Mantis: Lightweight Calibrated Foundation Model for User-Friendly Time Series Classification

Anonymous Authors[1]

Abstract

In recent years, there has been increasing interest in developing foundation models for time series data that can generalize across diverse downstream tasks. While numerous forecasting-oriented foundation models have been introduced, there is a notable scarcity of models tailored for time series classification. To address this gap, we present **Mantis**, a new open-source foundation model for time series classification based on the Vision Transformer (ViT) architecture that has been pre-trained using a contrastive learning approach. Our experimental results show that Mantis outperforms existing foundation models both when the backbone is frozen and when fine-tuned, while achieving the lowest calibration error. In addition, we propose several adapters to handle the multivariate setting, reducing memory requirements and modeling channel interdependence.

1 Introduction

The advent of large foundation models (Bommasani et al., 2021) in computer vision (He et al., 2015; Dosovitskiy et al., 2021) and natural language processing (Achiam et al., 2023; Touvron et al., 2023) has significantly transformed research and applications. These models are pre-trained on extensive, diverse datasets to generalize across a wide range of downstream tasks, simplifying model design and reducing the need for large amounts of labeled data.

In recent years, foundation models for time series (TSFMs) have emerged as a growing area of research. For time series forecasting, many models have been proposed—either trained from scratch on large-scale data (Woo et al., 2024; Wang et al., 2024; Ansari et al., 2024) or adapted from pre-trained language models (Gruver et al., 2024). Some TSFMs target multiple tasks at once, such as forecasting, classification, and imputation (Zhou et al., 2023; Goswami et al., 2024). However, general-purpose designs may underperform on classification tasks, as pre-training objectives like masked reconstruction or next-step prediction are not always ideal for discriminative learning.

Despite the broad use of time series classification (Bagnall et al., 2018; Dau et al., 2019; Dempster et al., 2020), relatively few foundation models are tailored specifically for it. To address this gap, we introduce Mantis, a lightweight, high-performance foundation model for time series classification. Drawing on recent advances in time series representation learning (Eldele et al., 2021; Zhang et al., 2022) and transformer-based architectures (Nie et al., 2023; Ilbert et al., 2024; Lin et al., 2024), Mantis uses a patch-based encoder combined with a Vision Transformer (ViT) backbone (Dosovitskiy et al., 2021). It is contrastively pre-trained on 1.89M univariate time series and contains 8M parameters.

We conduct extensive experiments to demonstrate the superior performance and calibration of Mantis across a wide range of classification tasks. Additionally, to improve scalability for multivariate datasets, we explore lightweight channel-level adapters that reduce the number of input channels prior to encoding. While not the focus of this work, these adapters are described in Appendix D.

2 Methodology

In this section, we give the main technical details of Mantis.

2.1 Problem Setup

We aim to train a time series classification foundation model, an encoder $F : \mathbb{R}^t \to \mathbb{R}^q$, that maps a fixed-length univariate time series $\mathbf{x} \in \mathbb{R}^t$ to a latent representation in $\mathbb{R}^q$. During pre-training, we use a large unlabeled dataset X_0 to learn task-agnostic representations. In the fine-tuning stage, given a labeled dataset (X, Y), we either: (1) extract embeddings $\mathrm{Z} = \{F(\mathbf{x}) \,|\, \mathbf{x} \in \mathrm{X}\}$ to train a classifier $h : \mathbb{R}^q \to \{1, \ldots, K\}$, or (2) jointly fine-tune a classification head $h : \mathbb{R}^q \to \mathbb{R}^K$ atop F by minimizing supervised loss.

