User-friendly Foundation Model Adapters for Multivariate Time Series Classification

Romain Ilbert*[1,2] Vasilii Feofanov*[1] Malik Tiomoko[1] Ievgen Redko[1] Themis Palpanas[2]
[1]Huawei Noah's Ark Lab, Paris, France [2]LIPADE, Université Paris-Cité, Paris, France

***Abstract*—Foundation models, while highly effective, are often resource-intensive, requiring substantial inference time and memory. This paper addresses the challenge of making these models more accessible with limited computational resources through *meta-channel learning* approaches. Our goal is to enable users to run large pre-trained foundation models on standard GPUs without sacrificing performance. We propose a *latent space compression* strategy that restructures the feature space while preserving essential temporal information. Surprisingly, we show that reducing the latent space to only 2.10% of its original size retains 96.15% of the classification accuracy of the full-sized model. To achieve this, we investigate both classical methods and neural network-based adapters for optimizing multivariate time series representations. Our experiments demonstrate up to a 10× speedup compared to the baseline model without performance degradation, while allowing up to 4.5× more datasets to fit on a single GPU. This enhancement makes foundation models more practical and scalable for real-world applications.**

***Index Terms*—Foundation Models, Dimensionality Reduction, Multivariate Time Series, Neural Adapters, PCA.**

I. Introduction

In recent years, the field of time series analysis has seen notable progress, particularly through the development of deep learning models [1]–[6]. These models have shown promising capabilities in learning expressive representations tailored to temporal data. However, in contrast to fields like natural language processing (NLP) and computer vision, time series research lacks large-scale unified benchmarks. As a result, models are generally designed to be compact and efficient, often relying on regularization strategies or data augmentation techniques to improve generalization and robustness [7]–[11].

Nevertheless, with the steady enrichment of classification-oriented benchmarks such as UCR and the UEA archive—which regularly incorporate new datasets—the landscape is evolving. This growing diversity of public benchmarks has paved the way for the emergence of Time Series Foundation Models (TSFM), which aim to learn general-purpose representations transferable across a wide range of tasks. These pre-trained models have revolutionized NLP [12], [13] and computer vision [14]. Inspired by these successes, TSFMs are now striving to generalize effectively across a wide range of tasks by leveraging large heterogeneous datasets. The main appeal of these models lies in their ability to generate universal representations that can be easily adapted to various applications through simple fine-tuning, thus significantly reducing the requirement for extensive task-specific annotated data.

Despite their remarkable performance, foundation models present significant practical challenges. Specifically, their utilization is often hindered by substantial computational resource requirements, including GPU memory and inference time. This problem is exacerbated in the case of multivariate time series, which frequently consist of hundreds or even thousands of channels. Consequently, directly applying existing foundation models to this type of data quickly becomes impractical, especially with limited hardware resources. This limitation restricts the broad adoption of foundation models in many real-world scenarios.

In response to these computational constraints, this paper aims to make foundation models accessible for multivariate time series classification, even when standard GPU resources are limited. We propose an innovative approach based on latent space compression techniques, significantly reducing the dimensionality of the problem while preserving the crucial information contained within the data. Specifically, we explore classical methods such as Principal Component Analysis (PCA), Singular Value Decomposition (SVD), and neural network-based adapters designed specifically to optimize representations of multivariate time series.

Our experiments conducted on a diverse set of datasets from the UEA archive clearly illustrate the effectiveness of our strategy. Our results indicate that despite significant compression (reducing dimensionality to just 2.10% of its original size), we manage to retain 96.15% of the full model's classification accuracy. Furthermore, we demonstrate substantial speed-ups, achieving up to $10\times$ faster inference, and enabling the simultaneous handling of up to $4.5\times$ more datasets on a single GPU. These remarkable results significantly enhance the practical usability and scalability of foundation models for real-world applications involving multivariate time series.

II. Related Work

Classical models for time series classification, including those based on Dynamic Time Warping [15], [16], kernel methods, shapelet-based algorithms [17], tree-based models [18], and dictionary-based approaches [19], [20], are effective for univariate time series but face challenges when extended to multivariate time series (MTS). Deep learning methods and random convolution techniques like ROCKET [21] and Multi-ROCKET show promise but typically treat each channel independently, leading to scalability and computational issues.

* Equal Contribution
Corresponding authors: Romain Ilbert <romain.ilbert@hotmail.fr>and Vasilii Feofanov <vasilii.feofanov@huawei.com>.

TABLE I: Average accuracy over 3 runs under full fine-tuning without an adapter (i.e., using all initial channels).

Model	Duck	Face	Finger	Hand	Heart	Insect	Vowels	Motor	NATOPS	PEMS	Phoneme	SpokeA
Mantis	COM	COM	COM	0.401 ± 0.021	COM	COM	0.981 ± 0.005	COM	0.937 ± 0.012	COM	0.342 ± 0.002	0.987 ± 0.001
MOMENT	COM	COM	COM	0.356 ± 0.016	COM	COM	0.925 ± 0.002	COM	TO	COM	TO	TO

TSFMs [22]–[26], inspired by advances in NLP and computer vision, offer potential for MTS classification but still struggle with complexity and inter-channel dependencies.

In recent years, TSFMs have gained traction for learning transferable representations from large-scale heterogeneous time series. Among them, we highlight MOMENT [22] that is based on T5 architecture that captures both global and local temporal patterns. Containing 341 million parameters, it is pre-trained via masked reconstruction loss to generalize across different tasks including classification. On the other hand, Mantis [27] is a foundation model with 8 million parameters that was pre-trained specifically for classification. Extracting convolution features and enriching them with statistical features, Mantis generates tokens to further feed them into a transformer and pre-train the whole network using a contrastive approach [28], [29] to promote robust representation learning.

III. Problem Formulation

Let N denote the number of samples, T the number of time steps, D the number of channels in each multivariate time series, and D' the reduced number of dimensions after applying dimensionality reduction ($D' \leq D$).

Objective. Our goal is to enable efficient multivariate time series classification using pre-trained models while preserving high classification accuracy. We focus on achieving rapid fine-tuning within a 2-hour window on a single GPU without significant performance degradation. To this end, we explore various dimensionality reduction techniques, which preprocess the input data before being processed by foundation models. We then evaluate different fine-tuning strategies to optimize performance under computational constraints.

Challenges. Table I presents the accuracy results of two TSFMs, Mantis and MOMENT, on a range of multivariate time series datasets under full fine-tuning without the use of any adapter, i.e., without dimensionality reduction. Notably, the results indicate that most of the foundation models encounter severe computational limitations when applied to multivariate data on standard hardware (NVIDIA Tesla V100-32GB GPU), as indicated by COM (CUDA Out of Memory error) and TO (2 hours Time Out) entries. These computational constraints underscore the difficulty of directly applying existing foundation models to multivariate time series with numerous channels, often leading to excessive resource consumption and failures to complete the fine-tuning process. This evidence motivates our exploration of dimensionality reduction techniques, which aim to alleviate these computational bottlenecks and enable foundation models to handle multivariate data more effectively without compromising accuracy.

