BRIDGING THE FINE-TUNING GAP IN TIME SERIES CLASSIFICATION WITH SYNTHETIC DATA AND INFERENCE STRATEGIES

Vasilii Feofanov[*]
42.com

Songkang Wen
Noah's Ark Lab

Lujia Pan
Noah's Ark Lab

Jianfeng Zhang
Noah's Ark Lab

Ievgen Redko
Noah's Ark Lab

ABSTRACT

Developing foundation models for time series classification is of high practical relevance, as such models can serve as universal feature extractors for diverse downstream tasks. Despite the promise of this approach, a substantial performance gap remained between frozen and fine-tuned encoders. In this work, we introduce a methodology that significantly strengthens linear probing performance for time series classification. Considering the Mantis architecture as a backbone, we pre-train it entirely on synthetic time series, monitoring its performance in a layer by layer, epoch by epoch fashion. We reveal that the model benefits from data scaling, but this improvement is hidden in one of the intermediate transformer layers. In addition, we propose inference strategies such as output-token aggregation and self-ensembling that allow us to achieve state-of-the-art linear probing performance on the UCR benchmark. Finally, we show that when Mantis' embeddings are fused with those of another foundation model, we match the performance of a fine-tuned encoder, providing a proof-of-concept that the gap between frozen and fine-tuned paradigms can be bridged.

Track: Research

1 INTRODUCTION

Time series classification arise in various domains including human activity recognition (Chen et al., 2025; Li et al., 2025b), power electronics (Liao et al., 2025; Li et al., 2025a), healthcare (Alchieri et al., 2025; Wong et al., 2025), finance (Lee et al., 2023), and neuroscience (Wang et al., 2024; Gnassounou et al., 2025). Nowadays, the development of time series foundation models (TSFMs) has become an active research direction. These models aim to serve as universal feature extractors for diverse downstream tasks, reducing the need for extensive labeled data and simplifying the process of model selection and tuning.

Over the past three years, a wide variety of TSFMs have been introduced. Several approaches adopt decoder-only architectures for forecasting (Cohen et al., 2025; Auer et al., 2025b; Ansari et al., 2025), while others rely on masked autoencoders (Goswami et al., 2024), adapt large language (Zhou et al., 2023; Ashok et al., 2025) or vision models (Chen et al., 2024; Roschmann et al., 2025). For TSFMs designed specifically for classification, the main goal is to learn discriminative embeddings, thereby making self-distillation (Lin et al., 2024) and contrastive objectives prevail (Albelwi, 2022). Among these, Mantis (Feofanov et al., 2025), when used as a universal feature extractor, stands out for achieving strong performance while remaining lightweight and keeping the encoder frozen.

Despite this progress, two important limitations remain: (1) the scaling benefits do not appear clear for time series foundation models, (2) the performance gap between frozen and fine-tuned encoders remain substantial. To address these issues, we introduce a new methodology built around two

[*]Correspondence to vfeofanov@42.com.

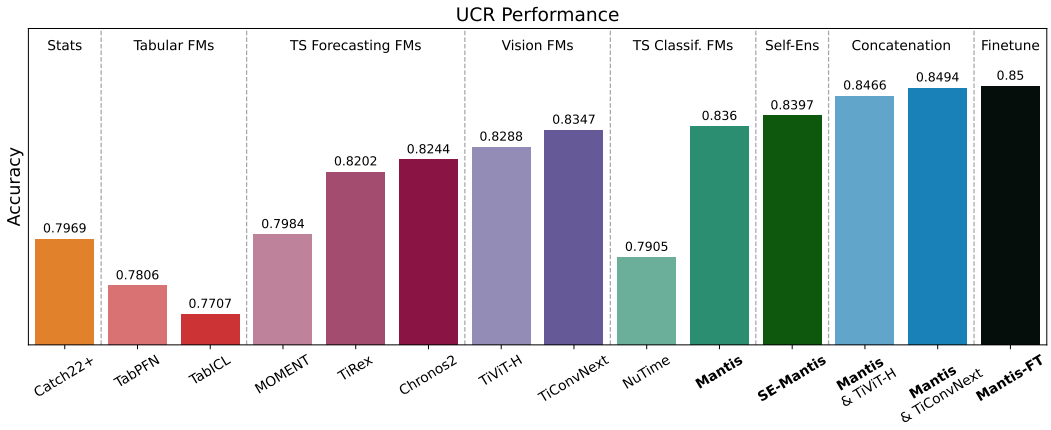


Figure 1: Final Performance on the UCR Benchmark.

key ideas: (a) following (Xie et al., 2025), we pre-train Mantis on large-scale synthetic dataset to cover a broad range of temporal patterns; by thoroughly monitoring the downstream performance evolution, we find that intermediate-layers become more discriminative than the final layer during long training while clearly benefiting from increasing the dataset size, (b) we propose a comprehensive inference-time pipeline that enhances representation quality without any additional pre-training by incorporating an improved output-token aggregation, input-perturbation self-ensembling, and cross-model embedding fusion. Together, these components substantially improve the robustness and expressiveness of the frozen encoder, allowing it to compete with the fine-tuned one.

Throughout this paper, we occasionally use the term ”zero-shot feature extraction” to describe the linear probing setup and emphasize that the encoder's weights remain completely untouched (zero-shot transfer) during the downstream task adaptation. Both linear probing and zero-shot feature extraction refer to the same frozen-encoder paradigm.

2 PRELIMINARIES

In this section, we describe our foundation model pipeline. What concerns architecture and pre-training process, we follow Mantis (Feofanov et al., 2025). For pre-training data, we use CauKer synthetic data generation algorithm (Xie et al., 2025).

Pre-training Data. CauKer algorithm generates synthetic time-series as follows: (1) multiple Gaussian process priors are generated with different mean functions and covariance kernels, (2) these priors are used as root nodes of a structural causal graph with a non-linear activation at each node. Extracting a subset of graph's nodes and repeating the experiment multiple times results in a final dataset. Xie et al. (2025) experimentally showed that this synthetic pre-training dataset yields the performance comparable with the real corpus, also avoiding any potential data leakage and improving sample efficiency. In addition, synthetic data are particularly useful for contrastive pre-training that promotes *uniformity*, evenly distributing representations (Wang & Isola, 2020). Achieving such uniformity requires high data diversity, which CauKer provides in a scalable and controllable manner.

Architecture. Mantis adapts the Transformer architecture to time-series data through a dedicated tokenization scheme that consists of three branches: (1) patch-level embeddings are extracted via convolution and mean pooling from a normalized input, (2) in the same spirit, features are derived from the first-order differential of the input to emphasize stationarity, (3) to preserve scale information, embeddings of patch-wise means and standard deviations from the raw signal are generated via scalar encoder (Lin et al., 2024). The three views are concatenated and projected into 32 tokens that are then fed to a Transformer consisting of six pre-normalized layers with multi-head self-attention. A learnable classification token is used to aggregate global information, positional encoding provides temporal order. During pre-training, a projection head is used for loss calculation, while a task-specific head is appended for fine-tuning.

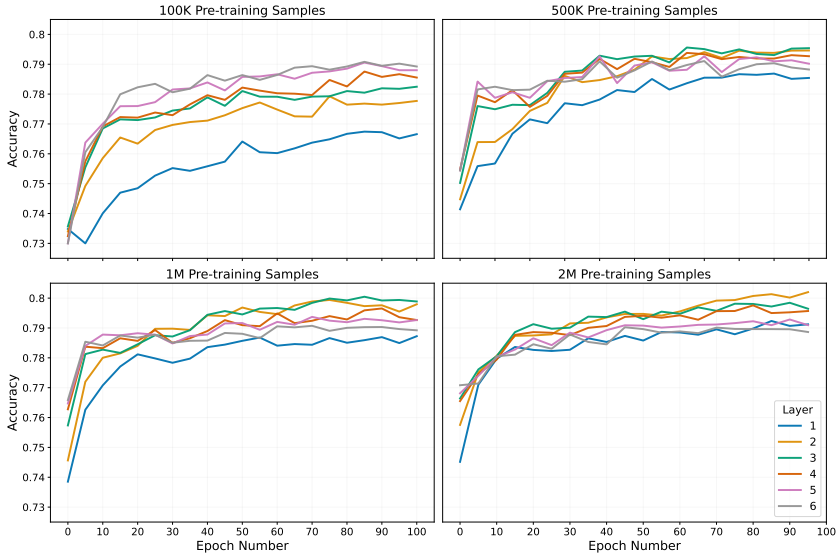


Figure 2: Layer by layer, epoch by epoch pre-training.

Contrastive Pre-training. We pre-train using a self-supervised contrastive objective that encourages similar embeddings for two augmented versions of the same time series while pushing apart representations from different series within a batch. Each input is transformed twice using randomly sampled augmentations, passed through the encoder and a projection head, and compared via cosine similarity. Training minimizes a temperature-scaled cross-entropy contrastive loss (He et al., 2020). For augmentation, the model uses Random Crop Resize (RCR), which randomly removes a small contiguous portion of a time series (up to 20%) and resizes the remaining segment back to the original length. This introduces moderate temporal distortions while preserving overall structure, encouraging invariance to slight temporal stretching or compression.

3 Key Improvements

In this section, we describe the proposed methodology. All experimental results indicate zero-shot performance in average over 128 UCR datasets (Dau et al., 2019).

3.1 Layer by Layer, Epoch by Epoch

Skean et al. (2025) showed that intermediate layers of large language models may outperform final layers on downstream tasks, while Roschmann et al. (2025) showed the efficiency of intermediate layers for cross-domain transfer. We extend this analysis further in the context of time series classification. We experimentally show that intermediate representations improve the performance of Mantis, keep a comparable performance yet reducing model size for MOMENT (Goswami et al., 2024) and NuTime (Lin et al., 2024), and enable classification for TiRex (Auer et al., 2025b) and Chronos2 (Ansari et al., 2025) (see more details in Appendix B.1).