For multivariate time series $\mathbf{x} \in \mathbb{R}^{d \times t}$, each channel

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

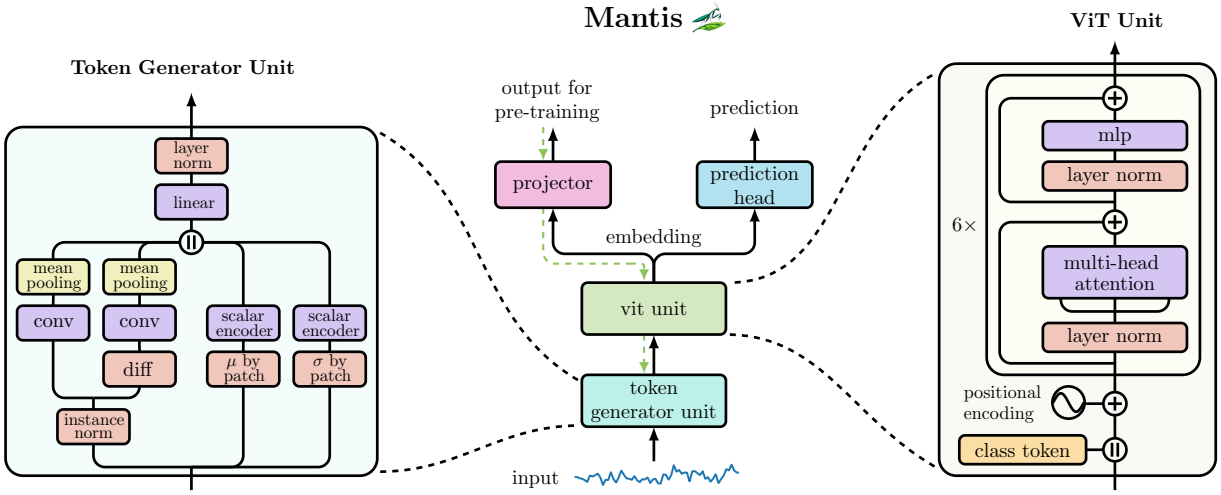

Figure 1: Architecture. By symbol + we denote the sum operator, while || designates the vector concatenation operator.

$\mathbf{x}_i$ is encoded independently: $\mathbf{z} = \text{concat}(F(\mathbf{x}_i))_{i=1}^{d} \in \mathbb{R}^{d \times q}$. To reduce dimensionality when d is large, we introduce an adapter $a : \mathbb{R}^{d \times t} \to \mathbb{R}^{d_{\text{new}} \times t}$ and define $\mathbf{z} = \text{concat}(F(a(\mathbf{x})_i))_{i=1}^{d_{\text{new}}}$.

To assess prediction confidence, we apply softmax $\sigma : \mathbb{R}^K \to \Delta_K$ to model outputs, and define confidence as $\text{conf}(\mathbf{x}) = \max[\sigma(h \circ F(\mathbf{x}))]$ and predicted label as $\hat{y} = \text{argmax}[\sigma(h \circ F(\mathbf{x}))]$.

2.2 Architecture

The architecture of Mantis is based the Vision Transformer (ViT, (Dosovitskiy et al., 2021)) adapted to the time series domain through a new token generator unit (see Figure 1).

Input Processing. To handle the diversity of time series inputs such as varying sequence lengths and unit scales, Mantis performs several pre-processing steps before token generation. Inspired by fixed-resolution inputs in computer vision, we provide the model with interpolated input sequences of length 512. To address unit variability, we apply instance-level standardization: for each input channel, we subtract the mean and divide by the standard deviation across time steps. This normalization is embedded directly into the model's forward pass.

Token Generation. After normalization, the sequence is encoded into tokens. Unlike prior approaches that split sequences directly into patches (Lin et al., 2024; Nie et al., 2023), we apply a 1D convolution followed by mean pooling to generate 32 patches, each representing 256 convolutional features. To enhance robustness to trends and improve stationarity, we also compute the first-order difference of the time series and process it similarly to create another set of 32 patches. To retain scale-specific information, we split the raw signal into 32 non-overlapping patches, compute per-patch mean and standard deviation, and encode them using the Multi-Scaled Scalar Encoder (Lin et al., 2024). The features from the original signal, its differential, and statistical encodings are concatenated, passed through a linear projector, and then normalized using layer normalization (Ba et al., 2016), yielding the final sequence of 32 tokens of dimension 256.

ViT Unit. These tokens are then passed to a ViT module. We prepend a learnable class token, which is used to aggregate information across the sequence. Positional information is added using sinusoidal embeddings (Vaswani et al., 2017). The resulting sequence of 33 tokens is processed by six standard transformer layers with 8 attention heads each. During pre-training, we apply dropout with a rate of 10%. The final output of the model is the class token embedding produced by the last transformer layer.