Problem Definition. Let $X \in \mathbb{R}^{T \times D}$ denote a multivariate time series with T time steps and D channels, and let $y \in \mathcal{Y} = \{1, \ldots, K\}$ be the corresponding class label for a K-class classification task. We assume that a pre-trained foundation model f encodes each time series channel independently to an embedding vector of size p. Assuming D large, we introduce an adapter that performs latent space compression by mapping the original D channels onto $D' \leq D$ channels to enable efficient processing of high-dimensional data:

$$g : \mathbb{R}^{T \times D} \to \mathbb{R}^{T \times D'}.$$

We consider a set $\mathcal{G}$ of candidate dimensionality reduction techniques (e.g., PCA, truncated SVD, random projection, or neural-network–based linear combiners). The overall classification pipeline is then given by

$$H(X) = h\big(f\big(g(X)\big)\big),$$

where $h : \mathbb{R}^{D' \times p} \to \mathcal{Y}$ is a classification head. Our goal is to maximize the classification accuracy under different fine-tuning strategies while respecting a strict resource budget (i.e., fine-tuning must be finished within 2 hours on a single GPU).

Case 1: Adapter + Head Fine-Tuning

In this setting, the pre-trained foundation model f is kept frozen. The adapter g is parameterized by θ (denoted as g_θ) and the classification head h is parameterized by ϕ. The pipeline is defined as

$$H(X) = h_\phi\Big(f\big(g_\theta(X)\big)\Big).$$

The optimization problem is then:

$$\max_{\theta,\,\phi} \; \frac{1}{N} \sum_{i=1}^{N} \mathbb{I}\Big(h_\phi\big(f\big(g_\theta(X_i)\big)\big) = y_i\Big),$$

subject to:

$$g_\theta : \mathbb{R}^{T \times D} \to \mathbb{R}^{T \times D'}, \quad D' \leq D,$$

and the constraint that the overall fine-tuning (of θ and ϕ) is completed within two hours on a single GPU.

Case 2: Full Fine-Tuning

In this scenario, the foundation model f is parameterized by ψ and denoted as f_ψ, so that the entire pipeline is fine-tuned. Keeping both parameterized adapter and head the pipeline becomes:

$$H(X) = h_\phi\Big(f_\psi\big(g_\theta(X)\big)\Big).$$

The corresponding optimization problem is:

$$\max_{\theta,\psi,\,\phi} \; \frac{1}{N} \sum_{i=1}^{N} \mathbb{I}\Big(h_\phi\big(f_\psi\big(g_\theta(X_i)\big)\big) = y_i\Big),$$

subject to the same mapping constraint:

$$g_\theta : \mathbb{R}^{T \times D} \to \mathbb{R}^{T \times D'}, \quad D' \leq D,$$

and the same resource constraint (two hours on a single GPU).

Case 3: Head Fine-Tuning

This baseline configuration employs the identity mapping $g_{\text{id}}\colon \mathbb{R}^{T\times D} \to \mathbb{R}^{T\times D}$, thus passing all D channels directly to the foundation model f. Only the classification head h is fine-tuned, providing a reference scenario without dimension reduction where :

$$H(X) \;=\; h\big(f\big(g_{\text{id}}(X)\big)\big) \;=\; h\big(f(X)\big).$$

Thus, the optimization objective is:

$$\max_{\phi} \frac{1}{N} \sum_{i=1}^{N} \mathbf{1}\Big(h\big(f(X_i)\big) = y_i\Big),$$

under the same resource constraints.

In summary, three distinct approaches are investigated: **(1)** freezing f while fine-tuning the adapter and head, **(2)** fully fine-tuning $\{g_\theta, f_\psi, h_\phi\}$, and **(3)** relying on the identity mapping and training only the head. Our primary objective is to reduce channels from D to D' without compromising classification accuracy, while adhering to strict computational limits.

IV. Proposed Approach

To address the computational challenges described in our problem formulation, we propose the use of an adapter to reduce the number of channels from D to D' ($D' \leq D$). This adapter is selected from a candidate set $\mathcal{G}$ of dimensionality reduction techniques to maximize classification accuracy under strict resource constraints, as highlighted in Figure 1. In what follows, we briefly describe each candidate method in $\mathcal{G}$:

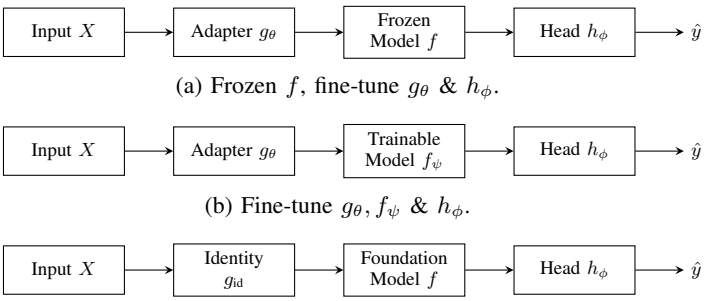
a) Principal Component Analysis (PCA) [30], [31]: seeks to find an orthogonal basis of principal components where a few components capture most of the data's variance. Applying PCA to 3D matrices (N, T, D) poses challenges. A common approach reshapes the data into $(N, T \times D)$ and projects it to $(N, T \times D')$, but this disrupts the temporal structure. Additionally, when $N \ll T \times D$, PCA becomes computationally unstable. To address this, we reshape the data to $(N \times T, D)$, allowing PCA to focus on correlations between channels over all time steps, effectively capturing spatial correlations while preserving temporal information. The learned rotation matrix $W \in \mathbb{R}^{D'\times D}$ linearly combines the original channels into a lower-dimensional space, applied consistently across all time steps.

b) Truncated Singular Value Decomposition (SVD) [32]: Unlike PCA, SVD operates directly on the data matrix without centering it, decomposing it into its top k singular values and vectors. This method effectively captures the principal directions of variance.

c) Random Projection (Rand Proj) [33]: is a computationally efficient technique that projects the data onto a lower-dimensional subspace using randomly generated directions. Unlike PCA, it does not aim to capture the most variance but instead focuses on providing a quick dimensionality reduction solution with minimal computational cost.

d) Variance-Based Feature Selection (VAR) [34]: is a simple but effective method that selects features with the highest variance. Features with low variance are considered less informative and can be discarded without significantly affecting the overall representation of the data.

e) Linear Combiner (lcomb): introduces a new learnable adapter that performs a linear combination of channels before passing the data to the encoder and classification head. In contrast to unsupervised methods like PCA, this approach learns the rotation matrix $W \in \mathbb{R}^{D'\times D}$ in a supervised manner, either by fine-tuning the adapter and head or the entire network. Given the large search space for possible linear combinations, we apply a top-k rule to each row of W, retaining only the top k entries to ensure more efficient optimization.



(a) Frozen f, fine-tune g_θ & h_ϕ.

(b) Fine-tune g_θ, f_ψ & h_ϕ.

(c) No adapter, fine-tune only h_ϕ.

Fig. 1: Three fine-tuning scenarios in which each adapter g is selected from $\mathcal{G} = \{\text{PCA}, \text{Truncated SVD}, \text{Rand Proj}, \text{VAR}, \text{lcomb}\}$.