Motivated by this observation, we investigate the evolution of layer-wise representations during pre-training. In Figure 2, we track the UCR classification performance of all six Transformer layers across pre-training 100 epochs and vary the number of pre-training examples from 100K to 2M. First, we find that the relative improvement of intermediate layers appears to be closely tied to the total number of parameter updates. Longer pre-training leads to contrastive loss overfitting so the performance of last layers does not improve (see more details in Appendix B.2).

Second, when we compare pre-training over different dataset sizes, we can observe that although the final-layer performance remains largely unchanged (or may even degrade) as the pre-training dataset grows, the *best-performing intermediate layer* improves consistently with increased data scale. In other words, **intermediate representations unlock the scaling benefits of pre-training**.

Table 1: Comparison of Model Sizes.

# of Params.	TabPFN	TabICL	MOMENT	TiRex	Chronos2	TiViT-H	TiConvNext	NuTime	Mantis
Original	7.2M	27.1M	341.2M	35.3M	119.5M	630.8M	843.4M	2.4M	4.2M
After Pruning	-	-	161.4M	16.5M	44M	276.6M	180.3M	2M	2.2M

By selecting the most informative layer, Mantis exhibits a clear and monotonic improvement in performance as the number of pre-training samples increases. Based on these findings, we perform the final pre-training using 2 million synthetic time series for 200 epochs. We provide the final layer by layer, epoch by epoch performance curve in Appendix B.2.

3.2 Inference Strategies

Apart from intermediate representations, we find other several ways to improve the performance at inference-time. We briefly describe them below, deferring other details to Appendix B.

Output-token aggregation. Although using the hidden state of the classification token as the final representation is a standard approach in Transformers, it can be suboptimal when intermediate-layer representations are exploited. In this case, we have found that non-classification tokens remain still important, so we propose a strategy that uses a concatenation of classification token with the mean of remaining tokens as the output of the model. In Section B.3, we show that the proposed aggregation strategy leads to performance improvement.

Self-Ensembling (SE). We exploit the fact that interpolating a time series to different sequence lengths changes the effective receptive field of each token, resulting in complementary representations even for the same input. We therefore generate multiple interpolated versions of the same input, encode each independently, and concatenate the resulting embeddings. We also extend this strategy to first-order differences of the input signal and combine both representations. We refer to this strategy as *Self-Ensembling (SE)*, denoting the resulting model as *SE-Mantis*. While Self-Ensembling requires multiple forward passes, this strategy remains drastically cheaper than the hundreds of forward and backward passes required for full fine-tuning. In Section B.4, we perform an ablation study to demonstrate the benefit of all components of the proposed strategy.

Cross-model Embedding Fusion. Since foundation models can be pre-trained with different objectives and architectures, their representations may capture complementary aspects of time series data. Motivated by this, we investigate whether zero-shot feature extraction performance can be further improved by combining Mantis embeddings with those of competing models pretrained on other modalities and dedicated to the forecasting task. In Section B.5, we experimentally show that Mantis benefits the most when combined with vision models.

4 Experimental Results

In this section, we compare Mantis with the proposed improvements against the following baselines, whose implementation details can be found in Appendix A: (1) **Catch22+**, which is our modification of Catch22 (Lubba et al., 2019), (2) **TabPFN** (Hollmann et al., 2022), (3) **TabICL** (Qu et al., 2025), (4) **MOMENT** (Goswami et al., 2024), (5) **TiRex** (Auer et al., 2025b), (6) **Chronos2** (Ansari et al., 2025), (7) **TiViT-H**, (8) **TiConvNext** (Roschmann et al., 2025), (9) **NuTime** (Lin et al., 2024). When relevant, we truncate model to the best intermediate layer (see Table 1 for model sizes). More details on the experimental setup can be found in Appendix A.1.

Figure 1 displays the performance of the considered models in average over 128 UCR datasets, while the complete table with results is deferred to Appendix B.6. From the results, we can see that the improved version of Mantis outperforms all the models under consideration, though modern forecasting models (TiRex, Chronos2) and adapted vision models (TiViT-H, TiConvNext) are very competitive and establish strong baselines. Catch22+, which is a set of statistical features, also establishes a good baseline, even outperforming NuTime. Tabular foundation models are competitive as well: although their average performance is rather low, their win rate is high (see Appendix B.6). This indicates that a considerable portion of datasets in UCR are nearly tabular, so their sequential

nature can simply be ignored, as it was also found by Zhang et al. (2025). In Figure 1, we also illustrate the performance of SE-Mantis, and two best cross-model fusion results, namely, combination of Mantis with TiViT-H and TiConvNext. Finally, we display the performance of Mantis when it is fine-tuned to each dataset (more details can be found in Appendix A.4). The obtained results are very encouraging: the combination of frozen Mantis and TiConvNext encoders not only improves the performance of Mantis by $1.34\,\%$, but also reaches the fine-tuned Mantis, thereby closing the zero-shot gap been present before.

5 Conclusion

In this paper, we analyzed time series classification foundation model pre-training on synthetic data and showed that by leveraging intermediate-layer representations, we do not only improve the performance, but also unlock better data scaling laws. In addition, we proposed additional inference strategies that boost the linear probing performance, allowing it to match the fine-tuning case. This proof-of-concept demonstrates that frozen features already capture the requisite information to match fine-tuning, setting a clear target for future, unified foundation models to achieve this performance without the need for model ensembling. Another promising direction for future work could be to investigate strategies that unlock model scaling laws and allow us to improve performance by training larger models.

References

Saleh Albelwi. Survey on self-supervised learning: Auxiliary pretext tasks and contrastive learning methods in imaging. *Entropy*, 24(4):551, 2022.

Leonardo Alchieri, Lino Candian, Nouran Abdalazim, Lidia Alecci, Giovanni De Felice, and Silvia Santini. Exploring generalist foundation models for time series of electrodermal activity data. In *Companion of the 2025 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, pp. 885–892, 2025.

Abdul Fatir Ansari, Oleksandr Shchur, Jaris Küken, Andreas Auer, Boran Han, Pedro Mercado, Syama Sundar Rangapuram, Huibin Shen, Lorenzo Stella, Xiyuan Zhang, et al. Chronos-2: From univariate to universal forecasting. *arXiv preprint arXiv:2510.15821*, 2025.

Arjun Ashok, Andrew Robert Williams, Vincent Zhihao Zheng, Irina Rish, Nicolas Chapados, Étienne Marcotte, Valentina Zantedeschi, and Alexandre Drouin. Beyond naïve prompting: Strategies for improved zero-shot context-aided forecasting with LLMs. *arXiv preprint arXiv:2508.09904*, 2025.

Andreas Auer, Daniel Klotz, Sebastinan Böck, and Sepp Hochreiter. Pre-trained forecasting models: Strong zero-shot feature extractors for time series classification. *arXiv preprint arXiv:2510.26777*, 2025a.

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. *arXiv preprint arXiv:2505.23719*, 2025b.

Maximilian Beck, Korbinian Pöppel, Markus Spanring, Andreas Auer, Oleksandra Prudnikova, Michael Kopp, Günter Klambauer, Johannes Brandstetter, and Sepp Hochreiter. xlstm: Extended long short-term memory. *Advances in Neural Information Processing Systems*, 37:107547–107603, 2024.

Leo Breiman. Random forests. *Machine learning*, 45:5–32, 2001.

Baiyu Chen, Wilson Wongso, Zechen Li, Yonchanok Khaokaew, Hao Xue, and Flora Salim. CO-MODO: Cross-modal video-to-imu distillation for efficient egocentric human activity recognition. *arXiv preprint arXiv:2503.07259*, 2025.

Mouxiang Chen, Lefei Shen, Zhuo Li, Xiaoyun Joy Wang, Jianling Sun, and Chenghao Liu. VisionTS: Visual masked autoencoders are free-lunch zero-shot time series forecasters. *arXiv preprint arXiv:2408.17253*, 2024.

Ben Cohen, Emaad Khwaja, Youssef Doubli, Salahidine Lemaachi, Chris Lettieri, Charles Masson, Hugo Miccinilli, Elise Ramé, Qiqi Ren, Afshin Rostamizadeh, et al. This time is different: An observability perspective on time series foundation models. *arXiv preprint arXiv:2505.14766*, 2025.

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

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. *arXiv preprint arXiv:2502.15637*, 2025.

Théo Gnassounou, Yessin Moakher, Shifeng Xie, Vasilii Feofanov, and Ievgen Redko. Leveraging generic time series foundation models for EEG classification. *arXiv preprint arXiv:2510.27522*, 2025.

Mononito Goswami, Konrad Szafer, Arjun Choudhry, Yifu Cai, Shuo Li, and Artur Dubrawski. MOMENT: A family of open time-series foundation models. *arXiv preprint arXiv:2402.03885*, 2024.

Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Guo, Mohammad Gheshlaghi Azar, et al. Bootstrap your own latent-a new approach to self-supervised learning. *Advances in neural information processing systems*, 33:21271–21284, 2020.

Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 9729–9738, 2020.

Noah Hollmann, Samuel Müller, Katharina Eggensperger, and Frank Hutter. TabPFN: A transformer that solves small tabular classification problems in a second. *arXiv preprint arXiv:2207.01848*, 2022.

Gabriel Ilharco, Mitchell Wortsman, Ross Wightman, Cade Gordon, Nicholas Carlini, Rohan Taori, Achal Dave, Vaishaal Shankar, Hongseok Namkoong, John Miller, Hannaneh Hajishirzi, Ali Farhadi, and Ludwig Schmidt. OpenCLIP, July 2021. URL https://doi.org/10.5281/zenodo.5143773.

Jean Lee, Hoyoul Luis Youn, Josiah Poon, and Soyeon Caren Han. StockEmotions: Discover investor emotions for financial sentiment analysis and multivariate time series. *arXiv preprint arXiv:2301.09279*, 2023.

Deyu Li, Xinyuan Liao, Shaowei Chen, and Shuai Zhao. Data-efficient motor condition monitoring with time series foundation models. *arXiv preprint arXiv:2511.23177*, 2025a.