Projector and Prediction Head. Depending on the usage stage, the final embedding is passed through different heads. During pre-training, we use a linear projection on top of layer normalization to produce representations for contrastive similarity learning. During fine-tuning, we replace this with a classification head that outputs class logits for downstream tasks.

2.3 Pre-training

We pre-train Mantis in a self-supervised way using a contrastive learning approach that aims to train an encoder that outputs similar representations for two random augmentations of the same sample (positive pair) and dissimilar representations for augmentations of two different samples (negative pair). More formally, let $\mathcal{T}$ be a considered space of transformations (augmentations) such that $\forall \phi \in \mathcal{T}, \mathbf{x} \in \mathcal{X}$ we have $\phi(\mathbf{x}) \in \mathcal{X}$. To measure the similarity of two embed-

dings, we first project the output of the foundation model $F(\mathbf{x})$ to a new dimension using $g : \mathbb{R}^q \to \mathbb{R}^{q'}$ and then compute the cosine similarity between the two vectors:

$$s_{\cos}(\mathbf{a}, \mathbf{b}) := \frac{\mathbf{a}^\top \mathbf{b}}{\|\mathbf{a}\| \cdot \|\mathbf{b}\|}, \qquad \forall (\mathbf{a}, \mathbf{b}) \in \mathbb{R}^{2q'}.$$

Given a batch $B = \{\mathbf{x}_i\}_{i=1}^b$, for each example $\mathbf{x}_i$, we sample two augmentation functions ϕ and ψ uniformly from $\mathcal{T}$, i.e., $\phi, \psi \sim \mathcal{U}(\mathcal{T})$, compute the pairwise similarities between all the examples in the following way:

$$\mathbf{s}_i(\phi, \psi) = \left[s_{\cos}\left(g \circ F \circ \phi(\mathbf{x}_i),\, g \circ F \circ \psi(\mathbf{x}_j)\right) \right]_{j=1}^b \in \mathbb{R}^b.$$

Following (Oord et al., 2018) as well as (He et al., 2020) and denoting the cross-entropy error function by $l_{\text{ce}} : \mathbb{R}^b \times \{1, \ldots, b\} \to \mathbb{R}$, we update the weights of F and g by minimizing the contrastive loss which we define as

$$\sum_{i=1}^{b} l_{\text{ce}} \left(\frac{\mathbf{s}_i(\phi, \psi)}{T},\, i \right),$$

where $T \in (0, +\infty)$ is a temperature – fixed to 0.1. As augmentation, we use `RandomCropResize`, which randomly crops and then resizes a portion of the time series. We vary the crop rate between 0–20% to maintain structural integrity while introducing minor distortions.

Our pre-training dataset combines various public datasets, totalling 1.89M time series examples (see Appendix A.1). We pre-train the model for 100 epochs with a batch size equal to 2048 on 4 NVIDIA Tesla V100-32GB GPUs.

3 Adapters

To efficiently handle multivariate time series and reduce the computational cost of processing each channel independently, we explore lightweight channel-level adapters that project the original input into a reduced number of virtual channels before passing it to the encoder. These adapters are model-agnostic and preserve temporal structure while enabling better scalability.

We evaluate both unsupervised adapters – such as PCA, SVD, random projections, and variance-based channel selection – and a supervised adapter, the Differentiable Linear Combiner (LComb), which is trained jointly with the model. A full description of each adapter and corresponding experimental results is provided in Appendix D.

4 Experimental Results

In this section, we assess Mantis by conducting experiments, performed on a single NVIDIA Tesla V100-32GB GPU.

4.1 Setup

Baselines. We compare Mantis against four recent foundation models: UniTS (Gao et al., 2024), GPT4TS (Zhou et al., 2023), NuTime (Lin et al., 2024), and MOMENT (Goswami et al., 2024). In Appendix B.1, we give more details on their architectures, scales, and pre-training strategies.

Datasets. We evaluate on diverse univariate and multivariate datasets, including the full UCR archive (Dau et al., 2019), a subset of UEA (`UEA-27`, Bagnall et al., 2018), and other four real-world datasets. More details on datasets can be found in Appendix B.2.

