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

The remarkable success of pre-trained models in natural language processing [1], [2] and computer vision [3] has inspired the extension of this paradigm to time series data. Time Series Foundation Models (TSFMs) aim to generalize across diverse downstream tasks by learning versatile encoders from large, heterogeneous pre-training datasets. This strategy offers both flexibility and efficiency, as deploying TSFMs for new tasks requires only fine-tuning, thus reducing the reliance on extensive labeled training data.

Depending on their pre-training objectives, TSFMs can be specialized for tasks like forecasting [4]–[6], classification [7], or designed to tackle various time series problems [8], [9]. However, most existing models are univariate, requiring separate processing for each channel in multivariate data. This approach poses significant limitations when dealing with datasets that contain hundreds or thousands of channels [10], [11], leading to excessive runtime and memory consumption, especially when fine-tuning on limited computational resources.

In this paper, we address the challenge of integrating foundation models with multivariate time series by introducing latent space compression techniques. While dimensionality reduction and feature selection are well-studied independently, their application to foundation models presents specific challenges, particularly in balancing computational efficiency and performance retention.

To tackle this, we explore various compression strategies, including Principal Component Analysis (PCA), Singular Value Decomposition (SVD), and neural-network-based adapters. These methods transform high-dimensional time series into compact representations, enabling efficient adaptation to foundation models, whether fine-tuned or with frozen weights. Our experiments show up to a 10× speedup while allowing 4.5× more datasets to fit on a single GPU, while reducing the latent space to only 2.10% of its original size and retaining 96.15% of the classification accuracy of the full-sized model. These results highlight the potential of latent space compression for improving the efficiency and accessibility of foundation models in multivariate time series classification.

II. Related Work

Classical models for time series classification, including those based on Dynamic Time Warping [12], [13], kernel methods, shapelet-based algorithms [14], tree-based models [15], and dictionary-based approaches [16], [17], 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 [18] and Multi-ROCKET show promise but typically treat each channel independently, leading to scalability and computational issues. TSFMs [4]–[6], [8], [9], 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` [9] 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` [19] 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 [20], [21] to promote robust representation learning.

* 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

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). The results of this scenario are shown in Appendix F.

Case 3: Head Fine-Tuning

This baseline configuration employs the identity mapping $g_{\text{id}} : \mathbb{R}^{T \times D} \to \mathbb{R}^{T \times D}$, 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 $\mathcal{G}$:

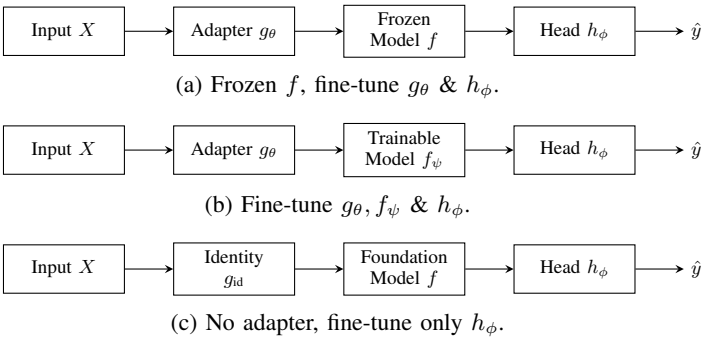
a) Principal Component Analysis (PCA) [22], [23]: seeks to find an orthogonal basis of principal components where a few components capture most of the data's variance. Applying PCA to 3D matrices (N, T, D) poses challenges. A common approach reshapes the data into $(N, T \times D)$ and projects it to $(N, T \times D')$, 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) [24]: 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) [25]: 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) [26]: 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.

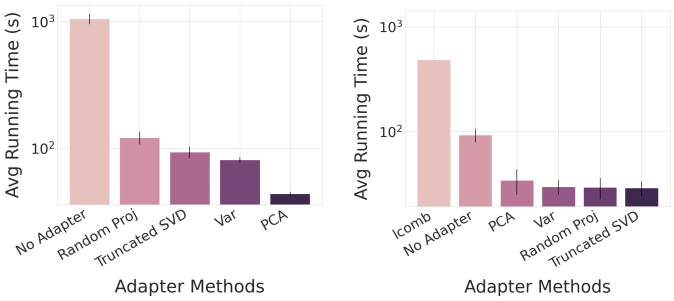


(a) Frozen f, fine-tune g_θ & h_ϕ.