V. Experimental Evaluation

Experimental Setup. All experiments were conducted using an NVIDIA Tesla V100 GPU (32GB) with a strict runtime constraint of two hours per fine-tuning task. Models exceeding these limits are reported as either `COM` (CUDA Out-of-Memory) or `TO` (Time-Out).

Foundation Models. We evaluate two representative TSFMs:

- MOMENT [22]: A large-scale transformer-based model pre-trained via masked reconstruction (341M parameters).
- Mantis [27]: A smaller Vision Transformer (ViT)-based model pre-trained via contrastive learning (8M parameters).

Datasets. This study draws on 12 UEA datasets [35], each containing at least 10 channels, to ensure that dimensionality reduction (from D to D') confers a tangible advantage. The UEA archive comprises 30 multivariate datasets, but those with fewer than 10 channels generally derive limited benefit from such reduction. While our method is applicable to any D, it provides the greatest impact when D is sufficiently large. The experimental results presented in this work are based on a diverse set of datasets, whose main characteristics are summarized in Table III. These datasets span a variety of domains

TABLE II: Performance comparison between different adapter configurations for MOMENT and Mantis foundation models; new number of channels=5. Best performance in **bold**; 2nd best in *italic*. Results for fine-tuning head only given for reference.

Dataset	Model	head	adapter+head					
		no adapter	PCA	SVD	Rand_Proj	VAR	lcomb	lcomb_top_k
DuckDuckGeese	MOMENT	$0.460_{\pm 0.016}$	*0.627*$_{\pm 0.023}$	**0.667**$_{\pm 0.012}$	$0.500_{\pm 0.040}$	$0.407_{\pm 0.012}$	$0.427_{\pm 0.046}$	$0.393_{\pm 0.114}$
	Mantis	$0.420_{\pm 0.020}$	*0.558*$_{\pm 0.023}$	**0.600**$_{\pm 0.032}$	$0.487_{\pm 0.023}$	$0.400_{\pm 0.060}$	$0.360_{\pm 0.020}$	$0.393_{\pm 0.031}$
FaceDetection	MOMENT	$0.623_{\pm 0.006}$	**0.567**$_{\pm 0.002}$	*0.566*$_{\pm 0.001}$	$0.552_{\pm 0.014}$	$0.555_{\pm 0.001}$	TO	TO
	Mantis	$0.595_{\pm 0.004}$	**0.554**$_{\pm 0.001}$	*0.551*$_{\pm 0.007}$	$0.533_{\pm 0.004}$	$0.539_{\pm 0.007}$	$0.548_{\pm 0.008}$	$0.550_{\pm 0.008}$
FingerMovement	MOMENT	$0.573_{\pm 0.012}$	*0.593*$_{\pm 0.032}$	$0.573_{\pm 0.012}$	$0.573_{\pm 0.025}$	**0.613**$_{\pm 0.021}$	$0.573_{\pm 0.032}$	$0.540_{\pm 0.017}$
	Mantis	$0.627_{\pm 0.015}$	**0.593**$_{\pm 0.044}$	$0.530_{\pm 0.030}$	$0.570_{\pm 0.075}$	*0.582*$_{\pm 0.040}$	$0.580_{\pm 0.020}$	$0.567_{\pm 0.046}$
HandMovementDirection	MOMENT	$0.401_{\pm 0.008}$	*0.410*$_{\pm 0.043}$	$0.365_{\pm 0.036}$	$0.405_{\pm 0.041}$	$0.369_{\pm 0.039}$	$0.378_{\pm 0.047}$	**0.414**$_{\pm 0.008}$
	Mantis	$0.342_{\pm 0.021}$	**0.396**$_{\pm 0.021}$	*0.351*$_{\pm 0.089}$	$0.329_{\pm 0.083}$	$0.329_{\pm 0.031}$	$0.320_{\pm 0.034}$	$0.320_{\pm 0.028}$
Heartbeat	MOMENT	$0.740_{\pm 0.003}$	$0.732_{\pm 0.000}$	$0.732_{\pm 0.005}$	**0.756**$_{\pm 0.005}$	$0.725_{\pm 0.006}$	*0.737*$_{\pm 0.005}$	*0.737*$_{\pm 0.013}$
	Mantis	$0.811_{\pm 0.010}$	$0.766_{\pm 0.005}$	$0.737_{\pm 0.012}$	$0.776_{\pm 0.013}$	**0.780**$_{\pm 0.010}$	$0.748_{\pm 0.006}$	*0.779*$_{\pm 0.014}$
InsectWingbeat	MOMENT	$0.284_{\pm 0.003}$	**0.239**$_{\pm 0.003}$	*0.224*$_{\pm 0.003}$	$0.193_{\pm 0.027}$	$0.195_{\pm 0.004}$	$0.167_{\pm 0.014}$	$0.213_{\pm 0.010}$
	Mantis	$0.614_{\pm 0.005}$	$0.344_{\pm 0.013}$	*0.352*$_{\pm 0.010}$	$0.333_{\pm 0.035}$	$0.238_{\pm 0.012}$	$0.171_{\pm 0.013}$	**0.354**$_{\pm 0.041}$
JapaneseVowels	MOMENT	$0.885_{\pm 0.002}$	$0.801_{\pm 0.009}$	*0.803*$_{\pm 0.003}$	$0.796_{\pm 0.011}$	$0.734_{\pm 0.008}$	$0.797_{\pm 0.035}$	**0.819**$_{\pm 0.027}$
	Mantis	$0.979_{\pm 0.006}$	**0.922**$_{\pm 0.009}$	$0.897_{\pm 0.012}$	*0.902*$_{\pm 0.008}$	$0.885_{\pm 0.010}$	$0.798_{\pm 0.070}$	$0.816_{\pm 0.027}$
MotorImagery	MOMENT	$0.643_{\pm 0.015}$	$0.590_{\pm 0.010}$	**0.607**$_{\pm 0.012}$	$0.567_{\pm 0.032}$	$0.550_{\pm 0.010}$	$0.583_{\pm 0.015}$	*0.593*$_{\pm 0.025}$
	Mantis	$0.600_{\pm 0.036}$	*0.593*$_{\pm 0.025}$	$0.590_{\pm 0.017}$	$0.577_{\pm 0.029}$	**0.607**$_{\pm 0.025}$	$0.557_{\pm 0.045}$	**0.607**$_{\pm 0.055}$
NATOPS	MOMENT	$0.872_{\pm 0.011}$	*0.776*$_{\pm 0.008}$	$0.739_{\pm 0.017}$	$0.774_{\pm 0.032}$	**0.813**$_{\pm 0.020}$	$0.596_{\pm 0.017}$	$0.769_{\pm 0.031}$
	Mantis	$0.944_{\pm 0.011}$	**0.874**$_{\pm 0.014}$	$0.820_{\pm 0.012}$	*0.852*$_{\pm 0.038}$	$0.850_{\pm 0.035}$	$0.787_{\pm 0.003}$	$0.826_{\pm 0.036}$
PEMS-SF	MOMENT	$0.834_{\pm 0.026}$	$0.678_{\pm 0.007}$	$0.511_{\pm 0.022}$	$0.644_{\pm 0.027}$	$0.611_{\pm 0.015}$	**0.740**$_{\pm 0.010}$	*0.697*$_{\pm 0.013}$
	Mantis	$0.923_{\pm 0.023}$	**0.674**$_{\pm 0.032}$	*0.640*$_{\pm 0.045}$	$0.615_{\pm 0.023}$	$0.615_{\pm 0.055}$	$0.584_{\pm 0.025}$	$0.594_{\pm 0.065}$
PhonemeSpectra	MOMENT	$0.234_{\pm 0.001}$	*0.234*$_{\pm 0.002}$	$0.212_{\pm 0.002}$	**0.245**$_{\pm 0.003}$	$0.228_{\pm 0.004}$	TO	TO
	Mantis	$0.296_{\pm 0.003}$	$0.270_{\pm 0.003}$	$0.259_{\pm 0.001}$	*0.293*$_{\pm 0.002}$	**0.294**$_{\pm 0.004}$	$0.279_{\pm 0.002}$	$0.286_{\pm 0.001}$
SpokenArabicDigits	MOMENT	$0.977_{\pm 0.001}$	*0.972*$_{\pm 0.000}$	**0.978**$_{\pm 0.000}$	$0.961_{\pm 0.008}$	$0.935_{\pm 0.002}$	TO	TO
	Mantis	$0.940_{\pm 0.003}$	**0.962**$_{\pm 0.003}$	$0.933_{\pm 0.001}$	$0.879_{\pm 0.004}$	*0.946*$_{\pm 0.003}$	$0.834_{\pm 0.019}$	$0.873_{\pm 0.019}$
Avg Ratio to head only	MOMENT	1.000	0.973	0.939	0.930	0.893	0.870	0.904
	Mantis	1.000	0.950	0.920	0.900	0.882	0.823	0.875