Experiments. We assess Mantis across three tasks: (1) *Zero-shot* feature extraction using a frozen encoder followed by Random Forest learning on the embeddings, evaluated on 159 datasets (`159-D`); (2) *Full fine-tuning* of the entire model on 131 datasets (`131-D`); and (3) *Calibration* for uncertainty quantification performances, on the same `131-D` subset. GPT4TS and UniTS are excluded from the zero-shot setting as they do not support this regime.

Below we display the main experimental results. The ablation study and experiments on adapters can be found in Appendix C and D.

4.2 Zero-shot Feature Extraction Regime

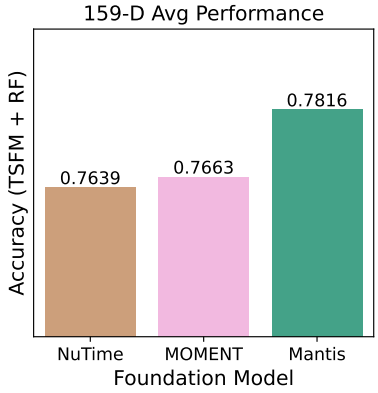

Figure 2: Accuracy of models with frozen encoder in average over 3 random seeds and `159-D` datasets.

In Figure 2, we can see that on average Mantis outperforms NuTime and MOMENT by 1.77% and 1.53%, respectively. Mantis is the best on 74 datasets, while MOMENT and NuTime have the best performance on 53 and 38 datasets, respectively. Compared to other foundation models, we can see that Mantis achieves the best balance in terms of the complexity of the model and its accuracy. On the one hand, Mantis outperforms a smaller model, NuTime, by increasing the encoder's capacity and leveraging more information from the pre-training set. On the other hand, a much bigger

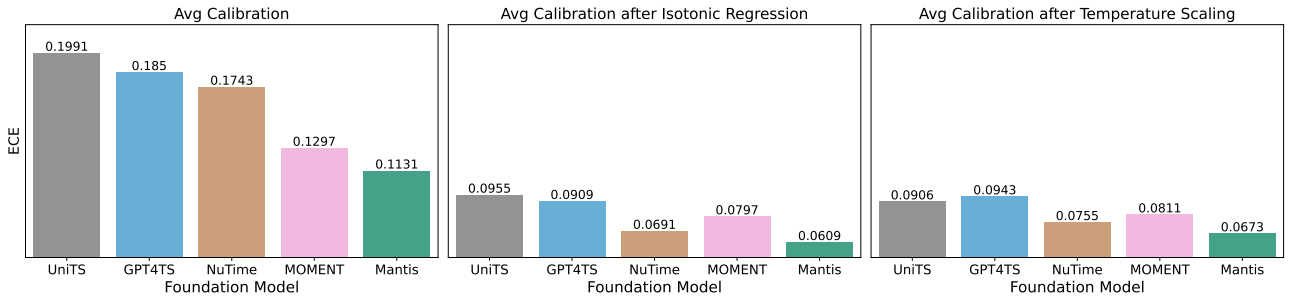

Figure 3: Averaged expected calibrated error of different methods over `131-D` datasets.

model, MOMENT, with 385 million parameters does not manage to achieve a better performance despite its size.

4.3 Full Fine-tuning Regime

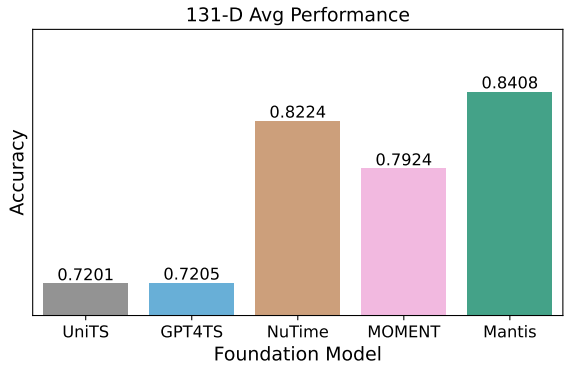

Figure 4: Accuracy of fine-tuned models in average over 3 random seeds and `131-D` datasets.