Zechen Li, Baiyu Chen, Hao Xue, and Flora D Salim. ZARA: Zero-shot motion time-series analysis via knowledge and retrieval driven llm agents. *arXiv preprint arXiv:2508.04038*, 2025b.

Xinyuan Liao, Xinyue Zhang, Xing Wei, Junwei Liu, Shuai Zhao, Siqi Bu, and Yi Zhang. PE-TSFM: Self-supervised time-series learning for generalizable power converter health monitoring under unseen conditions. *arXiv preprint arXiv:2511.08250*, 2025.

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. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856. URL https://openreview.net/forum?id=TwiSBZ0p9u.

Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A ConvNet for the 2020s. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

Ilya Loshchilov, Frank Hutter, et al. Fixing weight decay regularization in Adam. *arXiv preprint arXiv:1711.05101*, 5, 2017.

Carl H Lubba, Sarab S Sethi, Philip Knaute, Simon R Schultz, Ben D Fulcher, and Nick S Jones. catch22: CAnonical Time-series CHaracteristics: Selected through highly comparative time-series analysis. *Data mining and knowledge discovery*, 33(6):1821–1852, 2019.

José Luis Morales and Jorge Nocedal. Remark on "algorithm 778: L-BFGS-B: Fortran subroutines for large-scale bound constrained optimization". *ACM Transactions on Mathematical Software (TOMS)*, 38(1):1–4, 2011.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in Python. *the Journal of machine Learning research*, 12:2825–2830, 2011.

Jingang Qu, David Holzmüller, Gaël Varoquaux, and Marine Le Morvan. TabICL: A tabular foundation model for in-context learning on large data. In *International Conference on Machine Learning*, 2025.

Alec Radford, Jong Wook Kim, Chris Hallacy, A. Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In *ICML*, 2021.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67, 2020.

Simon Roschmann, Quentin Bouniot, Vasilii Feofanov, Ievgen Redko, and Zeynep Akata. Time series representations for classification lie hidden in pretrained vision transformers. *arXiv preprint arXiv:2506.08641*, 2025. URL https://arxiv.org/abs/2506.08641.

Christoph Schuhmann, Romain Beaumont, Richard Vencu, Cade W Gordon, Ross Wightman, Mehdi Cherti, Theo Coombes, Aarush Katta, Clayton Mullis, Mitchell Wortsman, Patrick Schramowski, Srivatsa R Kundurthy, Katherine Crowson, Ludwig Schmidt, Robert Kaczmarczyk, and Jenia Jitsev. LAION-5B: An open large-scale dataset for training next generation image-text models. In *Thirty-sixth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2022. URL https://openreview.net/forum?id=M3Y74vmsMcY.

Noam Shazeer. GLU variants improve transformer. *arXiv preprint arXiv:2002.05202*, 2020.

Oscar Skean, Md Rifat Arefin, Dan Zhao, Niket Patel, Jalal Naghiyev, Yann LeCun, and Ravid Shwartz-Ziv. Layer by layer: Uncovering hidden representations in language models. *arXiv preprint arXiv:2502.02013*, 2025.

Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. RoFormer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024.

Jiquan Wang, Sha Zhao, Zhiling Luo, Yangxuan Zhou, Haiteng Jiang, Shijian Li, Tao Li, and Gang Pan. CBraMod: A criss-cross brain foundation model for EEG decoding. *arXiv preprint arXiv:2412.07236*, 2024.

Tongzhou Wang and Phillip Isola. Understanding contrastive representation learning through alignment and uniformity on the hypersphere. In *International conference on machine learning*, pp. 9929–9939. PMLR, 2020.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pp. 38–45, Online, October 2020. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/2020.emnlp-demos.6.

Sheng Wong, Ravi Shankar, Beth Albert, and Gabriel Davis Jones. Large language models surpass domain-specific architectures for antepartum electronic fetal monitoring analysis. *arXiv preprint arXiv:2509.18112*, 2025.

Shifeng Xie, Vasilii Feofanov, Marius Alonso, Ambroise Odonnat, Jianfeng Zhang, Themis Palpanas, and Ievgen Redko. CauKer: classification time series foundation models can be pretrained on synthetic data only. *arXiv preprint arXiv:2508.02879*, 2025.

Han-Jia Ye, Si-Yang Liu, and Wei-Lun Chao. A closer look at TabPFN v2: Strength, limitation, and extension. *arXiv preprint arXiv:2502.17361*, 2025. URL https://arxiv.org/abs/2502.17361.

Biao Zhang and Rico Sennrich. Root mean square layer normalization. *Advances in neural information processing systems*, 32, 2019.

Yunrui Zhang, Gustavo Batista, and Salil S Kanhere. Revisit time series classification benchmark: The impact of temporal information for classification. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pp. 193–205. Springer, 2025.

Tian Zhou, Peisong Niu, Liang Sun, Rong Jin, et al. One fits all: Power general time series analysis by pretrained LM. *Advances in neural information processing systems*, 36:43322–43355, 2023.

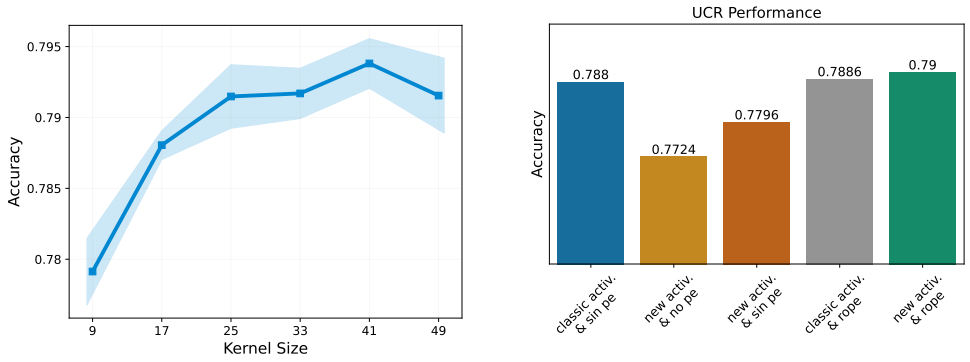
Ciyou Zhu, Richard H Byrd, Peihuang Lu, and Jorge Nocedal. Algorithm 778: L-BFGS-B: Fortran subroutines for large-scale bound-constrained optimization. *ACM Transactions on mathematical software (TOMS)*, 23(4):550–560, 1997.

A Experimental Setup

In this section, we provide more details on the evaluation setup and the chosen models. We also experimentally justify Catch22+, a strong baseline that we have proposed.

A.1 Pre-training and Inference

We follow the official Mantis pipeline (Feofanov et al., 2025) and use their code for contrastive pre-training. We also use Mantis architecture as a backbone. Before the final pre-training, we ablate some parameters of the architecture. We vary kernel size within 9, 17, 25, 33, 41, 49 and observe that performance improves monotonically up to a kernel size of 41 (Figure 3a). In addition, we tried different configurations of Transformer. Originally, Feofanov et al. (2025) used the standard sinusoidal position encoding, layer norm, GELU in the feedforward layers. We found that a Transformer with Rotary Positional Encoding (RoPE; Su et al., 2024), RMS Layer Normalization (Zhang & Sennrich, 2019) and SwiGLU activations (Shazeer, 2020) in the feedforward layers perform slightly better (Figure 3b).



(a) Ablation on convolution kernel size.

(b) Different transformer configurations.

Figure 3: Ablation on Mantis architecture.

Our zero-shot feature extraction experiment is performed as follows. For each downstream dataset, we keep the encoder frozen and use the produced embeddings together with the training labels to learn a classifier, which is then evaluated on a test set. For the choice of the classifier, there are multiple candidates: Goswami et al. (2024) used a SVM, Feofanov et al. (2025) used a Random Forest, while Roschmann et al. (2025) used Logistic Regression. We experimented with all three classifiers and found two observations. First, the best performing classifier is a combination of the Standard Scaler and Logistic Regression from the scikit-learn package (Pedregosa et al., 2011). We set the maximum number of iterations to 500 and leave the other hyperparameters to the default ones, including the solver, which is L-BFGS-B (Zhu et al., 1997; Morales & Nocedal, 2011). This classifier is used to produce the final results illustrated in Figure 1. However, Logistic Regression is practically slow which hinders extensive evaluation. This is why for our ablation experiments (Section 3 and all other experiments in Appendix) are run with Random Forest with 200 trees and maximal tree depth Breiman (2001) that is significantly faster. Overall, we have found that this choice doesn't alter the relative ranking of the studied methods.

A.2 Baselines

In this section, we give more implementation details of all baselines.

- Catch22 (Lubba et al., 2019) is a set of statistical features powerful for time series classification. In this paper, we have significantly improved this baseline by incorporating also patched statistics. We name this modification as **Catch22+** and refer to Appendix A.3 for more details and the corresponding ablation study. We use the official python implementation `pycatch22==0.4.5`.