and tasks, offering a comprehensive evaluation of the fine-tuning methods under consideration. For instance, the datasets include time-series data from physiological measurements (e.g., *Heartbeat*, *MotorImagery*), sensor readings (e.g., *PEMS-SF*), and acoustic signals (e.g., *PhonemeSpectra*, *SpokenArabicDigits*). The number of channels, sequence lengths, and class distributions vary significantly across datasets, ensuring that the results generalize across different data modalities and problem settings. In the case of the *InsectWingbeat* dataset, we specifically subsampled 1000 examples from the original training set (which contains 30,000 examples) and 1000 from the original test set (of 20,000 examples) to reduce computational overhead while maintaining sufficient variety in the data for robust model evaluation. Each dataset was carefully chosen to challenge the models across different feature spaces, class imbalances, and temporal dependencies. For example, the *JapaneseVowels* dataset focuses on speaker classification based on vowel sounds, while the *DuckDuckGeese* dataset involves distinguishing animal sounds with varying levels of complexity in terms of sequence length and channel dimensionality. By including these datasets, we ensure that the evaluation framework captures the performance of fine-tuning methods across a wide spectrum of classification tasks.

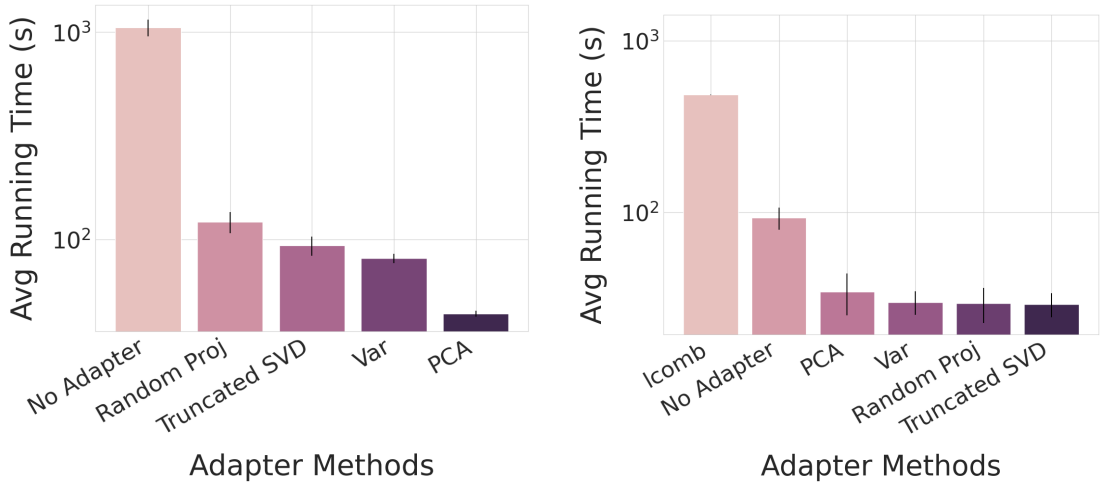
Definitions. To provide clarity for subsequent discussions and experiments, we begin by defining two central terms used throughout this work: *head* and *adapter*. A *head* refers specifically to the linear classification layer appended at the end of a foundation model's output, responsible for mapping learned representations into classification predictions. In contrast, an *adapter* is an intermediate processing module inserted upstream of the foundation model. Its primary function is to modify or compress the input representations, potentially enhancing efficiency and adaptability for fine-tuning on downstream tasks.

Full Fine-Tuning Regime. We initially explored the implications of fully fine-tuning foundation models without employing any adapters. Our experimental constraints, notably limited GPU memory and a maximum runtime of two hours per task, revealed significant limitations associated with this approach. Specifically, attempts at full fine-tuning often resulted in errors such as Time-Out (TO) or CUDA Out of Memory (COM), indicating that such comprehensive fine-tuning is computationally prohibitive for practical use.

For a more in-depth understanding, we analyzed the few datasets capable of completing full fine-tuning without adapters. We compared these outcomes with a simpler configuration—fine-tuning only the head after reducing dimensionality through adapters. On the *Hand* dataset, full fine-

TABLE III: Main characteristics of the considered datasets.

Dataset	Train Size	Test Size	# of channels	Sequence Len	# of classes
DuckDuckGeese (Duck)	60	40	1345	270	5
FaceDetection (Face)	5890	3524	144	62	2
FingerMovements (Finger)	316	100	28	50	2
HandMovementDirection (Hand)	320	147	10	400	4
Heartbeat (Heart)	204	205	61	405	2
InsectWingbeat (Insect)	1000	1000	200	78	10
JapaneseVowels (Vowels)	270	370	12	29	9
MotorImagery (Motor)	278	100	64	3000	2
NATOPS	180	180	24	51	6
PEMS-SF (PEMS)	267	173	963	144	7
PhonemeSpectra (Phoneme)	3315	3353	11	217	39
SpokenArabicDigits (SpokeA)	6599	2199	13	93	10



(a) `MOMENT` models (b) `Mantis` models

Fig. 2: Comparison of running times for `MOMENT` and `Mantis` models, averaged over all datasets and three seeds.

tuning yielded an average accuracy of 0.401 for Mantis and 0.356 for MOMENT, while head-only fine-tuning resulted in similar accuracies (0.401 for Mantis and 0.342 for MOMENT), with a notably high variance for MOMENT. These results indicate that the extensive computational demands of full fine-tuning do not necessarily provide meaningful improvements in performance, at least for this dataset.