We evaluate the performance of foundation models in a full fine-tuning setting, where task-specific prediction heads are trained jointly with encoder layers (see Appendix B.3 for details). In Figure 4, one can see that Mantis has the highest performance among all the considered foundation models. We note that the reported performance results of our competitors may diverge from those reported by the authors of these models. This may be explained by the difference in the chosen fine-tuning scheme. In our case, we have fixed the scheme for a fair comparison while trying to find the best learning rate for every model based on a validation set. Thus, our experimental results reveal that Mantis is the most robust to the choice of a fine-tuning scheme and does not necessarily require tedious hyperparameter searching.

4.4 Calibration

Accurate uncertainty estimation is essential in safety-critical systems and for reliable evaluation (Wang, 2023; Xie et al., 2024). In this section, we assess the calibration of Mantis and other time series foundation models, focusing on how well their predicted confidences align with oberved accuracy (Guo et al., 2017). We use the Expected Calibration Error (ECE) (Naeini et al., 2015) to quantify miscalibration, averaging the gap between predicted confidence and empirical accuracy over 10 confidence bins (see Appendix B.4 for details). Following the fine-tuning setup from Section 4.3, we compute ECE over `131-D`, both before and after applying *post-hoc* calibration. We evaluate two standard *post-hoc* techniques: Temperature Scaling (Guo et al., 2017), which regulates the amplitude of logits before applying softmax, and Isotonic Regression (Zadrozny & Elkan, 2002), which adjusts predicted probabilities by fitting a piecewise constant function.

As shown in Figure 3, Mantis is the most calibrated model on average, even without post-hoc adjustment—an advantage when validation data is limited. All models benefit from calibration, but Mantis consistently retains the lowest ECE. Example of reliability diagrams are provided in Appendix, Figure 5.

5 Conclusion and Future Work

We introduced Mantis, a lightweight foundation model for time series classification, which outperforms larger models and achieves strong calibration. We also proposed an adapter-based fine-tuning strategy suitable for resource-constrained settings.

Our findings highlight key challenges in building foundation models for time series. Notably, we observed no clear link between model size and performance, and a substantial gap remains between zero-shot and fine-tuning accuracy. In addition, improving the interpretability of the model output is an important direction for future work. Specifically, ensuring strong calibration properties of the foundation model could enable various applications including uncertainty quantification and unsupervised performance prediction. Recently, Wen et al. (2024) showed that the hidden space itself can be performance-predictive when using contrastive pre-training, so exploring Mantis from this perspective could be promising as well.

References

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

Andrzejak, R. G., Lehnertz, K., Mormann, F., Rieke, C., David, P., and Elger, C. E. Indications of nonlinear deterministic and finite-dimensional structures in time series of brain electrical activity: Dependence on recording region and brain state. *Physical Review E*, 64(6):061907, 2001.

Anguita, D., Ghio, A., Oneto, L., Parra, X., Reyes-Ortiz, J. L., et al. A public domain dataset for human activity recognition using smartphones. In *Esann*, volume 3, pp. 3, 2013.

Ansari, A. F., Stella, L., Turkmen, C., Zhang, X., Mercado, P., Shen, H., Shchur, O., Rangapuram, S. S., Arango, S. P., Kapoor, S., et al. Chronos: Learning the language of time series. *arXiv preprint arXiv:2403.07815*, 2024.

Ba, J. L., Kiros, J. R., and Hinton, G. E. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

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

Benechehab, A., Hili, Y. A. E., Odonnat, A., Zekri, O., Thomas, A., Paolo, G., Filippone, M., Redko, I., and Kégl, B. Zero-shot model-based reinforcement learning using large language models. *arXiv preprint arXiv:2410.11711*, 2024.

Bommasani, R., Hudson, D. A., Adeli, E., Altman, R., Arora, S., von Arx, S., Bernstein, M. S., Bohg, J., Bosselut, A., Brunskill, E., et al. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*, 2021.

Chicaiza, K. O. and Benalcázar, M. E. A brain-computer interface for controlling IoT devices using EEG signals. In *2021 IEEE Fifth Ecuador Technical Chapters Meeting (ETCM)*, pp. 1–6. IEEE, 2021.