(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 $\mathcal{G} = \{\text{PCA}, \text{Truncated SVD}, \text{Rand Proj}, \text{VAR}, \text{lcomb}\}$.



(a) `MOMENT` models

(b) `Mantis` models

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

V. Experimental Evaluation

Experimental Setup. All experiments were performed on a single NVIDIA Tesla V100-32GB GPU, with a 2-hour limit per run. Runs exceeding this limit are marked `TO` (Time Out), while those facing CUDA out-of-memory issues are labeled `COM` (CUDA Out of Memory).

Foundation Models. We evaluate two TSFMs: `MOMENT`, a large-scale model (341 million parameters) [9], and `Mantis`, a smaller ViT-based model (8 million parameters).

Datasets. This study draws on 12 UEA datasets [11], 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. Appendix A details the dataset characteristics.

We present an experimental comparison of multiple adapters when fine-tuning both the adapter and the head of a foundation model. The head is a linear classification layer added to the output of the foundation model, while the adapter is inserted upstream of the foundation model. We evaluate `MOMENT` and `Mantis` on twelve multivariate time series datasets from the UEA archive with more than ten features

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

(see Appendix A), reducing the data from an average of 240 channels to 5. Rather than discarding channels, these adapters construct new *metachannels* through linear or non-linear transformations, thereby retaining important information in a significantly lower-dimensional space ($\frac{5}{240} \approx 2.08\%$ of the original dimension). Remarkably, the PCA-based adapter preserves on average 97.30% of the accuracy of the no-adapter configuration for `MOMENT` and 95.00% for `Mantis`, despite this drastic dimensionality reduction. A deeper analysis of PCA's hyperparameter sensitivity is shown in Appendix B.

We also report results for the baseline scenario in which only the classification head is fine-tuned (i.e., without an adapter). As shown in Table II, accompanied by statistical tests in Appendix E, there is no statistically significant difference among the methods on average over all datasets, including the head-only baseline as shown in Appendix D. Nevertheless, Figure 2 demonstrates that using adapters drastically reduces computation time: for `MOMENT`, they are on average over ten times faster than the no-adapter setting, and for `Mantis`, they yield a two-fold speedup.

An exception is the Linear Combiner (`lcomb`) adapter, a deep learning-based model that requires invoking the foundation model at each fine-tuning step. In contrast, other (non-deep) methods only transform the data *once* into embeddings, then train the classification head without repeatedly running the foundation model. This significantly reduces runtime compared to approaches such as `lcomb`. A variant of this method has been analyzed in Appendix C.

Notably, Table II indicates that the no-adapter strategy outperforms on certain datasets, suggesting that the best dimensionality may vary per dataset. Consequently, more sophisticated adapters may be needed for robust dimension reduction in challenging cases.

Finally, by comparing results in Appendix F with Table I, we see that `lcomb` now enables fine-tuning on 12/12 datasets for `Mantis` and 9/12 for `MOMENT` on a single GPU, whereas full fine-tuning only accommodated 5 and 2 datasets, respectively. This corresponds to a $2.40\times$ increase for `Mantis` and a $4.50\times$ increase for `MOMENT` in terms of the number of datasets that fit in a single GPU (within two hours).

VI. Conclusions

We presented a latent space compression framework that preserves 96.15% of baseline accuracy while retaining only 2.1% of the original embedding dimensions, yielding up to a $10\times$ speedup and enabling $4.5\times$ more datasets per GPU. These gains demonstrate the effectiveness of adapters in scaling foundation models under limited resources. Future directions include refining compression techniques and extending the approach to more diverse time series domains.

Acknowledgment

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

References

[1] 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.

[2] 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.

[3] 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.

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

[5] 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.

[6] 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.

[7] C. Lin, X. Wen, W. Cao, C. Huang, J. Bian, S. Lin, and Z. Wu, "Nutime: Numerically multi-scaled embedding for large- scale time-series pretraining," *Transactions on Machine Learning Research*, 2024. [Online]. Available: https://openreview.net/forum?id=TwiSBZ0p9u

[8] 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.

[9] 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.

[10] W. W. Wei, *Multivariate time series analysis and applications*. John Wiley & Sons, 2018.

[11] 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.

[12] 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.

[13] 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.

[14] 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.

[15] 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.

[16] 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.

[17] 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.

[18] 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.