Furthermore, the *Vowels* dataset, the second dataset that satisfied our computational constraints, showed slightly superior performance when fully fine-tuned compared to head-only fine-tuning (0.981 vs. 0.979 for Mantis and 0.925 vs. 0.885 for MOMENT). However, given that MOMENT comprises 341 million parameters, the marginal gains obtained from full fine-tuning are overshadowed by the significant computational overhead. Indeed, among the 12 datasets initially selected for evaluation, only two successfully completed full fine-tuning with MOMENT without encountering memory or runtime constraints, further emphasizing the practical infeasibility of extensive full-model fine-tuning.

Head Only vs Adapter+Head. Given the prohibitive computational cost of full fine-tuning, our subsequent experiments focused on comparing the simpler head-only fine-tuning scenario with a more computationally manageable strategy: fine-tuning the head together with an upstream adapter. One might question the necessity of adapter+head fine-tuning, especially considering the favorable computational profile of head-only fine-tuning. However, our empirical findings indicate compelling advantages of integrating adapters despite the seemingly minimal difference in performance.

Indeed, adapter+head fine-tuning maintains an average of 97.15% of the accuracy obtained from head-only fine-tuning across both foundation models considered (MOMENT and Mantis). Moreover, the adoption of adapters dramatically accelerates fine-tuning—achieving approximately 10 times faster training for MOMENT and about two times faster for Mantis compared to head-only fine-tuning. Thus, the integration of adapters represents a significant computational advantage, enabling efficient fine-tuning of large foundation models without compromising their classification capabilities.

Statistical tests conducted (Figure 6) reinforce this finding, demonstrating no statistically significant difference between head-only fine-tuning and adapter+head fine-tuning across the datasets tested. Yet, as clearly depicted in Figure 2, adapters significantly reduce computational time. This efficiency is

critical, particularly when computational resources or runtime are constrained, further justifying the adapter-based approach.

Results. Our detailed experimental analysis includes multiple adapter configurations applied during the fine-tuning process of the head and adapter modules in two foundation models: MOMENT and Mantis. We conducted evaluations across 12 diverse datasets from the UEA archive, each initially possessing over ten channels, and subsequently reduced the dimensionality from an average of 240 channels to merely 5 using adapters. Rather than merely eliminating channels, adapters leverage linear or nonlinear transformations to create *metachannels*, effectively compressing the data while preserving significant informational content.

Remarkably, the PCA-based adapter preserved approximately 97.30% of the original accuracy for MOMENT and 95.00% for Mantis compared to the no-adapter baseline, despite reducing dimensionality to just 2.08% of the original dimension. Statistical analyses (Figure 6) confirmed that performance differences across various adapter methods, including the baseline (no adapter), were statistically negligible.

Nonetheless, adapters exhibited substantial computational advantages. For MOMENT, adapters achieved an average speed-up exceeding 10 times relative to head-only fine-tuning without adapters, while for Mantis, adapters achieved around double the speed. An exception arose with the Linear Combiner (lcomb) adapter—a deep-learning approach requiring repeated invocations of the foundation model at each fine-tuning step, thereby incurring higher computational costs compared to simpler adapter methods such as PCA or SVD.

Moreover, certain datasets exhibited superior performance without any adapters, indicating that optimal dimensionality reduction may depend on dataset characteristics. Future investigations could explore more sophisticated adaptive dimensionality reduction strategies to enhance performance further.

Finally, our results highlight an important practical benefit: adapters significantly enhance the scalability of foundation models under constrained computational resources. Specifically, the lcomb adapter enabled fine-tuning on all 12 datasets for Mantis and 9 out of 12 datasets for MOMENT within a single GPU environment, compared to only 5 datasets for Mantis and 2 datasets for MOMENT in the full fine-tuning regime without adapters. This represents an increase in scalability of 2.4× for Mantis and 4.5× for MOMENT, emphasizing the substantial practical gains achievable by employing our adapter-based methodology.

VI. Qualitative Study

Hyperparameter Sensitivity of PCA. In this experiment, we implemented a variant of PCA called Patch PCA. Unlike the traditional approach where the input time series of shape (N, T, D) is reshaped into $(N \times T, D)$ before applying PCA, our method reshapes the input into $(N \times n_p, pws \times D)$, where n_p represents the number of patches in the sequence and pws refers to the patch window size. The case where $pws = 1$ corresponds to the standard PCA approach. We compare the results across different patch window sizes ($pws = 1, 8, 16$),

TABLE IV: Performance comparison between fine-tuning methods with different adapter configurations for the `MOMENT` foundation model

Dataset	adapter+head			
	PCA	Scaled PCA	Patch_8	Patch_16
DuckDuckGeese	$0.667_{\pm 0.012}$	$0.533_{\pm 0.031}$	$0.567_{\pm 0.031}$	$0.573_{\pm 0.031}$
FaceDetection	$0.566_{\pm 0.001}$	COM	$0.582_{\pm 0.003}$	$0.558_{\pm 0.004}$
FingerMovement	$0.573_{\pm 0.012}$	$0.563_{\pm 0.032}$	$0.633_{\pm 0.012}$	$0.563_{\pm 0.015}$
HandMovementDirection	$0.365_{\pm 0.036}$	$0.356_{\pm 0.043}$	$0.464_{\pm 0.021}$	$0.383_{\pm 0.021}$
Heartbeat	$0.732_{\pm 0.005}$	$0.728_{\pm 0.003}$	$0.738_{\pm 0.007}$	$0.741_{\pm 0.013}$
InsectWingbeat	$0.224_{\pm 0.003}$	$0.239_{\pm 0.003}$	$0.458_{\pm 0.002}$	$0.459_{\pm 0.004}$
JapaneseVowels	$0.803_{\pm 0.003}$	$0.723_{\pm 0.020}$	$0.967_{\pm 0.002}$	$0.963_{\pm 0.002}$
MotorImagery	$0.607_{\pm 0.012}$	$0.590_{\pm 0.020}$	$0.577_{\pm 0.006}$	$0.597_{\pm 0.015}$
NATOPS	$0.739_{\pm 0.017}$	$0.731_{\pm 0.012}$	$0.857_{\pm 0.003}$	$0.915_{\pm 0.003}$
PEMS-SF	$0.511_{\pm 0.022}$	$0.678_{\pm 0.007}$	$0.719_{\pm 0.012}$	$0.696_{\pm 0.018}$
PhonemeSpectra	$0.212_{\pm 0.002}$	$0.227_{\pm 0.008}$	$0.224_{\pm 0.001}$	$0.186_{\pm 0.001}$
SpokenArabicDigits	$0.978_{\pm 0.000}$	$0.963_{\pm 0.001}$	$0.967_{\pm 0.001}$	$0.956_{\pm 0.001}$

TABLE V: Performance comparison between fine tuning methods with different adapter configurations for `Mantis` foundation model