- **TabPFN** (Hollmann et al., 2022) and **TabICL** (Qu et al., 2025) are tabular foundation models. In this case, we treat each timestamp as a feature and ignore the sequential nature of data. We use the default version of `TabPFNClassifier` from `tabpfn==2.2.1`. As TabPFN does not support more than 10 classes, we use the `ManyClassClassifier` wrapper from TabPFN Extensions (Ye et al., 2025). For TabICL, we use the official implementation with default parameters from `tabicl==0.1.3`.
- **MOMENT** (Goswami et al., 2024) is a T5-based auto-encoder (Raffel et al., 2020) pre-trained in a self-supervised way on 1.13 billion samples. Following Section 3.1, we use the output of its 10th transformer layer, which shrinks the model size from 341.2 to 161.4 million parameters. We follow Feofanov et al. (2025), using the MOMENT-large model (`d_model`=1024) and interpolate sequences to 512 instead of zero-padding as done in the original paper.
- **TiRex** (Auer et al., 2025b) a time series forecasting foundation model based on the xLSTM architecture (Beck et al., 2024). We use its 5th layer for the output, reducing the model size from 35.3 to 16.5 million parameters. We use the official implementation with default parameters from `tirex-ts==1.1.1`.
- **Chronos2** (Ansari et al., 2025) is a transformer-based time series forecasting foundation model. We use the output of its 4th layer, shrinking the model size from 119.5 to 44 million parameters. We use the official implementation with default parameters from `chronos-forecasting==2.0.0`.
- **TiViT-H** is the approach proposed by (Roschmann et al., 2025) to leverage a pre-trained vision model for time series classification. We follow their recommendations and use the 14th layer of CLIP ViT-H leading to 276.6 million parameters. **TiConvNext** leverages the pre-trained CLIP ConvNext in the same spirit. We use its 15th layer which results in 180.3 million parameters. We use the official implementation available at GitHub. For CLIP ViT-H, we use this checkpoint, while CLIP ConvNext's checkpoint can be found here (Wolf et al., 2020; Radford et al., 2021; Ilharco et al., 2021; Liu et al., 2022; Schuhmann et al., 2022).
- **NuTime** (Lin et al., 2024) is a classification TSFM based on the BYOL self-distillation pre-training (Grill et al., 2020). Their pre-training dataset is similar to the one used for pre-training of the original Mantis (1.89 million time series examples). We use its 5th layer to get embeddings, which reduces the model size from 2.4 to 2 million parameters. What it concerns implementation, we use their offical GitHub repository and follow Feofanov et al. (2025) regarding the zero-shot experiment.

As we explained in Section A.1, for all models, except TabPFN and TabICL, we use Logistic egression to perform zero-shot feature extraction.

A.3 CATCH22+

Catch22 (Lubba et al., 2019) is a manually selected set of time series statistics with a high discriminative power. Despite the authors later proposed an improvement by adding the global mean and standard deviation to the set (so-called Catch24), we have found that it can be further improved by splitting the input into non-overlapping patches (we follow Auer et al. (2025a) and set their number to 8) and computing the mean and standard deviation for each of them. Using these means and standard deviations as features for classification we further refer by Stats. Figure 4 illustrates the average accuracy of different configurations across UCR datasets. One can notice that the combination of Catch22 with Stats yields a big improvement by 3.58%, while the improvement from adding global mean and standard deviation (Catch24) is more modest (0.75%). For this reason, we have considered Catch22+ (concatenation of Catch22 and Stats) as a baseline in the main part of our paper.

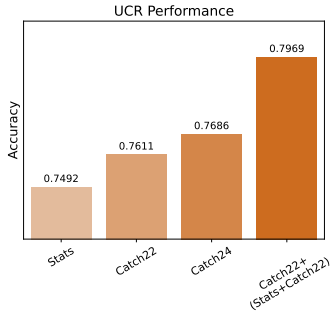


Figure 4: Catch22+ ablation study.

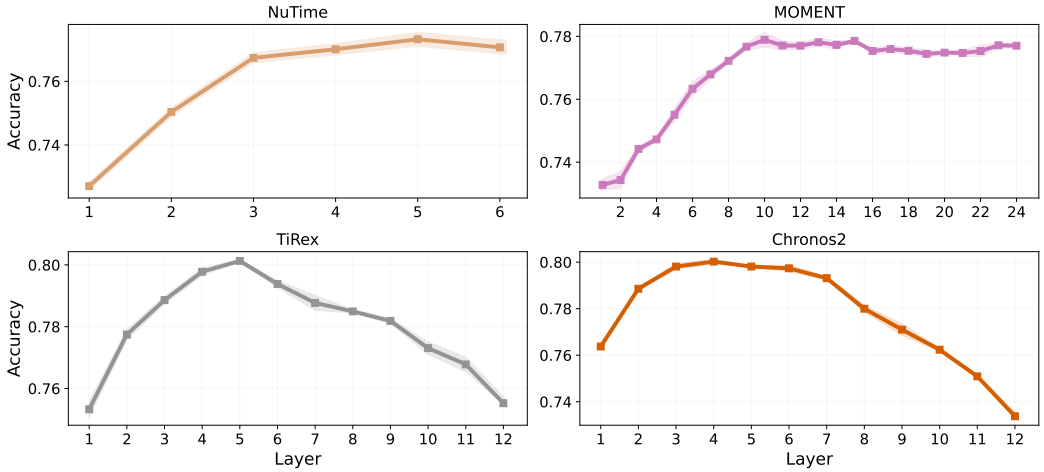


Figure 6: Layer-by-Layer for other models.

A.4 Fine-tuning

We follow Feofanov et al. (2025) and append a prediction head after an encoder, fine-tune all layers on the training data of a dataset. The prediction head is a layer normalization step followed by a linear layer. We fix a fine-tuning scheme: we minimize the cross-entropy loss for 500 epochs with a fixed batch size equal to 128, using an AdamW optimizer (Loshchilov et al., 2017) with a learning rate of $2 \cdot 10^{-4}$ and a weight decay of 0.05. We report the performance of a model at the last epoch in average over 3 experimental runs.

B Additional Experiments

In this section, we extend our experiments by showing how we can push the zero-shot feature extraction performance further. We introduce two simple strategies, namely, self-ensembling and cross-model fusion, and show the importance of the choice of the classification method.

B.1 Layer by Layer for Other Models

We have analyzed the performance of original Mantis in the layer by layer fashion and found that its performance can be significantly improved by taking embeddings from the third Transformer layer (Figure 5). In addition, for a more comprehensive comparison with baselines, we conduct a layer-by-layer analysis of most of the considered foundation models, including NuTime (Lin et al., 2024), MOMENT (Goswami et al., 2024), TiRex (Auer et al., 2025a), and Chronos2 (Ansari et al., 2025). Figure 6 summarizes the results. TiRex and Chronos2 exhibit strong discriminative power in early layers, while their final layers appear to be more specialized for forecasting. MOMENT's performance plateaus after approximately the 10th layer, suggesting that the model could be significantly compressed by truncating its final layers. NuTime benefits the least from intermediate representations, with its best-performing layer located immediately before the final one.

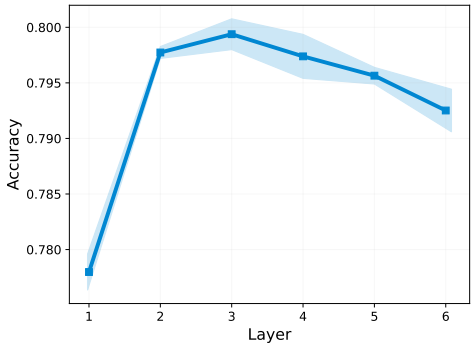


Figure 5: Mantis, layer by layer performance.

B.2 Layer by Layer, Epoch by Epoch

We revisit Figure 2. In our training setup, each epoch processes the full dataset, so increasing the dataset size directly increases the number of updates. With 100K samples, the final layer remains the strongest throughout training. In contrast, for larger datasets, intermediate layers steadily improve over time and eventually surpass the final layer. To test whether this behavior is indeed driven by the number of updates rather than dataset size per se, we pre-train Mantis on 100K samples for 1,000 epochs, matching the number of updates used for 1M samples over 100 epochs. The corresponding results (Figure 7) show an interesting pattern where the intermediate layers start to dominate in performance when increasing the number of updates. This suggests that the final layer may overfit the contrastive objective, so intermediate layers start to generalize better. However, increasing the number of epochs for the 100K dataset does not lead to further gains in overall performance. Instead, performance follows a double-descent–like behavior and converges to the same level achieved during the first 100 epochs.

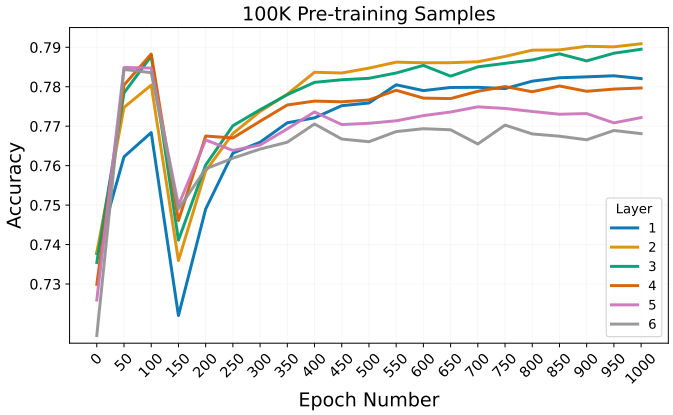


Figure 7: Layer-by-layer downstream performance (UCR) over 1000 epochs with 100K pre-training samples.

In Figure 8, we show the layer-by-layer downstream performance on UCR over 200 pre-training epochs with 2 million synthetic time series samples. These curves are used to derive final checkpoint.

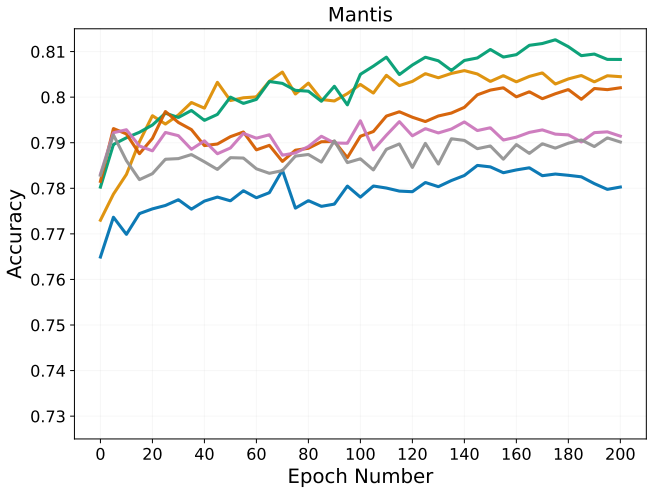


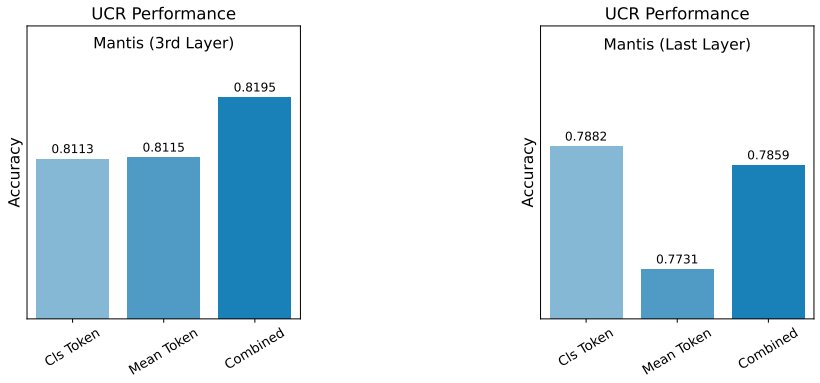
Figure 8: Final pre-training. 200 epochs with 2M pre-training samples.