[19] 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

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

[21] 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.

[22] 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.

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

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

[25] 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.

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

APPENDIX

EXPERIMENTAL SETUP

A. Datasets

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 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.

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

B. PCA's Hyperparameter Sensitivity

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 ($pws = 1, 8, 16$), 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 hyperprameters 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, optimizing both performance and computational efficiency.

C. lcomb's Hyperparameter Sensitivity

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 4 show the performance comparison between *lcomb* and *lcomb_top_k* across several datasets for both `MOMENT` and `Mantis` foundation models.

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	$0.962_{\pm 0.003}$	$0.952_{\pm 0.003}$	$0.921_{\pm 0.006}$	$0.899_{\pm 0.002}$

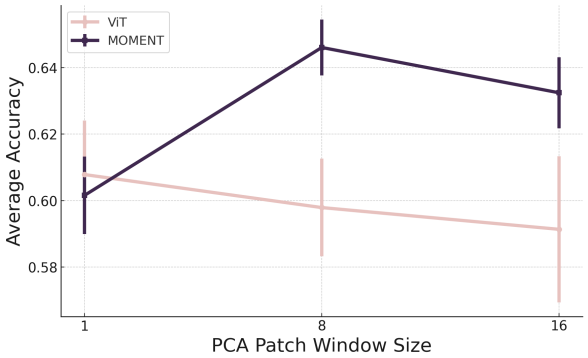


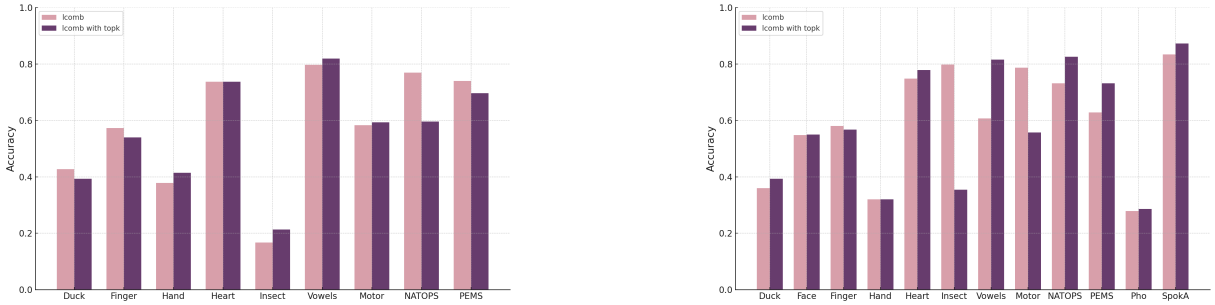
Fig. 3: Comparison of PCA and PatchPCA Methods for Mantis and MOMENT Models

D. Rank Comparisons

Figure 5 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.

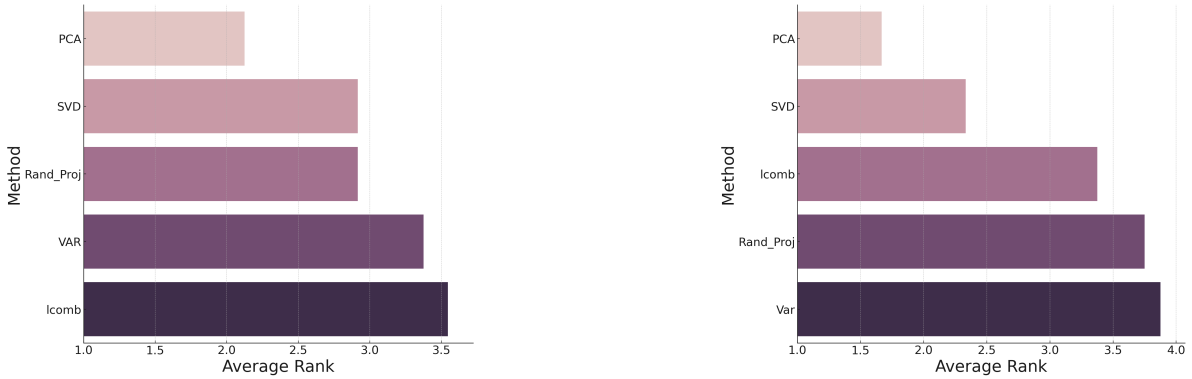
For the MOMENT foundation model, as depicted in Figure 5a, the *PCA* adapter ranks the lowest, indicating the best performance, while the *lcomb* adapter ranks the highest, showing relatively lower performance. The remaining adapters—*SVD*, *Rand_Proj*, and *VAR*—lie in between, with *Rand_Proj* and *SVD* showing close performance.