Dataset	adapter+head			
	PCA	Scaled PCA	Patch_8	Patch_16
DuckDuckGeese	$0.558_{\pm 0.023}$	$0.522_{\pm 0.023}$	$0.467_{\pm 0.031}$	$0.440_{\pm 0.035}$
FaceDetection	$0.554_{\pm 0.001}$	$0.550_{\pm 0.010}$	$0.551_{\pm 0.003}$	$0.547_{\pm 0.007}$
FingerMovement	$0.593_{\pm 0.044}$	$0.583_{\pm 0.023}$	$0.530_{\pm 0.036}$	$0.570_{\pm 0.053}$
HandMovementDirection	$0.367_{\pm 0.042}$	$0.327_{\pm 0.056}$	$0.396_{\pm 0.021}$	$0.369_{\pm 0.021}$
Heartbeat	$0.736_{\pm 0.010}$	$0.734_{\pm 0.014}$	$0.766_{\pm 0.005}$	$0.763_{\pm 0.018}$
InsectWingbeat	$0.344_{\pm 0.013}$	$0.268_{\pm 0.005}$	$0.287_{\pm 0.011}$	$0.266_{\pm 0.006}$
JapaneseVowels	$0.890_{\pm 0.008}$	$0.865_{\pm 0.016}$	$0.922_{\pm 0.009}$	$0.921_{\pm 0.011}$
MotorImagery	$0.567_{\pm 0.006}$	$0.552_{\pm 0.045}$	$0.593_{\pm 0.025}$	$0.573_{\pm 0.065}$
NATOPS	$0.837_{\pm 0.012}$	$0.840_{\pm 0.017}$	$0.874_{\pm 0.014}$	$0.870_{\pm 0.008}$
PEMS-SF	$0.584_{\pm 0.010}$	$0.613_{\pm 0.025}$	$0.634_{\pm 0.013}$	$0.674_{\pm 0.032}$
PhonemeSpectra	$0.270_{\pm 0.003}$	$0.262_{\pm 0.008}$	$0.234_{\pm 0.002}$	$0.205_{\pm 0.006}$
SpokenArabicDigits	$0.962_{\pm 0.003}$	$0.952_{\pm 0.003}$	$0.921_{\pm 0.006}$	$0.899_{\pm 0.002}$

as seen in Figure 3. These experiments show no clear pattern in performance across the different patch sizes, suggesting that the patch window size can be treated as a hyperparameter to be tuned based on the specific dataset.

Furthermore, we introduced two key hyperparameters for our PCA implementation: the patch window size (pws) and the option to scale the data before performing PCA. The results of PCA presented in Tables IV and V reflect the accuracy obtained for each configuration of these two hyperparameters, allowing us to explore the impact of different settings on performance and to choose the best hyperparameters to present the results in Table II. This flexibility in the PCA configuration allows us to adapt the method to a wide range of tasks,

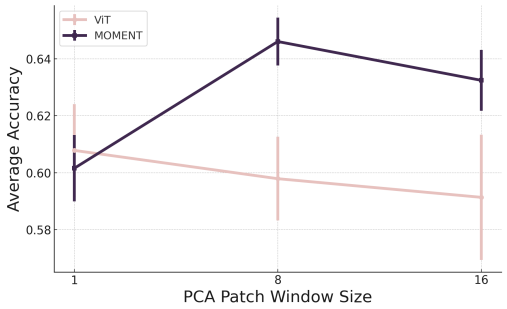
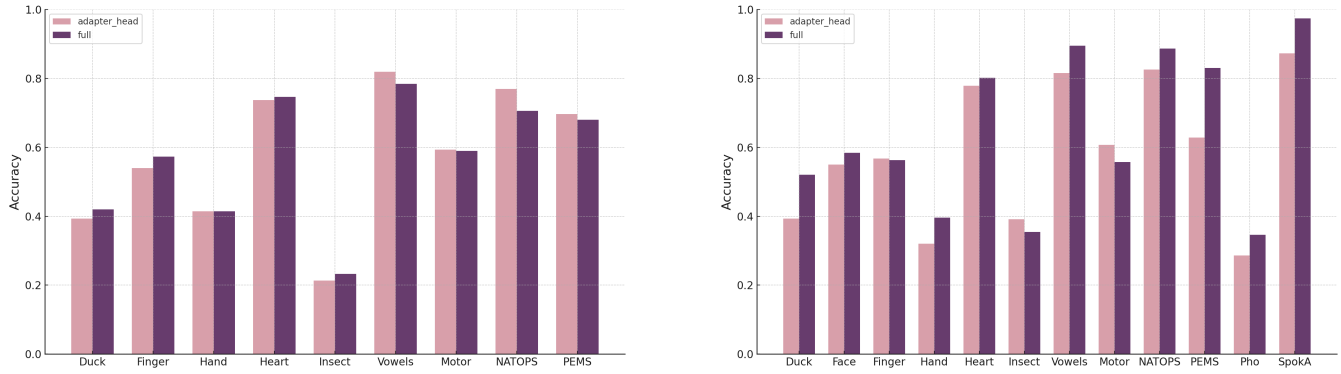


Fig. 3: Comparison of PCA and PatchPCA Methods for `Mantis` and `MOMENT` Models



(a) MOMENT (b) Mantis

Fig. 4: Full fine-tuning vs tuning adapter+head for lcomb.

optimizing both performance and computational efficiency.

Hyperparameter Sensitivity of lcomb. In addition to the standard *lcomb* configuration, we evaluated a variant called *lcomb_top_k*, which introduces a form of regularization to make the attention mechanism more stable. In *lcomb_top_k*, only the top k largest attention weights are selected, and each row of the attention matrix is rescaled by dividing by the sum of these k weights. For our experiments, we set $k = 7$. This mechanism is designed to reduce noise in the attention distribution, focusing the model on the most important relationships between elements in the input. The results shown in Figure 5 show the performance comparison between *lcomb* and *lcomb_top_k* across several datasets for both MOMENT and Mantis foundation models.

Figure 7 shows a comparison of the average rank for different adapter methods used in the MOMENT and Mantis foundation models. The average ranks were computed across all datasets and averaged over three seeds. The comparison gives insight into the relative performance of each adapter method when applied to these two models.

VII. Tests and Comparisons

Statistical Tests. The heatmap shown in Fig. 6 presents the pairwise p-values between different fine-tuning methods applied to the MOMENT and Mantis foundation models across several datasets. The methods compared include *No Adapter*, *PCA*, *SVD*, *Rand Proj*, *VAR*, and *lcomb*. The p-values were calculated using a two-sample Student's t-test with unequal variances, based on accuracy results obtained from three different seeds for each method.

The null hypothesis for each comparison states that there is no significant difference in the mean performance, in terms of accuracy, between the two methods being compared. A p-value close to 1 supports this hypothesis, indicating that the two methods yield statistically similar performance. In contrast, a p-value close to 0 suggests a significant difference. In the MOMENT heatmap, the lowest p-value observed is 0.46, while for Mantis, the minimum p-value is 0.25. These visualizations indicate that there is no statistically significant difference between fine-tuning using adapter + head with different adapters, and similarly, no difference is observed between adapter + head and head-only fine-tuning, regardless of the adapter used.