B.3 Aggregation of Output Tokens

As we leverage intermediate-layer representations, we have proposed a new output-token aggregation strategy, which we validate in this section. Specifically, we evaluate three approaches to generate the final embedding:

- using only the classification token (as in the original Mantis),
- computing the mean of all tokens except the classification token,
- concatenating the classification token with the mean of the remaining tokens.

Figure 9a reports the corresponding results on the UCR benchmark. We observe that non-classification tokens encode complementary and discriminative information and should not be discarded. Moreover, concatenating the classification token with the mean token consistently yields the best performance, improving accuracy by $0.8\,\%$.



(a) Intermediate layer.

(b) Last layer.

Figure 9: Ablation study on output-token aggregation.

We note that the effectiveness of this combined aggregation strategy may depend on the depth of the Transformer, as the number of layers determines how effectively the classification token aggregates information from the remaining tokens. We have performed the same experiment for the last layer of Mantis (Figure 9b) and confirm this argument: the strategy is not beneficial anymore in this case, and the classification token is the most discriminative alone.

B.4 Self-Ensembling

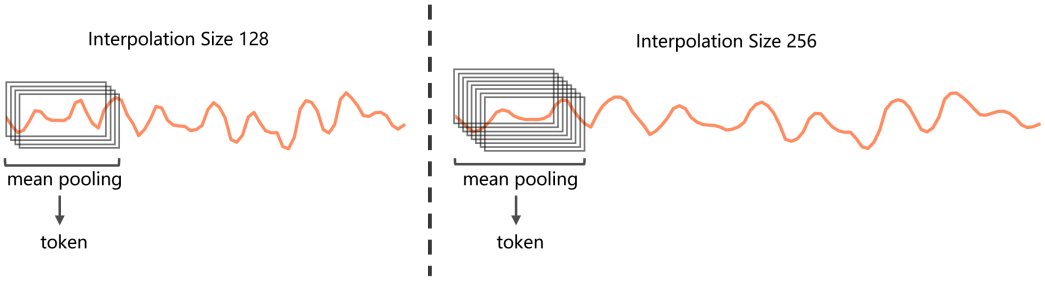


Figure 10: Motivation to look at different interpolation sizes that, in the context of our architecture, imply different degree of overlap between tokens.

First let us give more details on why different interpolation sizes lead to a different overlap between patches. In Figure 10, we illustrate how convolution + mean pooling generate one token for the same

example but interpolated to two different sizes. Since the number of tokens in our architecture is fixed, the pooling size increases with interpolation size, but not linearly. In this particular example, when the kernel size is set to 41, the receptive field of one token with interpolation sizes 128 and 256 will be $(41 + (128/32 - 1))/128 \approx 34\%$ and $(41 + (256/32 - 1))/256 \approx 19\%$, respectively. This implies that with different interpolation sizes the overlap between tokens will be different. As sequence length goes to infinity, the receptive field converges to $1/32$, i.e., to the case when the tokens are non-overlapping. Conversely, decreasing the sequence length makes tokens more and more overlapped.

We perform an ablation study over self-ensembling comparing the 4 following methods: (a) interpolate time series solely to 512 as we did before, (b) create 4 series by interpolating the input to 128, 256, 512 and 1024, pass all of them through the encoder and then concatenate all the embeddings, (c) perform the same multiple interpolations but for the first-order difference of the input, (d) concatenate the embeddings of (b) and (c). In Figure 11, the experimental results are displayed. One can see that multiple interpolations improve the performance Mantis by 0.55% on the UCR benchmark. Although the embeddings of the first-order difference are individually weak, their incorporation further improve the performance by 0.29%. It is an interesting observation since the feature extraction from the first-order differential is already incorporated to the architecture of Mantis, yet we can still benefit from this strategy.

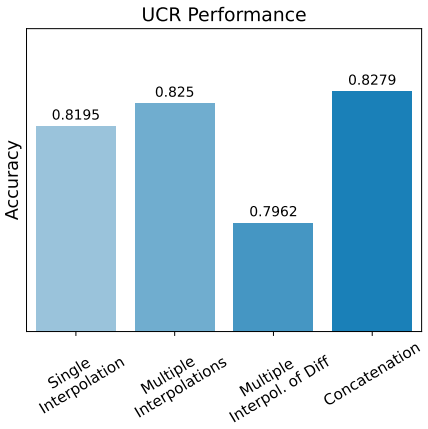


Figure 11: Self-Ensembling Ablation Study.

B.5 Cross-model Embedding Fusion

The experimental results for combining the embeddings of different pretrained models are reported in Figure 12. Similar to (Roschmann et al., 2025), we observe that model combination is consistently beneficial, even when Mantis is combined with hand-crafted statistical features (Catch22+). The smallest gains are obtained when combining Mantis with MOMENT and NuTime, which may be explained by limited representational complementarity in the case of NuTime, or by lower standalone performance. The highest overall performance is achieved when Mantis is combined with TiConvNeXt.

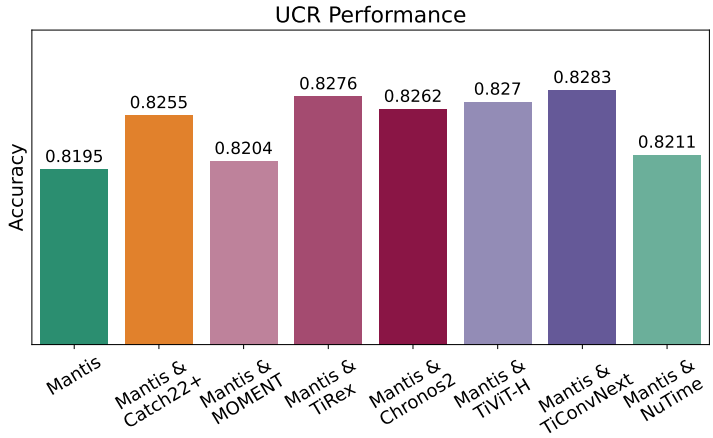


Figure 12: Combination of Mantis with other models.

B.6 Complete Results

Below we provide the complete results for Figure 1.

Table 2: Performance Comparison on UCR benchmark. First Part.

