwoClass	.857	.584	.766	.779	.766	.792	.805	.675	.857	.792	.688	.727	.792	.779	.818	.844	.753	**.870**
Yoga	.771	.844	**.927**	.910	.912	.895	.809	.785	.838	.824	.837	.840	.824	.827	.809	.799	.820	.879
Mean	.762	.750	.850	**.852**	.842	.819	.809	.764	.820	.761	.765	.762	.771	.766	.727	.813	.803	.841
Mean rank	12.78	12.79	4.80	**4.68**	5.39	7.81	8.35	11.38	7.45	11.95	11.59	11.86	12.05	12.40	13.88	8.02	8.49	5.32

Table 11: Per-dataset ROC AUC on the UEA archive (29 datasets; higher is better). Best per dataset in bold. Mean and mean rank are computed over the 27 datasets with results for all models. Models with a tuned variant also show their untuned variant; ranks are computed over all columns jointly.

Dataset	Catch22	Hydra	MiniRocket	InceptionTime	TS2Vec	Chronos-2	Chronos-2 (untuned)	MantisV2	MantisV2 (untuned)	Moment	Moment (untuned)	NuTime	NuTime (untuned)	TICT	TiRex	TiRex (untuned)	TimEE (PV)	TimEE (VP)
ArticularyWordRecognition	.999	.999	**1.000**	**1.000**	.999	.999	.998	**1.000**	.999	.999	.999	**1.000**	.999	.999	.999	.999	.998	.999
AtrialFibrillation	.323	.313	.227	.413	.303	.373	.453	.360	.303	.427	.413	.397	.410	.260	.507	.577	**.580**	.353
BasicMotions	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	.997	**1.000**	**1.000**	.997	.994	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
CharacterTrajectories	.998	**1.000**	**1.000**	**1.000**	**1.000**	.999	.997	.999	.999	**1.000**	**1.000**	**1.000**	.999	.994	.999	.999	.999	**1.000**
Cricket	.998	.999	**1.000**	**1.000**	.999	**1.000**	.999	**1.000**	.999	.998	.998	**1.000**	**1.000**	.999	**1.000**	**1.000**	.999	**1.000**
DuckDuckGeese	.794	.879	**.895**	.872	.842	.732	.791	.810	.761	.784	.768	.757	.757	.664	.714	.757	.682	.689
EigenWorms	.984	**1.000**	.996	.962	–	.975	.937	.972	.986	–	–	.953	.959	.931	.974	.968	.987	.968
Epilepsy	.999	**1.000**	**1.000**	.999	.998	**1.000**	.996	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	.998	**1.000**	**1.000**	**1.000**	**1.000**
ERing	.994	**1.000**	**1.000**	.994	.993	.999	.992	**1.000**	.991	.996	.997	.999	.998	.978	.999	.996	.997	.999
EthanolConcentration	.536	.762	.697	.522	.628	.609	.541	.662	.674	.508	.509	.680	.673	.520	.530	.538	**.833**	.831
FaceDetection	.597	.525	.623	**.753**	.582	.605	.538	.578	.536	.599	.598	.600	.601	.614	.663	.689	.657	.491
FingerMovements	.539	.556	.602	.567	.532	.506	.550	.565	.492	.573	.587	.546	.524	.549	.539	.559	.606	**.632**
HandMovementDirection	.499	.515	.642	**.679**	.563	.518	.526	.552	.514	.550	.523	.602	.586	.610	.551	.502	.629	.568
Handwriting	.799	.880	.899	**.951**	.843	.780	.763	.810	.762	.800	.789	.738	.733	.729	.765	.770	.837	.827
Heartbeat	.641	.771	**.799**	.767	.729	.750	.750	.782	.790	.693	.740	.784	.779	.628	.750	.733	.738	.761
JapaneseVowels	.961	.999	**1.000**	**1.000**	.997	.970	.968	.998	.500	.959	.955	.999	.999	.915	.965	.968	.968	.981
Libras	.991	.993	.995	.990	.991	.993	.988	**.996**	.991	.985	.984	.994	.995	.961	.994	.994	.991	.994
LSST	.878	.842	.873	.743	.905	.851	.880	**.906**	.887	.837	.829	.867	.874	.801	.854	.848	.881	.861
MotorImagery	.463	.497	.561	.595	.542	.584	**.633**	.505	.504	.571	.595	.532	.601	.541	.559	.444	.582	.478
NATOPS	.973	.975	.991	**.997**	.990	.970	.966	.983	.983	.971	.972	.976	.974	.941	.970	.964	.968	.977
PEMS-SF	**1.000**	.967	.967	.970	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	.993	**1.000**	**1.000**	.997	.995
PenDigits	.998	.998	–	**1.000**	.999	.999	.999	.999	.500	.999	.999	.998	.998	.991	.999	.999	.994	.996
PhonemeSpectra	.846	.871	.867	**.908**	.837	.881	.831	.904	.872	.853	.853	.882	.879	.661	.877	.875	.906	.891
RacketSports	.952	.974	.966	.975	.962	.956	.954	.977	.500	.945	.937	**.978**	**.978**	.931	.952	.956	.967	.973
SelfRegulationSCP1	.790	.969	**.971**	.944	.907	.895	.816	.948	.873	.854	.808	.926	.918	.803	.860	.900	.959	.957
SelfRegulationSCP2	.581	**.593**	.541	.485	.526	.492	.530	.512	.534	.459	.446	.520	.504	.461	.458	.509	.472	.426
SpokenArabicDigits	.997	**1.000**	**1.000**	**1.000**	.998	.998	.992	.999	.996	.998	.997	.997	.997	.994	.997	.996	.996	.995
StandWalkJump	.530	.633	.693	.653	.377	.767	.807	.587	.583	.843	.763	.640	.653	**.880**	.767	.667	.580	.580
UWaveGestureLibrary	.980	**.992**	**.992**	.988	.991	.988	.953	.990	.956	**.992**	**.992**	.985	.986	.950	.983	.986	.986	.990