Rank comparisons. For the MOMENT foundation model, as depicted in Figure 7a, the *PCA* adapter ranks the lowest, indicating the best performance, while the *lcomb* adapter ranks the highest, showing relatively lower performance. The remaining adapters—*SVD*, *Rand_Proj*, and *VAR*—lie in between, with *Rand_Proj* and *SVD* showing close performance.

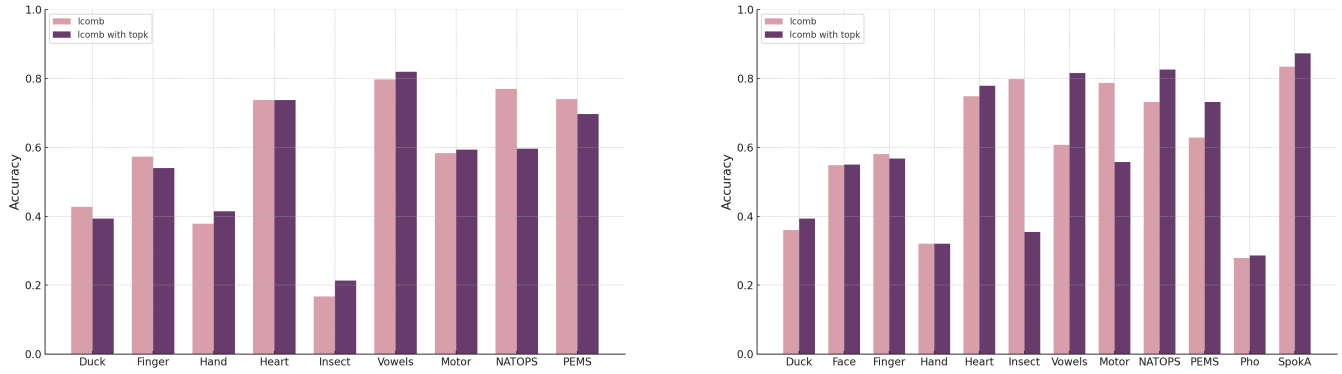
Similarly, in the case of the Mantis foundation model (Figure 7b), *PCA* exhibits the lowest average rank, implying superior performance. *Rand_Proj* also performs relatively worse in this case. The consistency of PCA's superior performance across both models highlights its effectiveness.

VIII. Conclusion

In this paper, we addressed the critical computational challenges associated with using foundation models for multivariate time series classification. Through a rigorous approach employing latent space compression adapters, we demonstrated that significant dimensionality reduction is achievable without substantial loss in accuracy. This method facilitates the efficient use of these models even with limited hardware resources, thereby making foundation models significantly more accessible and practical for a wide range of real-world applications.

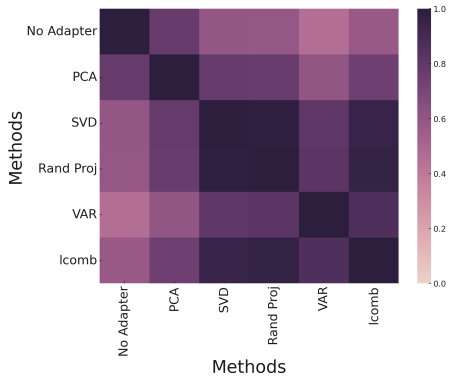
Our experimental results demonstrated that compressing the dimensionality to only 2.10% of the original embedding space retains 96.15% of the original performance, significantly reducing training times and memory requirements. Specifically, we achieved an average speed-up of about $10\times$ and enabled a substantial increase (up to $4.5\times$) in the number of datasets processed simultaneously on a single GPU. These advances clearly highlight the potential and value of our approach in overcoming practical barriers to the widespread adoption of foundation models in resource-constrained real-world contexts.

Several future research directions could further enhance this promising approach. In particular, future work could explore

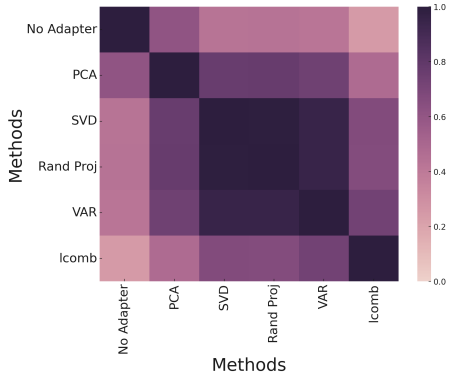


(a) MOMENT (b) Mantis

Fig. 5: Performance Comparison Between *lcomb* and *lcomb_top_k* Fine-Tuning Configurations for both MOMENT and Mantis Models

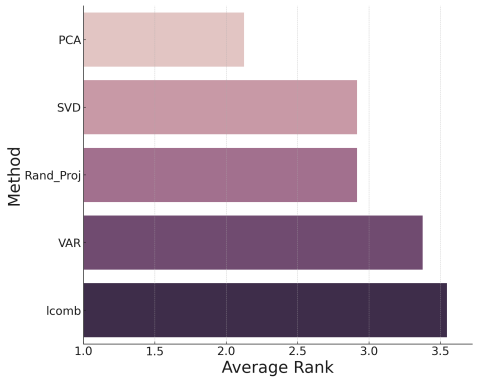


(a) Heatmap of Pairwise p-values for Adapter Methods for MOMENT Foundation Model

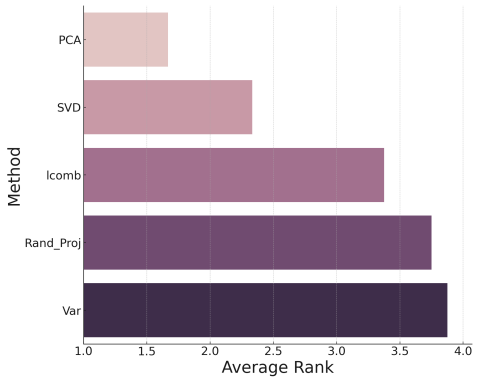


(b) Heatmap of Pairwise p-values for Adapter Methods for Mantis Foundation Model

Fig. 6: Heatmap of Pairwise p-values for Adapter Methods for MOMENT and Mantis Foundation Models averaged across all datasets and three different seeds



(a) Adapter's Average Rank for MOMENT Foundation Model



(b) Adapter's Average Rank for Mantis Foundation Model

Fig. 7: Comparison of Adapter's Average Rank for MOMENT and Mantis Foundation Models averaged across all datasets and three different seeds

more sophisticated compression techniques, combining linear and nonlinear adapters, and advanced machine learning methods to better capture complex temporal dependencies within multivariate series. Extending this framework to other temporal data types and additional applications such as forecasting and anomaly detection represents another promising area for future exploration.

ACKNOWLEDGMENTS

Supported by EU Horizon projects AI4Europe (101070000), TwinODIS (101160009), ARMADA (101168951), DataGEMS (101188416), RECITALS (101168490).

REFERENCES

[1] H. Wu, Y. Xie, Q. Tian *et al.*, "Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting," *Advances in Neural Information Processing Systems*, vol. 34, pp. 22 419–22 430, 2021.