	Catch22+	TabPFN	TabICL	MOMENT	TiRex	Chronos2	TiViT-H	TiConvNext	NuTime	Mantis	SE-Mantis	Mantis TiViT-H	Mantis TiConvNext	Mantis-FT
ACSF1	$0.8233_{\pm 0.021}$	0.8	0.81	0.75	0.75	**0.86**	0.83	0.83	0.7	0.76	0.77	0.81	0.82	$0.77_{\pm 0.01}$
Adiac	$0.734_{\pm 0.007}$	0.8031	0.8031	0.7775	0.7826	0.8286	0.7187	0.7059	0.8005	0.8363	**0.8465**	0.7519	0.7749	$0.8321_{\pm 0.007}$
AllGestureWiimoteX	$0.5957_{\pm 0.01}$	0.6229	0.5043	0.6614	0.7014	0.6843	0.67	0.6943	0.6071	0.73	0.6871	0.7229	0.7357	$\mathbf{0.779}_{\pm 0.006}$
AllGestureWiimoteY	$0.6319_{\pm 0.016}$	0.6329	0.5114	0.7029	0.7314	0.7043	0.7371	0.7229	0.6529	0.7471	0.7271	0.7557	0.7486	$\mathbf{0.7905}_{\pm 0.004}$
AllGestureWiimoteZ	$0.5462_{\pm 0.005}$	0.5329	0.4529	0.6057	0.6343	0.66	0.6771	0.6586	0.5414	0.6814	0.6357	0.6886	0.7043	$\mathbf{0.7505}_{\pm 0.017}$
ArrowHead	$0.741_{\pm 0.012}$	0.7543	0.7429	0.7771	0.7829	0.8514	0.8229	0.8229	0.7257	0.8514	0.8514	**0.88**	0.8457	$0.8457_{\pm 0.006}$
BME	$\mathbf{1.0}_{\pm 0.0}$	**1.0**	0.98	0.9933	0.9933	**1.0**	**1.0**	**1.0**	0.9667	**1.0**	**1.0**	**1.0**	**1.0**	$0.9933_{\pm 0.007}$
Beef	$0.6889_{\pm 0.051}$	0.8	0.7667	0.8	**0.9333**	0.7	0.7667	0.8333	0.7333	0.7333	0.8	0.7667	0.8333	$0.7778_{\pm 0.069}$
BeetleFly	$0.7833_{\pm 0.029}$	0.9	0.8	**0.95**	0.9	0.8	0.85	**0.95**	0.7	0.85	**0.95**	**0.95**	**0.95**	$0.8833_{\pm 0.029}$
BirdChicken	$0.85_{\pm 0.0}$	0.85	0.75	**1.0**	0.9	0.9	0.9	0.95	**1.0**	0.9	0.9	0.9	0.9	$0.9_{\pm 0.0}$
CBF	$0.9763_{\pm 0.002}$	0.9133	0.9244	0.9889	0.9967	**1.0**	0.9989	0.9989	0.99	0.9967	0.9944	0.9989	**1.0**	$0.9959_{\pm 0.003}$
Car	$0.7556_{\pm 0.025}$	0.7833	0.8167	**0.9**	0.8	0.85	0.8167	0.85	0.7333	0.8167	**0.9**	**0.9**	**0.9**	$0.8667_{\pm 0.0}$
Chinatown	$0.9796_{\pm 0.003}$	**0.9854**	0.9796	0.9825	0.9825	**0.9854**	0.9679	0.9388	0.9621	0.9621	0.9534	0.9592	0.9504	$0.9738_{\pm 0.008}$
ChlorineConcentration	$0.6682_{\pm 0.001}$	0.95	**0.9773**	0.7789	0.8018	0.7672	0.7622	0.7771	0.6086	0.7115	0.7867	0.7914	0.8026	$0.818_{\pm 0.009}$
CinCECGTorso	$0.8872_{\pm 0.013}$	0.8341	0.8225	0.7681	**0.9565**	0.8986	0.9355	0.942	0.8551	0.8507	0.9268	0.9507	0.9449	$0.83_{\pm 0.007}$
Coffee	$0.9881_{\pm 0.021}$	0.9643	**1.0**	0.9643	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	$\mathbf{1.0}_{\pm 0.0}$
Computers	$0.7347_{\pm 0.006}$	0.62	0.656	0.612	0.8	0.768	0.764	**0.828**	0.788	0.704	0.724	0.756	0.804	$0.716_{\pm 0.011}$
CricketX	$0.6974_{\pm 0.01}$	0.6667	0.6641	0.7436	0.7	0.7564	0.759	0.7487	0.7051	0.7821	0.8	0.7769	0.7718	$\mathbf{0.8256}_{\pm 0.003}$
CricketY	$0.6923_{\pm 0.008}$	0.7	0.6333	0.7077	0.7538	0.7359	0.7872	0.7462	0.6897	0.8154	0.8179	0.8231	0.8026	$\mathbf{0.8427}_{\pm 0.006}$
CricketZ	$0.7427_{\pm 0.008}$	0.6718	0.6692	0.7436	0.7333	0.7231	0.7974	0.759	0.659	0.8128	0.8179	0.8103	0.7846	$\mathbf{0.8556}_{\pm 0.013}$
Crop	$0.7523_{\pm 0.001}$	0.7989	**0.812**	0.7234	0.7029	0.7163	0.6867	0.6737	0.6895	0.7348	0.732	0.7231	0.7171	$0.7634_{\pm 0.004}$
DiatomSizeReduction	$0.9401_{\pm 0.008}$	0.9608	0.951	0.9412	0.9281	0.9477	0.9673	0.9673	0.915	0.9281	0.9281	0.9575	**0.9706**	$0.9412_{\pm 0.018}$
DistalPhalanxOutlineAgeGroup	$0.717_{\pm 0.004}$	0.7626	0.7626	0.6691	0.7482	0.741	0.7122	0.7266	0.705	**0.777**	0.7122	0.705	0.7338	$0.7362_{\pm 0.018}$
DistalPhalanxOutlineCorrect	$\mathbf{0.7911}_{\pm 0.002}$	0.7826	0.7754	0.7536	0.75	0.7645	0.7536	0.7681	0.7428	0.7681	0.7536	0.75	0.7681	$0.7899_{\pm 0.016}$
DistalPhalanxTW	$0.6475_{\pm 0.019}$	0.6978	0.6835	0.6259	0.6547	0.6547	0.6691	0.6547	0.6835	0.6835	**0.705**	0.6691	0.6763	$0.6547_{\pm 0.019}$
DodgerLoopDay	$0.6417_{\pm 0.052}$	0.6125	**0.725**	0.4375	0.55	0.525	0.5	0.5625	0.55	0.5	0.525	0.5125	0.55	$0.5333_{\pm 0.007}$
DodgerLoopGame	$0.8333_{\pm 0.007}$	0.7899	0.7971	0.8406	0.7899	0.8913	0.8406	0.8406	0.8188	0.8623	0.8551	0.8188	0.8841	$\mathbf{0.901}_{\pm 0.022}$
DodgerLoopWeekend	$\mathbf{0.9855}_{\pm 0.0}$	**0.9855**	0.9783	**0.9855**	0.9565	0.9638	0.9493	0.913	0.9638	0.971	0.971	**0.9855**	0.9565	$0.9758_{\pm 0.004}$
ECG200	$0.85_{\pm 0.017}$	0.89	0.88	**0.9**	0.83	0.84	0.86	0.86	0.85	0.88	0.87	0.85	0.87	$0.88_{\pm 0.017}$
ECG5000	$0.9398_{\pm 0.001}$	0.942	**0.9447**	0.9333	0.9391	0.9342	0.9378	0.9311	0.9193	0.9353	0.9329	0.9371	0.9289	$0.9404_{\pm 0.002}$
ECGFiveDays	$0.7975_{\pm 0.013}$	0.9245	0.9826	0.9698	0.9756	0.9907	0.9791	0.9895	0.899	0.993	**0.9977**	0.9895	0.9895	$0.9853_{\pm 0.007}$
EOGHorizontalSignal	$0.5948_{\pm 0.004}$	0.5276	0.5	0.5635	0.5718	0.5884	0.5967	0.5746	0.5331	0.5691	0.605	0.6381	0.605	$\mathbf{0.6464}_{\pm 0.022}$
EOGVerticalSignal	$0.5092_{\pm 0.002}$	0.489	0.4337	0.5	0.4171	0.4392	0.4724	0.4254	0.3315	0.5138	0.4807	0.489	0.4448	$\mathbf{0.5451}_{\pm 0.008}$
Earthquakes	$\mathbf{0.7506}_{\pm 0.008}$	0.7482	0.7482	0.705	0.6906	0.7482	0.705	0.6978	0.7338	0.7194	0.7122	0.6978	0.7122	$0.6835_{\pm 0.019}$
ElectricDevices	$0.7396_{\pm 0.003}$	0.7025	0.6614	0.7404	0.6709	0.7168	0.7635	**0.7764**	0.6786	0.7175	0.7257	0.7636	0.7679	$0.7505_{\pm 0.009}$
EthanolLevel	$0.38_{\pm 0.013}$	**0.848**	0.694	0.682	0.508	0.654	0.584	0.58	0.52	0.592	0.542	0.592	0.574	$0.7947_{\pm 0.005}$
FaceAll	$0.7682_{\pm 0.028}$	**0.8077**	0.771	0.7787	0.7876	0.7533	0.7521	0.7367	0.7207	0.7574	0.7568	0.7562	0.7444	$0.7964_{\pm 0.003}$
FaceFour	$0.8977_{\pm 0.02}$	0.9091	0.8864	0.875	0.9545	0.9091	0.875	0.9205	0.9659	**0.9886**	0.9432	0.9659	0.9659	$0.9697_{\pm 0.013}$
FacesUCR	$0.8558_{\pm 0.004}$	0.8766	0.8771	0.8912	0.8776	0.8576	0.8751	0.8561	0.8195	0.9117	0.9083	0.9088	0.898	$\mathbf{0.9416}_{\pm 0.004}$
FiftyWords	$0.726_{\pm 0.006}$	0.7385	0.7165	0.7758	0.7231	0.7956	0.7868	0.7714	0.7165	0.8198	0.8088	0.8264	0.8088	$\mathbf{0.8432}_{\pm 0.01}$
Fish	$0.7619_{\pm 0.013}$	0.88	0.8857	0.96	0.9029	0.9429	0.9486	0.9714	0.9143	0.96	0.9714	0.9657	**0.9829**	$\mathbf{0.9829}_{\pm 0.0}$
FordA	$0.9101_{\pm 0.004}$	0.897	0.8758	0.9061	0.947	0.928	0.9136	0.9227	0.9061	0.9432	0.9462	0.9242	0.9318	$\mathbf{0.95}_{\pm 0.005}$
FordB	$0.7292_{\pm 0.007}$	0.7556	0.7136	0.7556	0.816	0.8074	0.7963	0.779	0.7654	0.8099	0.8259	0.8136	0.8111	$\mathbf{0.8358}_{\pm 0.002}$
FreezerRegularTrain	$\mathbf{0.9996}_{\pm 0.0}$	0.9986	0.9877	0.9881	0.9912	0.9951	0.9965	0.9975	0.994	0.9961	0.9954	0.9972	0.9982	$0.9946_{\pm 0.0}$
FreezerSmallTrain	$0.9251_{\pm 0.002}$	0.8933	0.8098	0.8186	0.8912	0.9688	0.9839	0.9902	0.9863	0.9849	**0.9905**	0.9891	0.9895	$0.9473_{\pm 0.023}$
Fungi	$0.9229_{\pm 0.041}$	0.8656	0.7849	**1.0**	0.9516	0.9462	0.9785	0.9946	0.7473	0.9731	0.9946	0.9839	**1.0**	$0.9409_{\pm 0.023}$
GestureMidAirD1	$0.6795_{\pm 0.036}$	0.6231	0.6769	0.6846	0.6923	0.7308	0.7538	0.7615	0.6846	0.7769	0.7692	0.7769	0.7538	$\mathbf{0.7846}_{\pm 0.013}$
GestureMidAirD2	$0.6205_{\pm 0.016}$	0.5615	0.5846	0.5769	0.6154	0.7	0.6615	0.6769	0.5615	0.6462	0.6231	0.7	0.6923	$\mathbf{0.7026}_{\pm 0.025}$
GestureMidAirD3	$0.3923_{\pm 0.031}$	0.3692	0.3615	0.3692	0.4077	0.3538	0.4692	0.4923	0.4462	0.4462	0.4308	0.4385	**0.5**	$0.4641_{\pm 0.012}$
GesturePebbleZ1	$0.8779_{\pm 0.012}$	0.8488	0.8779	0.8953	0.8837	0.9012	0.907	0.8547	0.8953	0.9302	0.9302	0.936	0.9302	$\mathbf{0.9457}_{\pm 0.009}$
GesturePebbleZ2	$0.7384_{\pm 0.004}$	0.7848	0.7532	0.8924	0.8418	0.8671	0.8544	0.8165	0.7722	0.9051	0.8924	**0.9114**	0.8797	$0.8903_{\pm 0.007}$
GunPoint	$0.9467_{\pm 0.018}$	0.9667	0.9533	**1.0**	0.98	0.9933	0.9933	0.9933	0.9733	0.98	0.98	**1.0**	0.9933	$0.9933_{\pm 0.007}$
GunPointAgeSpan	$0.9884_{\pm 0.002}$	0.9905	0.9842	0.9778	0.9905	0.9905	0.9937	0.9873	0.9684	0.9937	0.9937	**0.9968**	0.9905	$\mathbf{0.9968}_{\pm 0.0}$
GunPointMaleVersusFemale	$0.9937_{\pm 0.0}$	0.9968	**1.0**	0.9968	0.9937	0.9937	**1.0**	0.9968	0.9873	**1.0**	**1.0**	**1.0**	**1.0**	$0.9968_{\pm 0.0}$
GunPointOldVersusYoung	$\mathbf{1.0}_{\pm 0.0}$	**1.0**	**1.0**	0.927	0.9651	0.9778	0.9873	0.9937	**1.0**	0.9968	0.9968	0.9968	0.9968	$0.9968_{\pm 0.0}$
Ham	$0.6063_{\pm 0.005}$	**0.7429**	0.7238	0.6571	0.7048	0.6952	**0.7429**	0.7238	0.7333	0.7048	0.6667	**0.7429**	0.7333	$0.6635_{\pm 0.02}$
HandOutlines	$0.9036_{\pm 0.004}$	0.9162	0.927	0.9405	0.8757	0.9378	**0.9432**	0.9216	0.8865	0.9297	0.9108	0.9378	0.9135	$0.9306_{\pm 0.004}$
Haptics	$0.4838_{\pm 0.015}$	0.4708	0.461	0.4903	0.4773	0.5032	0.4935	0.5227	0.4253	0.5032	0.5195	0.5325	0.5519	$\mathbf{0.5714}_{\pm 0.017}$
Herring	$0.5208_{\pm 0.024}$	0.5938	0.6406	0.6406	0.6094	0.5156	0.5312	0.6094	0.5312	**0.6875**	0.6562	0.6562	0.6406	$0.6198_{\pm 0.009}$
HouseTwenty	$0.9692_{\pm 0.005}$	0.8487	0.7563	0.9328	**0.9832**	0.9748	**0.9832**	**0.9832**	0.8992	0.9328	0.9748	0.9748	0.9748	$0.9552_{\pm 0.005}$
InlineSkate	$0.3891_{\pm 0.005}$	0.3345	0.3436	0.3636	0.4382	0.4364	0.3982	0.4473	0.3564	0.3909	0.4309	0.4255	**0.46**	$0.4473_{\pm 0.026}$
InsectEPGRegularTrain	$\mathbf{1.0}_{\pm 0.0}$	**1.0**	**1.0**	0.9478	0.996	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	$\mathbf{1.0}_{\pm 0.0}$
InsectEPGSmallTrain	$\mathbf{1.0}_{\pm 0.0}$	**1.0**	**1.0**	0.8353	0.9558	0.9197	0.988	0.988	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	$\mathbf{1.0}_{\pm 0.0}$
InsectWingbeatSound	$0.629_{\pm 0.003}$	**0.6672**	0.6556	0.6212	0.6253	0.6313	0.551	0.5652	0.5101	0.596	0.5793	0.5899	0.6081	$0.6032_{\pm 0.008}$

