User-friendly Foundation Model Adapters for Multivariate Time Series Classification

Vasilii Feofanov[*1] **Romain Ilbert**[*1,2] **Malik Tiomoko**[1]
Themis Palpanas[2] **Ievgen Redko**[1]
[1]Huawei Noah's Ark Lab, Paris, France [2]LIPADE, Paris Descartes University, 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 by exploring dimensionality reduction techniques. Our goal is to enable users to run large pre-trained foundation models on standard GPUs without sacrificing performance. We investigate classical methods such as Principal Component Analysis alongside neural network-based adapters, aiming to reduce the dimensionality of multivariate time series data while preserving key features. Our experiments show up to a 10x speedup compared to the baseline model, without performance degradation, and enable up to 4.5x more datasets to fit on a single GPU, paving the way for more user-friendly and scalable foundation models.

1 Introduction

The remarkable success of pre-trained models in natural language processing (NLP) (Achiam et al., 2023; Touvron et al., 2023) and computer vision (Dosovitskiy et al., 2021) 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 (Garza and Mergenthaler-Canseco, 2023; Rasul et al., 2023; Wang et al., 2024), classification (Lin et al., 2024), or designed to tackle various time series problems (Zhou et al., 2023; Goswami et al., 2024). However, most existing models are univariate, necessitating separate applications to each channel in multivariate data. This approach poses significant limitations when dealing with datasets that have hundreds or thousands of channels (Wei, 2018; Bagnall et al., 2018), leading to increased runtime and memory consumption, especially when fine-tuning on limited computational resources.

In this paper, we address this overlooked challenge by integrating dimensionality reduction techniques with foundation models for multivariate time series analysis. While dimensionality reduction (Van Der Maaten et al., 2009) and feature selection (Guyon and Elisseeff, 2003) are well-established individually, their combination with foundation models introduces unique challenges and hidden obstacles. We explore various methods, including Principal Component Analysis (PCA) and neural network-based adapters, to preprocess multivariate data and alleviate computational and memory constraints.

Our experiments demonstrate up to a 10x speedup and enable up to 4.5x more datasets to fit on a single GPU, all while maintaining classification accuracy, as verified by pairwise p-value tests. These

[*]Equal Contribution

Preprint. Under review.

results highlight the potential of dimensionality reduction to make foundation models more efficient and accessible for multivariate time series classification.

2 Related Work

Classical models for time series classification, including those based on Dynamic Time Warping (Salvador and Chan, 2007; Cuturi and Blondel, 2017), kernel methods (Salvador and Chan, 2007; Cuturi and Blondel, 2017), shapelet-based algorithms (Lines et al., 2012), tree-based models (Deng et al., 2013), and dictionary-based approaches (Lin et al., 2007, 2012), 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 (Dempster et al., 2020) and Multi-ROCKET show promise but typically treat each channel independently, leading to scalability and computational issues. TSFMs (Goswami et al., 2024; Wang et al., 2024; Garza and Mergenthaler-Canseco, 2023; Zhou et al., 2023; Rasul et al., 2023), inspired by advances in NLP and computer vision, offer potential for MTS classification but still struggle with complexity and inter-channel dependencies.

3 Framework

3.1 Problem setup

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

Datasets. We use 12 multivariate datasets from the UEA repository (Bagnall et al., 2018), each with at least 10 channels to enable meaningful dimensionality reduction. Detailed dataset characteristics are provided in Appendix A.1.

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 with 341 million parameters (Goswami et al., 2024), and `ViT`, a smaller model with 8 million parameters, inspired by ViT-based models like Nu-Time (Lin et al., 2024) and PatchTST (Nie et al., 2022). More implementation details are provided in Appendix B.1.

Objective. Our goal is efficient multivariate time series classification using pre-trained models, with accuracy as the primary metric. We focus on rapid fine-tuning within a 2-hour window on a single GPU, without significant performance loss. To achieve this, we test various dimensionality reduction techniques—such as PCA and neural network-based adapters—integrated at the beginning of the foundation model pipeline, and evaluate different fine-tuning strategies.

3.2 Motivation

Table 1 presents the accuracy results of two TSFMs, `ViT` 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 the `COM` and `TO` 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.