continued on next page

(continued)

Dataset	Catch22	Hydra	MiniRocket	InceptionTime	TS2Vec	Chronos-2	Chronos-2 (untuned)	MantisV2	MantisV2 (untuned)	Moment	Moment (untuned)	NuTime	NuTime (untuned)	TICT	TiRex	TiRex (untuned)	TimEE (PV)	TimEE (VP)
Mean	.802	.834	.844	.843	.816	.823	.823	.830	.778	.822	.817	.830	.830	.792	.824	.823	**.845**	.824
Mean rank	12.57	7.70	**5.26**	6.72	9.89	9.22	12.00	6.54	11.67	10.76	11.37	7.41	7.69	14.65	9.76	9.81	8.98	9.00

Table 12: Per-dataset accuracy on the UEA archive (30 datasets; higher is better). Best per dataset in bold. Mean and mean rank are computed over the 27 datasets with results for all models. Models with a tuned variant also show their untuned variant; ranks are computed over all columns jointly.

Dataset	Catch22	DTW-1NN	Hydra	MiniRocket	InceptionTime	TS2Vec	Chronos-2	Chronos-2 (untuned)	MantisV2	MantisV2 (untuned)	Moment	Moment (untuned)	NuTime	NuTime (untuned)	TICT	TiRex	TiRex (untuned)	TimEE (PV)	TimEE (VP)
ArticularyWordRecognition	.983	.987	.990	**.993**	.987	.990	**.993**	.953	**.993**	**.993**	.990	.987	**.993**	**.993**	.970	.990	.990	.980	.990
AtrialFibrillation	.200	.200	.133	.133	.200	.200	.133	**.333**	.133	.200	.133	.267	.267	.200	.267	.267	.200	**.333**	.133
BasicMotions	.950	.975	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	.950	**1.000**	**1.000**	.950	.925	**1.000**	**1.000**	**1.000**	.950	**1.000**	**1.000**	**1.000**
CharacterTrajectories	.960	.990	.990	.994	**.996**	.987	.964	.929	.967	.957	.976	.973	.979	.973	.924	.957	.959	.985	.989
Cricket	.986	**1.000**	.986	.986	.986	.986	.986	.958	.986	.986	.944	.931	**1.000**	**1.000**	.931	**1.000**	**1.000**	.972	.972
DuckDuckGeese	.440	.600	.680	**.720**	.640	.480	.420	.460	.500	.580	.440	.480	.440	.400	.380	.340	.460	.340	.360
EigenWorms	.870	.618	**.985**	.947	.664	–	.786	.725	.824	.870	–	–	.725	.740	.672	.786	.794	.901	.847
Epilepsy	.949	.964	**1.000**	**1.000**	.964	.978	**1.000**	.949	**1.000**	**1.000**	.971	.978	**1.000**	**1.000**	.964	**1.000**	**1.000**	**1.000**	**1.000**
ERing	.926	.930	.985	.981	.919	.919	.981	.930	**.993**	.911	.952	.948	**.993**	.974	.856	.978	.974	.959	.970
EthanolConcentration	.293	.323	.548	.471	.262	.327	.380	.312	.414	.392	.274	.278	.414	.388	.274	.251	.289	.612	**.624**
FaceDetection	.566	.529	.523	.582	**.696**	.556	.567	.527	.552	.528	.565	.570	.561	.571	.584	.611	.635	.520	.493
FingerMovements	.530	.530	.550	.540	.530	.500	.510	.510	.510	.490	.570	.590	.530	.560	.510	.530	.570	**.610**	.580
HandMovementDirection	.189	.231	.216	.365	**.392**	.270	.297	.270	.311	.176	.378	.284	.324	.324	.351	.270	.243	.324	.324
Handwriting	.272	.607	.466	.521	**.667**	.332	.279	.235	.281	.247	.268	.269	.204	.206	.208	.256	.266	.422	.455
Heartbeat	.727	.717	.761	.756	.751	.771	.741	.722	.790	**.829**	.732	.732	.766	.785	.722	.722	.717	.717	.732
InsectWingbeat	–	**.115**	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
JapaneseVowels	.759	.949	.978	.981	**.992**	.946	.805	.792	.954	.078	.786	.754	.954	.957	.641	.770	.819	.784	.824
Libras	.844	.872	**.933**	.911	.878	.833	.867	.789	.928	.861	.800	.789	.883	.906	.678	.867	.883	.872	.889
LSST	.607	.551	.607	.638	.194	.639	.587	.604	**.663**	.633	.548	.552	.565	.563	.548	.573	.579	.604	.592
MotorImagery	.510	.500	.500	.520	.550	.490	.550	.530	.490	.470	.500	.530	.550	**.580**	.530	.540	.470	.570	.480