Table 3: Performance Comparison on UCR benchmark. Second Part.

	Catch22+	TabPFN	TabICL	MOMENT	TiRex	Chronos2	TiViT-H	TiConvNext	NuTime	Mantis	SE-Mantis	Mantis TiViT-H	Mantis TiConvNext	Mantis-FT
ItalyPowerDemand	$0.9537_{\pm 0.002}$	**0.9699**	0.9631	0.9543	0.9602	0.964	0.9407	0.9378	0.9116	0.9456	0.9475	0.9427	0.9446	$0.9517_{\pm 0.001}$
LargeKitchenAppliances	$0.8169_{\pm 0.009}$	0.6373	0.696	0.76	0.8293	0.8613	0.872	**0.8933**	0.7707	0.816	0.816	0.8507	0.888	$0.8596_{\pm 0.008}$
Lightning2	$0.7322_{\pm 0.009}$	0.6721	0.7049	0.7869	0.7377	0.7869	0.8197	0.8361	0.7377	0.7869	0.7705	**0.8525**	0.8033	$0.847_{\pm 0.025}$
Lightning7	$0.7671_{\pm 0.014}$	0.6986	0.7397	0.7397	0.8082	0.7945	0.8493	0.7671	0.7808	0.863	0.8219	**0.8767**	0.7945	$0.7717_{\pm 0.032}$
Mallat	$0.9555_{\pm 0.009}$	0.9689	0.9484	0.9186	0.9548	0.9467	**0.9765**	0.9642	0.8486	0.9198	0.9663	0.974	0.9697	$0.9512_{\pm 0.021}$
Meat	$0.9222_{\pm 0.01}$	0.9833	0.9333	**1.0**	0.9	0.9333	0.8167	0.85	0.9333	0.9667	0.8833	0.8833	0.8833	$0.8833_{\pm 0.033}$
MedicalImages	$0.7737_{\pm 0.005}$	0.7947	0.8079	0.7461	0.725	0.7474	0.7316	0.7513	0.7092	0.7711	0.7592	0.7566	0.7711	**0.8224**$_{\pm 0.003}$
MelbournePedestrian	$0.9597_{\pm 0.0}$	**0.9803**	**0.9803**	0.8954	0.8938	0.9049	0.8626	0.87	0.9332	0.9582	0.9537	0.9377	0.9426	$0.9641_{\pm 0.003}$
MiddlePhalanxOutlineAgeGroup	$0.5952_{\pm 0.004}$	0.6234	**0.6299**	0.5065	0.487	0.5519	0.5584	0.526	0.5519	0.5455	0.4675	0.539	0.539	$0.5498_{\pm 0.023}$
MiddlePhalanxOutlineCorrect	$0.811_{\pm 0.012}$	**0.8522**	0.8351	0.8076	0.8076	0.8351	0.7938	0.8179	0.7491	0.8419	0.811	0.8007	0.8144	$0.819_{\pm 0.022}$
MiddlePhalanxTW	$0.5844_{\pm 0.006}$	0.6169	**0.6234**	0.5455	0.5	0.5	0.5195	0.4805	0.4416	0.513	0.513	0.5195	0.5	$0.4935_{\pm 0.022}$
MixedShapesRegularTrain	$0.9306_{\pm 0.003}$	0.9344	0.9299	0.9357	0.9699	0.9666	0.9781	0.9781	0.9365	0.9744	0.974	0.9781	**0.9802**	$0.9739_{\pm 0.003}$
MixedShapesSmallTrain	$0.8827_{\pm 0.002}$	0.8293	0.8767	0.8899	0.9365	0.939	0.9501	0.9542	0.9208	0.946	**0.9625**	0.9567	0.9579	$0.9372_{\pm 0.002}$
MoteStrain	$0.8818_{\pm 0.015}$	0.889	0.8794	0.905	0.9241	0.9257	0.9161	0.9065	**0.9553**	0.9393	0.9545	0.9481	0.9497	$0.93_{\pm 0.004}$
NonInvasiveFetalECGThorax1	$0.8877_{\pm 0.004}$	0.941	0.9272	0.9237	0.8992	0.8845	0.9003	0.8911	0.8545	0.9293	0.9379	0.9298	0.9196	**0.943**$_{\pm 0.004}$
NonInvasiveFetalECGThorax2	$0.9091_{\pm 0.0}$	0.9476	0.9405	0.9369	0.9033	0.913	0.9237	0.9226	0.9028	0.9364	0.9481	0.9344	0.9293	**0.953**$_{\pm 0.001}$
OSULeaf	$0.6832_{\pm 0.009}$	0.5661	0.595	0.7934	0.938	0.938	0.9835	**0.9917**	0.8595	0.9628	0.9835	0.9876	0.9835	$0.9807_{\pm 0.002}$
OliveOil	$0.8444_{\pm 0.019}$	**0.9333**	0.9	0.9	**0.9333**	0.8333	0.7333	0.9	0.7667	0.8667	0.8333	0.8	0.9	$0.6889_{\pm 0.019}$
PLAID	$0.8752_{\pm 0.009}$	0.7896	0.5661	0.7858	0.8696	0.8864	0.9181	0.9311	0.7561	0.8529	0.8417	**0.9385**	0.9348	$0.9143_{\pm 0.005}$
PhalangesOutlinesCorrect	$0.8252_{\pm 0.003}$	0.8403	**0.8613**	0.8135	0.7925	0.8228	0.7995	0.7832	0.7471	0.7949	0.8054	0.7995	0.7972	$0.8287_{\pm 0.008}$
Phoneme	$0.3216_{\pm 0.005}$	0.1097	0.1361	0.3001	0.3898	**0.4167**	0.3956	0.3972	0.2758	0.355	0.4109	0.4088	0.4014	$0.3557_{\pm 0.006}$
PickupGestureWiimoteZ	$0.6933_{\pm 0.012}$	0.76	0.74	0.7	0.84	0.66	**0.92**	88	0.64	0.76	0.84	**0.92**	86	$78_{\pm 0.02}$
PigAirwayPressure	$0.2372_{\pm 0.003}$	0.0192	0.1538	0.1106	0.4183	0.4183	0.6298	**0.7452**	0.4519	0.5769	0.5625	0.6731	**0.7452**	$0.7083_{\pm 0.02}$
PigArtPressure	$0.891_{\pm 0.007}$	0.0337	0.2548	0.5481	0.8894	0.8413	0.8894	0.9327	0.9423	0.9087	0.9327	0.9471	**0.9663**	$0.9295_{\pm 0.007}$
PigCVP	$0.5128_{\pm 0.011}$	0.0192	0.1731	0.5337	0.8654	0.7885	0.7933	0.8317	0.8846	**0.9375**	0.9231	0.8798	0.9038	$0.9199_{\pm 0.007}$
Plane	**1.0**$_{\pm 0.0}$	0.9905	0.9905	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**$_{\pm 0.0}$
PowerCons	$0.9044_{\pm 0.0}$	**1.0**	**1.0**	0.9556	0.9389	0.9333	0.9056	0.9222	0.9722	0.9722	0.9556	0.9667	0.9556	$0.9648_{\pm 0.012}$
ProximalPhalanxOutlineAgeGroup	$0.8472_{\pm 0.007}$	**0.8585**	0.839	0.8293	0.8488	0.8341	0.8293	0.8293	0.8537	0.839	0.8341	0.8293	0.8293	$0.8439_{\pm 0.01}$
ProximalPhalanxOutlineCorrect	$0.8648_{\pm 0.012}$	0.9038	**0.9244**	0.8866	0.8866	0.8832	0.8729	0.8625	0.8385	0.8763	0.9107	0.8729	0.8557	$0.9049_{\pm 0.002}$
ProximalPhalanxTW	$0.8033_{\pm 0.007}$	0.8098	**0.8293**	0.7951	0.761	0.7561	0.761	0.7463	0.7854	0.761	0.7561	0.7561	0.761	$0.7431_{\pm 0.012}$
RefrigerationDevices	$0.5493_{\pm 0.023}$	0.504	0.4933	0.5147	0.5413	0.568	0.576	**0.5973**	0.5627	0.5253	0.56	0.584	0.5813	$0.552_{\pm 0.019}$
Rock	$0.62_{\pm 0.02}$	0.76	0.64	0.9	0.88	0.9	**0.96**	0.92	0.76	0.84	0.9	0.92	0.92	$0.8933_{\pm 0.042}$