Table 1: Accuracy averaged over 3 model runs when the models are 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
ViT	COM	COM	COM	.401 ± .021	COM	COM	.981 ± .005	COM	.937 ± .012	COM	.342 ± .002	.987 ± .001
MOMENT	COM	COM	COM	.356 +- .016	COM	COM	.925 +- .002	COM	TO	COM	TO	TO

3.3 Feature-Level Transformation Methods

We explore several dimensionality reduction techniques to preprocess multivariate time series data for foundation models.

Principal Component Analysis (PCA) 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 can become 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.

Truncated Singular Value Decomposition (SVD) also reduces dimensionality by retaining the most significant components. 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.

Random Projection (Rand Proj) 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.

Variance-Based Feature Selection (VAR) 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.

Linear Combiner (lcomb) introduces a 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.

4 Experimental Results

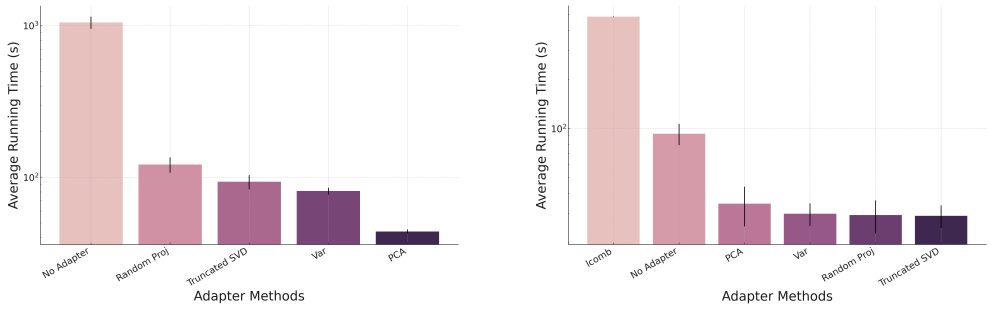
We present the experimental comparison between different adapters when fine-tuning both the adapter and the head a foundation model. The head refers to a classification linear layer at the end of the model, while the adapter is inserted before the foundation model. We report results for `MOMENT` and `ViT` across twelve datasets from the UEA archive with more than ten features (see Appendix A.1 for more details), reducing dimensionality to five channels. Also, we report the results when fine-tuning the head without an adapter.

The results, presented in Table 2, along with statistical tests in Appendix C.4, show no statistically significant difference between the method in average over all datasets, including fine-tuning the head only. However, as shown in Figure 1 , using adapters significantly reduces computation time. For instance, with `MOMENT`, adapters are on average over ten times faster than without adapters, and for `ViT`, they provide a two-fold speed increase.

The exception is the Linear Combiner (`lcomb`) adapter, a deep learning-based model requiring training and inference on the foundation model at every fine-tuning step. In contrast, other non-deep learning

Table 2: Performance comparison between different adapter configurations for `MOMENT` and `ViT` foundation models when the new number of channels is fixed to 5. Best performance is shown in **bold** and second best in *italic*. Results for fine-tuning the head only are given for the 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**$_{\pm 0.012}$	$0.500_{\pm 0.040}$	$0.407_{\pm 0.012}$	$0.427_{\pm 0.046}$	$0.393_{\pm 0.114}$
	ViT	$0.420_{\pm 0.020}$	*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
	ViT	$0.595_{\pm 0.004}$	**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**$_{\pm 0.021}$	$0.573_{\pm 0.032}$	$0.540_{\pm 0.017}$
	ViT	$0.627_{\pm 0.015}$	**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**$_{\pm 0.008}$
	ViT	$0.342_{\pm 0.021}$	**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*$_{\pm 0.013}$
	ViT	$0.811_{\pm 0.010}$	$0.766_{\pm 0.005}$	$0.737_{\pm 0.012}$	$0.776_{\pm 0.013}$	**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*$_{\pm 0.003}$	$0.193_{\pm 0.027}$	$0.195_{\pm 0.004}$	$0.167_{\pm 0.014}$	$0.213_{\pm 0.010}$
	ViT	$0.614_{\pm 0.005}$	$0.344_{\pm 0.013}$	*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**$_{\pm 0.027}$
	ViT	$0.979_{\pm 0.006}$	**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*$_{\pm 0.025}$
	ViT	$0.600_{\pm 0.036}$	*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**$_{\pm 0.020}$	$0.596_{\pm 0.017}$	$0.769_{\pm 0.031}$
	ViT	$0.944_{\pm 0.011}$	**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*$_{\pm 0.013}$
	ViT	$0.923_{\pm 0.023}$	**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
	ViT	$0.296_{\pm 0.003}$	$0.270_{\pm 0.003}$	$0.259_{\pm 0.001}$	*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
	ViT	$0.940_{\pm 0.003}$	**0.962**$_{\pm 0.003}$	$0.933_{\pm 0.001}$	$0.879_{\pm 0.004}$	*0.946*$_{\pm 0.003}$	$0.834_{\pm 0.019}$	$0.873_{\pm 0.019}$