Similarly, in the case of the Mantis foundation model (Figure 5b), *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



(a) MOMENT (b) Mantis

Fig. 4: Performance Comparison Between *lcomb* and *lcomb_top_k* Fine-Tuning Configurations for both MOMENT and Mantis Models



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

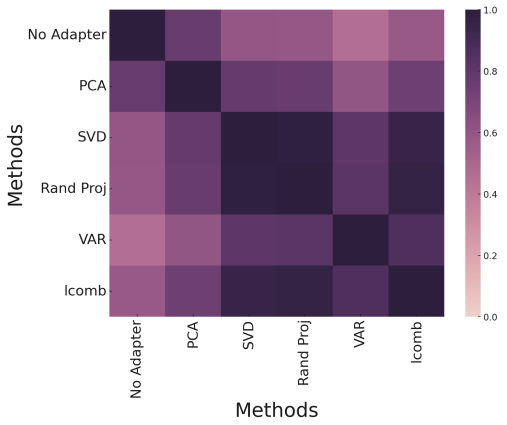
Fig. 5: Comparison of Adapter's Average Rank for MOMENT and Mantis Foundation Models averaged across all datasets and three different seeds

E. Statistical Tests

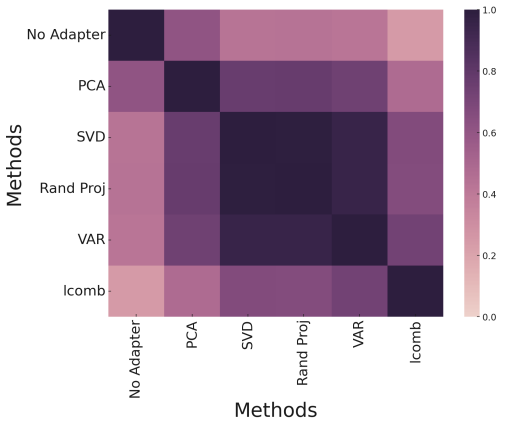
The heatmap shown in Fig. 6 presents the pairwise p-values between different fine-tuning methods applied to the MOMENT and Mantis foundation models across several datasets. The methods compared include *No Adapter*, *PCA*, *SVD*, *Rand Proj*, *VAR*, and *lcomb*. The p-values were calculated using a two-sample 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.

F. Full Fine-Tuning Regime

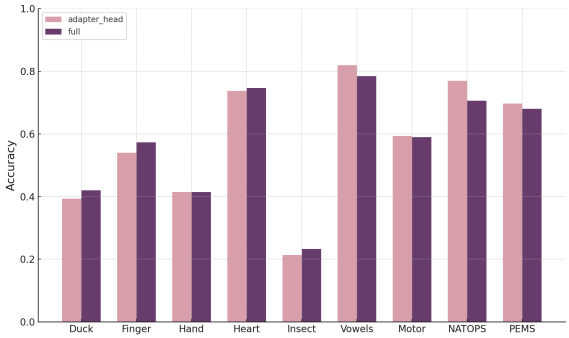


(a) Heatmap of Pairwise p-values for Adapter Methods for `MOMENT` Foundation Model

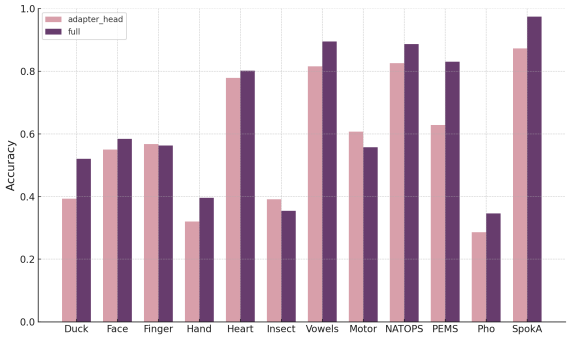


(b) Heatmap of Pairwise p-values for Adapter Methods for `Mantis` Foundation Model

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) `MOMENT`



(b) `Mantis`

Fig. 7: Full fine-tuning vs tuning adapter+head for `lcomb`.