ScreenType	$0.5271_{\pm 0.007}$	0.4187	0.4107	0.392	0.528	0.5387	0.536	0.4987	0.4693	0.5093	0.496	**0.544**	0.5253	$0.5218_{\pm 0.011}$
SemgHandGenderCh2	$0.9272_{\pm 0.003}$	**0.9467**	0.8867	0.765	0.8583	0.8883	0.9017	0.91	0.8617	0.8967	0.93	0.9217	0.9317	$0.9261_{\pm 0.011}$
SemgHandMovementCh2	**0.8615**$_{\pm 0.007}$	0.7711	0.5689	0.4444	0.52	0.5756	0.56	0.5911	0.6244	0.6244	0.6133	0.6467	0.6578	$0.7978_{\pm 0.029}$
SemgHandSubjectCh2	$0.8837_{\pm 0.007}$	**0.9356**	0.8333	0.7089	0.8133	0.8444	0.8644	0.8489	0.7556	0.8533	0.88	0.8933	0.9022	$0.8948_{\pm 0.008}$
ShakeGestureWiimoteZ	$0.8467_{\pm 0.031}$	82	0.74	0.84	0.88	0.92	0.84	86	0.9	0.92	0.9	0.88	88	**0.9533**$_{\pm 0.031}$
ShapeletSim	$0.9704_{\pm 0.008}$	0.4778	0.5056	0.9667	0.9667	**1.0**	**1.0**	**1.0**	0.9278	0.9778	0.9944	**1.0**	**1.0**	$0.9815_{\pm 0.014}$
ShapesAll	$0.8206_{\pm 0.002}$	0.8017	0.7917	0.8483	0.8633	0.8967	0.9117	0.8967	0.8717	0.9017	0.9167	**0.925**	0.9167	$0.9139_{\pm 0.01}$
SmallKitchenAppliances	**0.8302**$_{\pm 0.008}$	0.7867	0.7627	0.6107	0.8133	0.8107	0.8213	0.8267	0.8293	0.7867	0.7893	0.8	0.8267	$0.816_{\pm 0.003}$
SmoothSubspace	$0.9867_{\pm 0.007}$	**1.0**	**1.0**	98	0.9667	0.9667	0.9667	98	0.9467	0.9733	0.96	0.9667	0.9667	$0.9733_{\pm 0.0}$
SonyAIBORobotSurface1	$0.8397_{\pm 0.002}$	0.772	0.6722	0.8253	**0.9368**	0.8236	0.8636	0.8502	0.8153	0.8336	0.8952	0.8652	0.8552	$0.8907_{\pm 0.002}$
SonyAIBORobotSurface2	$0.8835_{\pm 0.021}$	0.809	0.8279	0.8909	0.8909	0.8846	0.9381	0.9454	0.8835	0.937	**0.9507**	0.9486	0.9496	$0.9307_{\pm 0.019}$
StarLightCurves	$0.9702_{\pm 0.001}$	0.9732	**0.9718**	0.9641	0.9739	0.9745	0.974	0.9716	0.9745	0.9733	0.9734	0.9751	0.9726	**0.978**$_{\pm 0.001}$
Strawberry	$0.9333_{\pm 0.003}$	0.9811	**0.9838**	0.9622	0.9649	0.973	0.9568	0.9541	0.9595	0.9622	0.9622	0.9676	0.9541	$0.9676_{\pm 0.005}$
SwedishLeaf	$0.9115_{\pm 0.002}$	0.9504	0.9456	0.9312	0.9536	0.9472	0.9584	0.9456	0.9392	**0.9728**	0.9664	0.968	0.9584	$0.9685_{\pm 0.002}$
Symbols	$0.9618_{\pm 0.005}$	0.8824	0.8945	0.9558	0.9668	0.9839	0.9859	0.9859	0.9688	0.9769	0.9839	0.9879	**0.9889**	$0.9792_{\pm 0.003}$
SyntheticControl	$0.9922_{\pm 0.002}$	0.99	0.9833	0.9767	0.9933	0.9967	**1.0**	**1.0**	0.9733	0.99	0.9867	**1.0**	**1.0**	$0.9989_{\pm 0.002}$
ToeSegmentation1	$0.8553_{\pm 0.016}$	0.5746	0.6667	0.9474	0.9693	0.9167	0.9386	0.9035	0.8421	0.9649	0.9737	0.9561	0.9693	**0.9795**$_{\pm 0.013}$
ToeSegmentation2	$0.7846_{\pm 0.013}$	0.6538	0.8154	0.9077	0.9154	0.8615	0.9	0.8769	0.8692	**0.9308**	0.9231	0.9154	0.9231	$0.9179_{\pm 0.012}$
Trace	**1.0**$_{\pm 0.0}$	0.91	0.98	0.99	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**$_{\pm 0.0}$
TwoLeadECG	$0.8584_{\pm 0.015}$	0.9508	0.9254	0.9921	0.9851	0.993	**1.0**	0.993	0.9622	0.9991	0.9982	**1.0**	0.9956	$0.9988_{\pm 0.001}$
TwoPatterns	$0.9935_{\pm 0.001}$	0.995	0.9032	0.9918	0.996	0.9948	0.998	0.999	0.9415	0.997	0.9992	0.9995	0.9992	**1.0**$_{\pm 0.0}$
UMD	$0.9398_{\pm 0.033}$	**1.0**	0.9375	0.9931	0.9583	0.9931	0.9931	0.9931	0.9722	0.9931	0.9931	0.9931	0.9931	$0.9931_{\pm 0.0}$
UWaveGestureLibraryAll	$0.9454_{\pm 0.001}$	0.9665	0.9629	0.9595	0.9428	0.9587	0.9436	0.9375	0.9319	0.9506	0.9573	0.9559	0.952	**0.9689**$_{\pm 0.003}$
UWaveGestureLibraryX	$0.8062_{\pm 0.003}$	0.8079	0.7984	0.7931	0.7998	0.821	0.8314	0.8417	0.8004	0.8445	0.8495	0.8431	0.8498	**0.8664**$_{\pm 0.004}$
UWaveGestureLibraryY	$0.7248_{\pm 0.002}$	0.715	0.7164	0.7144	0.7074	0.7387	0.7741	0.7744	0.6957	0.7658	0.7834	0.7783	0.787	**0.8135**$_{\pm 0.005}$
UWaveGestureLibraryZ	$0.7519_{\pm 0.002}$	0.7513	0.7379	0.7401	0.7328	0.7552	0.7739	0.7744	0.7554	0.7747	0.7956	0.7887	0.7859	**0.8113**$_{\pm 0.004}$
Wafer	$0.9088_{\pm 0.0}$	0.9953	0.9959	0.994	0.9992	0.9977	0.9992	**0.9998**	0.9961	0.9969	0.9995	0.9995	0.9995	$0.997_{\pm 0.001}$
Wine	$0.6358_{\pm 0.039}$	0.7778	0.7222	0.8333	0.8704	0.8519	0.6667	0.8333	0.7778	0.8333	0.8148	0.8333	**0.8889**	$0.7284_{\pm 0.057}$
WordSynonyms	$0.639_{\pm 0.008}$	0.6442	0.5972	0.685	0.5987	0.6677	0.6771	0.6771	0.6003	0.7116	0.7194	0.732	0.721	**0.7435**$_{\pm 0.013}$
Worms	$0.7229_{\pm 0.015}$	0.5714	0.5584	0.6883	0.7922	0.7792	0.7792	**0.8571**	0.7662	0.7792	0.7922	0.8052	0.8312	$0.7922_{\pm 0.039}$
WormsTwoClass	$0.8182_{\pm 0.013}$	0.6104	0.5714	0.7532	0.8052	0.8052	0.8182	0.8442	0.6753	0.7922	0.8312	0.8182	**0.8831**	$0.8182_{\pm 0.013}$
Yoga	$0.8289_{\pm 0.005}$	0.8603	0.8653	0.8507	0.7947	0.819	0.8403	0.8793	0.7493	0.8557	0.887	0.8713	**0.901**	$0.893_{\pm 0.012}$
Average	0.7969	0.7806	0.7707	0.7984	0.8202	0.8244	0.8288	0.8347	0.7905	0.836	0.8397	0.8466	0.8494	0.85
Best Counts	12	20	18	9	8	10	15	17	8	13	15	24	24	36