(a) Running Time for `MOMENT` Foundation Model

(b) Running Time for ViT Foundation Model

Figure 1: Comparison of running times for `MOMENT` and ViT Foundation Models averaged across all datasets and three different seeds

adapters process the data once to generate embeddings, allowing inference and fine-tuning of the head only, without repeatedly running the foundation model. This substantially reduces computation time compared to methods like `lcomb`.

In Table 2, we can see that the no adapter approach outperforms on some specific datasets, which indicates that the intrinsic dimension is dataset-dependent and there is need in more complex adapter configurations to achieve sparse dimension reduction in the general case.

By comparing the results in Appendix C.5 with those in Table 1 , we observe that with the `lcomb` method, for example, we can now fine-tune 12 out of 12 datasets for `ViT` and 9 out of 12 datasets for `MOMENT` on a single GPU, compared to previously only 5 and 2 datasets, respectively, for full fine-tuning. This represents 2.4x more datasets that fit on a single GPU in less than two hours for `ViT` and 4.5x more for `MOMENT`.

5 Conclusion

We addressed computational and memory challenges in fine-tuning foundation models for multivariate time series by introducing dimensionality reduction techniques. These methods significantly improved efficiency, achieving up to 10x faster fine-tuning and enabling up to 4.5x more datasets to fit on

a single GPU, while maintaining comparable performance. Our results highlight the potential of adapters to enhance the scalability of foundation models. Future work may focus on further optimizing these techniques and applying them to larger datasets and more complex time series tasks.

References

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

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

Cuturi, M. and Blondel, M. (2017). Soft-dtw: a differentiable loss function for time-series. In *International conference on machine learning*, pages 894–903. PMLR.

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

Deng, H., Runger, G., Tuv, E., and Martyanov, V. (2013). A time series forest for classification and feature extraction. *Information Sciences*, 239:142–153.

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

Garza, A. and Mergenthaler-Canseco, M. (2023). Timegpt-1. *arXiv preprint arXiv:2310.03589*.

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

Guyon, I. and Elisseeff, A. (2003). An introduction to variable and feature selection. *Journal of machine learning research*, 3(Mar):1157–1182.

He, K., Fan, H., Wu, Y., Xie, S., and Girshick, R. (2020). Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9729–9738.

Lin, C., Wen, X., Cao, W., Huang, C., Bian, J., Lin, S., and Wu, Z. (2024). Nutime: Numerically multi-scaled embedding for large- scale time-series pretraining. *Transactions on Machine Learning Research*.

Lin, J., Keogh, E., Wei, L., and Lonardi, S. (2007). Experiencing sax: a novel symbolic representation of time series. *Data Mining and knowledge discovery*, 15:107–144.

Lin, J., Khade, R., and Li, Y. (2012). Rotation-invariant similarity in time series using bag-of-patterns representation. *Journal of Intelligent Information Systems*, 39:287–315.

Lines, J., Davis, L. M., Hills, J., and Bagnall, A. (2012). A shapelet transform for time series classification. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 289–297.

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

Oord, A. v. d., Li, Y., and Vinyals, O. (2018). Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*.

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

Salvador, S. and Chan, P. (2007). Toward accurate dynamic time warping in linear time and space. *Intelligent Data Analysis*, 11(5):561–580.

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

Van Der Maaten, L., Postma, E. O., Van Den Herik, H. J., et al. (2009). Dimensionality reduction: A comparative review. *Journal of Machine Learning Research*, 10(66-71):13.

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

Wei, W. W. (2018). *Multivariate time series analysis and applications*. John Wiley & Sons.