NATOPS	.839	.883	.872	.933	**.961**	.944	.822	.789	.878	.878	.850	.850	.856	.867	.750	.850	.783	.856	.811
PEMS-SF	**1.000**	.711	.803	.821	.780	.994	.988	.988	.988	.988	**1.000**	**1.000**	**1.000**	.988	.902	**1.000**	.988	.954	.942
PenDigits	.942	.977	.968	–	**.991**	.964	.964	.959	.960	.104	.967	.963	.943	.940	.915	.969	.969	.921	.936
PhonemeSpectra	.209	.151	.329	.315	**.374**	.212	.276	.204	.317	.265	.213	.218	.263	.252	.136	.270	.275	.303	.285
RacketSports	.803	.803	**.914**	.882	.895	.888	.836	.836	**.914**	.283	.796	.776	.882	.908	.783	.829	.849	.895	.882
SelfRegulationSCP1	.727	.775	.867	**.908**	.829	.778	.788	.720	.819	.741	.795	.744	.788	.785	.720	.795	.812	.887	.877
SelfRegulationSCP2	.550	.539	**.589**	.533	.483	.522	.478	.522	.494	.517	.439	.439	.522	.511	.511	.467	.522	.439	.439
SpokenArabicDigits	.952	.963	.985	.986	**.991**	.953	.961	.916	.982	.940	.954	.948	.958	.951	.946	.961	.958	.958	.932
StandWalkJump	.400	.200	.400	.400	.400	.133	.533	**.667**	.400	.467	.600	.600	.400	.467	.400	.400	.467	.400	.400
UWaveGestureLibrary	.872	.903	.909	**.931**	.903	.888	.900	.722	.900	.738	.912	.900	.906	.900	.759	.862	.881	.866	.881
Mean	.668	.681	.723	**.733**	.712	.686	.691	.671	.710	.635	.679	.678	.704	.704	.639	.678	.689	.710	.698
Mean rank	12.56	10.93	6.83	**5.54**	7.81	10.13	9.56	13.52	7.46	11.70	11.39	11.78	7.65	8.02	14.72	10.93	9.85	9.35	10.28

F.4 Excluded UEA Datasets

We evaluate on 27 of the 30 UEA datasets. The remaining three are excluded because one or more methods cannot run on them:

- **EigenWorms**: TS2Vec and MOMENT run out of memory.
- **PenDigits**: the series length (8) is below MiniRocket's minimum of 9 time steps.
- **InsectWingbeat**: series have variable length (2–22 steps), which several baselines do not support. TimEE also cannot run on this dataset, as its memory footprint, which grows quadratically with the in-context training set size and linearly with the number of channels, exceeds GPU memory. We do not subsample the context, since the choice of subsampling scheme would confound the result.

G Ablation: Effect of Test-Time Augmentation

All main results in Section 4.2 use the default test-time augmentation (Appendix D.2), which averages the predicted class distributions of four preprocessed views. Table 13 reports TimEE with each view alone, i.e. a single forward pass and no averaging, alongside the four-view default.

Table 13: Effect of input preprocessing on TimEE across 128 UCR datasets. Each single-member row is one forward pass with that preprocessing. The default ensembles 4 members: interpolation to 256 and 512, each on the raw and the first-differenced series.

Preprocessing	ROC AUC	Accuracy (%)
Interpolate 256	93.77	81.51
Interpolate 256 + first diff.	93.86	81.20
Interpolate 512	93.75	81.53
Interpolate 512 + first diff.	93.99	81.66
4-member ensemble (default)	**94.85**	**84.13**

The four views perform almost identically on their own (ROC AUC 93.8–94.0, accuracy 81.2–81.7%): neither differencing nor the target length is individually decisive. The gain comes from averaging across views, which adds $+0.9$ ROC AUC and $+2.5$ pp accuracy over the best single view. This comes at a $4\times$ increase in forward passes; all timings in Section 4.2 and Appendix F.1 include all four views and thus upper-bound TimEE's inference cost.