[2] H. Zhou, S. Zhang, J. Peng, S. Zhang, G. Li, H. Xiong, W. Zhang, and J. Gao, "Informer: Beyond efficient transformer for long sequence time-series forecasting," *Proceedings of the AAAI Conference on Artificial Intelligence*, 2021.

[3] T. Zhou, Z. Ma, Q. Wen, X. Wang, L. Sun, and R. Jin, "FEDformer: Frequency enhanced decomposed transformer for long-term series forecasting," in *Proceedings of the 39th International Conference on Machine Learning (ICML)*, 2022, pp. 27 268–27 286.

[4] Y. Nie, N. H. Nguyen, P. Sinthong, and J. Kalagnanam, "A time series is worth 64 words: Long-term forecasting with transformers," *arXiv preprint arXiv:2211.14730*, 2022.

[5] Y. Liu, T. Hu, H. Zhang, H. Wu, S. Wang, L. Ma, and M. Long, "itransformer: Inverted transformers are effective for time series forecasting," in *The Twelfth International Conference on Learning Representations*, 2024. [Online]. Available: https://openreview.net/forum?id=JePfAI8fah

[6] R. Ilbert, A. Odonnat, V. Feofanov, A. Virmaux, G. Paolo, T. Palpanas, and I. Redko, "Unlocking the potential of transformers in time series forecasting with sharpness-aware minimization and channel-wise attention," 2024.

[7] R. Ilbert, T. V. Hoang, and Z. Zhang, "Data augmentation for multivariate time series classification: An experimental study," in *2024 IEEE 40th International Conference on Data Engineering Workshops (ICDEW)*, 2024, pp. 128–139.

[8] R. Ilbert, M. Tiomoko, C. Louart, V. Feofanov, T. Palpanas, and I. Redko, "Enhancing multivariate time series forecasting via multi-task learning and random matrix theory," in *NeurIPS Workshop on Time Series in the Age of Large Models*, 2024. [Online]. Available: https://openreview.net/forum?id=YaErRZ5SVG

[9] Q. Wen, L. Sun, F. Yang, X. Song, J. Gao, X. Wang, and H. Xu, "Time series data augmentation for deep learning: A survey," in *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence*. International Joint Conferences on Artificial Intelligence Organization, aug 2021.

[10] B. K. Iwana and S. Uchida, "An empirical survey of data augmentation for time series classification with neural networks," *PLOS ONE*, vol. 16, no. 7, p. e0254841, 2021.

[11] R. Ilbert, M. Tiomoko, C. Louart, A. Odonnat, V. Feofanov, T. Palpanas, and I. Redko, "Analysing multi-task regression via random matrix theory with application to time series forecasting," in *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

[12] J. Achiam, S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, F. L. Aleman, D. Almeida, J. Altenschmidt, S. Altman, S. Anadkat *et al.*, "Gpt-4 technical report," *arXiv preprint arXiv:2303.08774*, 2023.

[13] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar *et al.*, "Llama: Open and efficient foundation language models," *arXiv preprint arXiv:2302.13971*, 2023.

[14] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2021.

[15] S. Salvador and P. Chan, "Toward accurate dynamic time warping in linear time and space," *Intelligent Data Analysis*, vol. 11, no. 5, pp. 561–580, 2007.

[16] M. Cuturi and M. Blondel, "Soft-dtw: a differentiable loss function for time-series," in *International conference on machine learning*. PMLR, 2017, pp. 894–903.

[17] J. Lines, L. M. Davis, J. Hills, and A. Bagnall, "A shapelet transform for time series classification," in *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2012, pp. 289–297.

[18] H. Deng, G. Runger, E. Tuv, and V. Martyanov, "A time series forest for classification and feature extraction," *Information Sciences*, vol. 239, pp. 142–153, 2013.

[19] J. Lin, E. Keogh, L. Wei, and S. Lonardi, "Experiencing sax: a novel symbolic representation of time series," *Data Mining and knowledge discovery*, vol. 15, pp. 107–144, 2007.

[20] J. Lin, R. Khade, and Y. Li, "Rotation-invariant similarity in time series using bag-of-patterns representation," *Journal of Intelligent Information Systems*, vol. 39, pp. 287–315, 2012.

[21] A. Dempster, F. Petitjean, and G. I. Webb, "Rocket: exceptionally fast and accurate time series classification using random convolutional kernels," *Data Mining and Knowledge Discovery*, vol. 34, no. 5, pp. 1454–1495, 2020.

[22] M. Goswami, K. Szafer, A. Choudhry, Y. Cai, S. Li, and A. Dubrawski, "Moment: A family of open time-series foundation models," *arXiv preprint arXiv:2402.03885*, 2024.

[23] Y. Wang, Y. Qiu, P. Chen, K. Zhao, Y. Shu, Z. Rao, L. Pan, B. Yang, and C. Guo, "Rose: Register assisted general time series forecasting with decomposed frequency learning," *arXiv preprint arXiv:2405.17478*, 2024.

[24] A. Garza and M. Mergenthaler-Canseco, "Timegpt-1," *arXiv preprint arXiv:2310.03589*, 2023.

[25] T. Zhou, P. Niu, X. Wang, L. Sun, and R. Jin, "One fits all: Power general time series analysis by pretrained lm," *arXiv preprint arXiv:2302.11939*, 2023.

[26] K. Rasul, A. Ashok, A. R. Williams, A. Khorasani, G. Adamopoulos, R. Bhagwatkar, M. Biloš, H. Ghonia, N. V. Hassen, A. Schneider *et al.*, "Lag-llama: Towards foundation models for time series forecasting," *arXiv preprint arXiv:2310.08278*, 2023.

[27] V. Feofanov, S. Wen, M. Alonso, R. Ilbert, H. Guo, M. Tiomoko, L. Pan, J. Zhang, and I. Redko, "Mantis: Foundation model with adapters for multichannel time series classification," 2025. [Online]. Available: https://github.com/vfeofanov/mantis

[28] A. v. d. Oord, Y. Li, and O. Vinyals, "Representation learning with contrastive predictive coding," *arXiv preprint arXiv:1807.03748*, 2018.

[29] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick, "Momentum contrast for unsupervised visual representation learning," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 9729–9738.

[30] K. Pearson, "On lines and planes of closest fit to systems of points in space," *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, vol. 2, no. 11, pp. 559–572, 1901.

[31] I. Jolliffe, *Principal Component Analysis*, 2nd ed. Springer, 2002.

[32] G. H. Golub and C. F. Van Loan, *Matrix Computations*, 4th ed. Johns Hopkins University Press, 2013.

[33] E. Bingham and H. Mannila, "Random projection in dimensionality reduction: Applications and theoretical guarantees," in *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2001, pp. 245–250.

[34] I. Guyon and A. Elisseeff, "An introduction to variable and feature selection," *Journal of Machine Learning Research*, vol. 3, pp. 1157–1182, 2003.

[35] A. Bagnall, H. A. Dau, J. Lines, M. Flynn, J. Large, A. Bostrom, P. Southam, and E. Keogh, "The UEA multivariate time series classification archive, 2018," *arXiv preprint arXiv:1811.00075*, 2018.