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

A Experimental setup

A.1 Datasets

The experimental results presented in this work are based on a diverse set of datasets, whose main characteristics are summarized in Table 3. 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 3: 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 Implementation Details

B.1 Foundation Models

For the `MOMENT` model, we utilized the HuggingFace checkpoint provided by the authors (Goswami et al., 2024). In contrast, for `ViT`, we implemented and trained the model ourselves, initially aiming to replicate the Nu-Time architecture (Lin et al., 2024), as the source code is currently unavailable. However, since we were unable to achieve comparable experimental results, our implementation diverges in certain aspects. Specifically, we extract overlapping patches from the time series, which are further embedded with statistical embeddings to form tokens that are processed by a transformer. During training, we employ a variant of the InfoNCE loss (Oord et al., 2018) proposed by He et al. (2020).

C Experimental Details

C.1 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 2. 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 4 and 5 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 2. This flexibility in the PCA configuration allows us to adapt the method to a wide range of tasks, optimizing both performance and computational efficiency.

Table 4: 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	$0.978_{\pm 0.000}$	$0.963_{\pm 0.001}$	$0.967_{\pm 0.001}$	$0.956_{\pm 0.001}$

C.2 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 3 show the performance comparison between *lcomb* and *lcomb_top_k* across several datasets for both MOMENT and `ViT` foundation models.

Table 5: Performance comparison between fine tuning methods with different adapter configurations for `ViT` 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	$0.962_{\pm 0.003}$	$0.952_{\pm 0.003}$	$0.921_{\pm 0.006}$	$0.899_{\pm 0.002}$

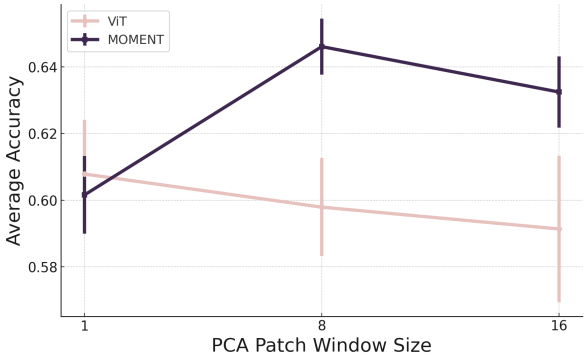
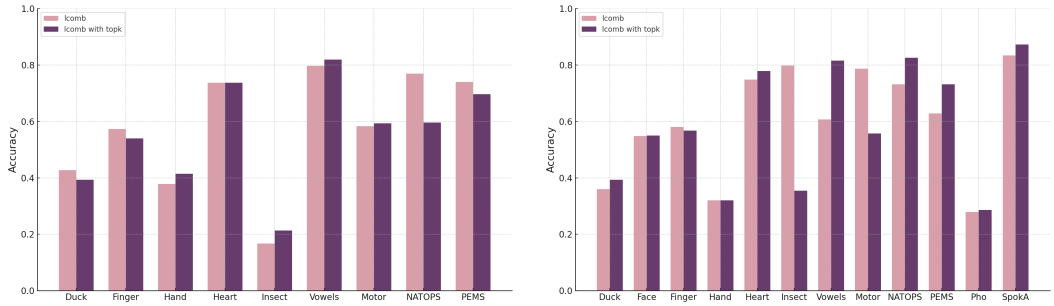


Figure 2: Comparison of PCA and PatchPCA Methods for `ViT` and `MOMENT` Models



(a) `MOMENT` (b) `ViT`

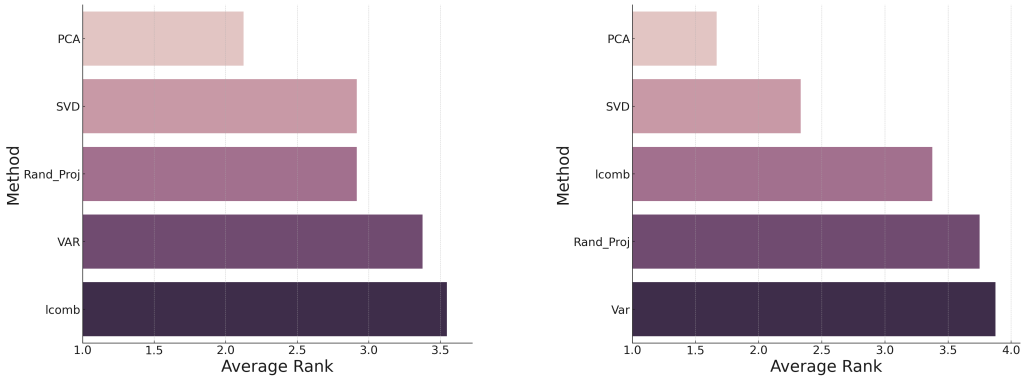
Figure 3: Performance Comparison Between *lcomb* and *lcomb_top_k* Fine-Tuning Configurations for both `MOMENT` and `ViT` Models

C.3 Rank Comparisons

Figure 4 shows a comparison of the average rank for different adapter methods used in the `MOMENT` and `ViT` 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 4a, 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 ViT foundation model (Figure 4b), *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) Adapter's Average Rank for MOMENT Foundation Model

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

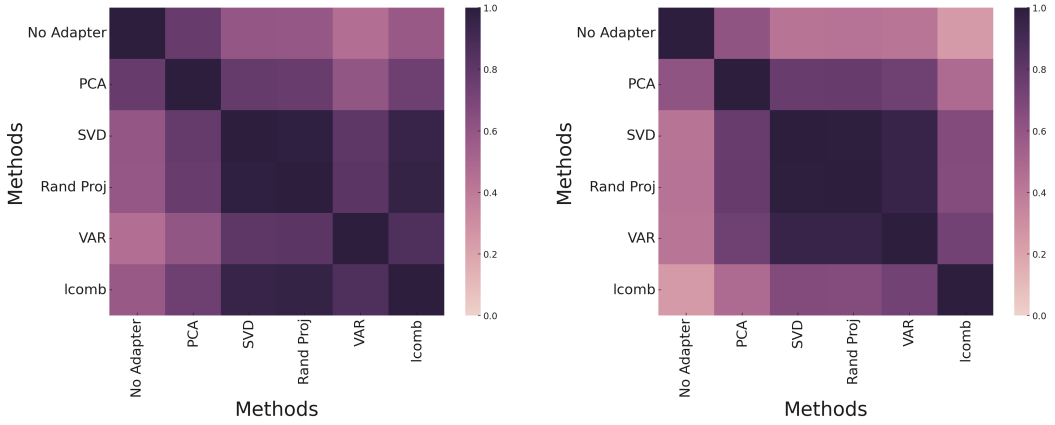
Figure 4: Comparison of Adapter's Average Rank for MOMENT and ViT Foundation Models averaged across all datasets and three different seeds

C.4 Statistical Tests

The heatmap shown in Fig. 5 present the pairwise p-values between different fine-tuning methods applied to the MOMENT and ViT 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 ViT, 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.

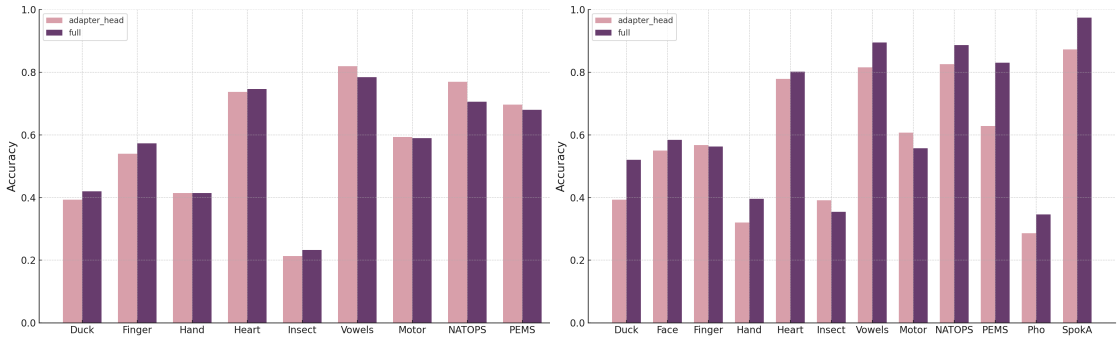
C.5 Full Fine-Tuning Regime



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

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

Figure 5: Heatmap of Pairwise p-values for Adapter Methods for `MOMENT` and `ViT` Foundation Models averaged across all datasets and three different seeds



(a) `MOMENT`

(b) `ViT`

Figure 6: Full fine-tuning vs tuning adapter+head for `lcomb`.