Clifford, G. D., Liu, C., Moody, B., Li-wei, H. L., Silva, I., Li, Q., Johnson, A., and Mark, R. G. AF classification from a short single lead ECG recording: The PhysioNet/computing in cardiology challenge 2017. In *2017 Computing in Cardiology (CinC)*, pp. 1–4. IEEE, 2017.

Dau, H. A., Bagnall, A., Kamgar, K., Yeh, C.-C. M., Zhu, Y., Gharghabi, S., Ratanamahatana, C. A., and Keogh, E. The UCR time series archive. *IEEE/CAA Journal of Automatica Sinica*, 6(6):1293–1305, 2019.

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

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. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2021.

Egede, J. O., Song, S., Olugbade, T. A., Wang, C., Amanda, C. D. C., Meng, H., Aung, M., Lane, N. D., Valstar, M., and Bianchi-Berthouze, N. Emopain challenge 2020: Multimodal pain evaluation from facial and bodily expressions. In *2020 15th IEEE International Conference on Automatic Face and Gesture Recognition (FG 2020)*, pp. 849–856. IEEE, 2020.

Eldele, E., Ragab, M., Chen, Z., Wu, M., Kwoh, C. K., Li, X., and Guan, C. Time-series representation learning via temporal and contextual contrasting. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, pp. 2352–2359, 2021.

Gao, S., Koker, T., Queen, O., Hartvigsen, T., Tsiligkaridis, T., and Zitnik, M. UniTS: A unified multi-task time series model. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

Goldberger, A. L., Amaral, L. A., Glass, L., Hausdorff, J. M., Ivanov, P. C., Mark, R. G., Mietus, J. E., Moody, G. B., Peng, C.-K., and Stanley, H. E. PhysioBank, PhysioToolkit, and PhysioNet: components of a new research resource for complex physiologic signals. *circulation*, 101(23):e215–e220, 2000.

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

Gruver, N., Finzi, M., Qiu, S., and Wilson, A. G. Large language models are zero-shot time series forecasters. *Advances in Neural Information Processing Systems*, 36, 2024.

Guo, C., Pleiss, G., Sun, Y., and Weinberger, K. Q. On calibration of modern neural networks. In *International conference on machine learning*, pp. 1321–1330. PMLR, 2017.

Halko, N., Martinsson, P.-G., and Tropp, J. A. Finding structure with randomness: Stochastic algorithms for constructing approximate matrix decompositions. *arXiv preprint arXiv:0909.4061*, 909, 2009.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*, 2015.

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

Ilbert, R., Odonnat, A., Feofanov, V., Virmaux, A., Paolo, G., Palpanas, T., and Redko, I. SAMformer: Unlocking the potential of transformers in time series forecasting with sharpness-aware minimization and channel-wise attention. In Salakhutdinov, R., Kolter, Z., Heller, K., Weller, A., Oliver, N., Scarlett, J., and Berkenkamp, F. (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 20924–20954. PMLR, 21–27 Jul 2024.

Kemp, B., Zwinderman, A. H., Tuk, B., Kamphuisen, H. A., and Oberye, J. J. Analysis of a sleep-dependent neuronal feedback loop: the slow-wave microcontinuity of the EEG. *IEEE Transactions on Biomedical Engineering*, 47 (9):1185–1194, 2000.

Lessmeier, C., Kimotho, J. K., Zimmer, D., and Sextro, W. Condition monitoring of bearing damage in electromechanical drive systems by using motor current signals of electric motors: A benchmark data set for data-driven classification. *PHM Society European Conference*, 3(1), 2016.

Lin, C., Wen, X., Cao, W., Huang, C., Bian, J., Lin, S., and Wu, Z. NuTime: Numerically multi-scaled embedding for large- scale time-series pretraining. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856. URL https://openreview.net/forum?id=TwiSBZ0p9u.

Liu, J., Zhong, L., Wickramasuriya, J., and Vasudevan, V. uWave: Accelerometer-based personalized gesture recognition and its applications. *Pervasive and Mobile Computing*, 5(6):657–675, 2009.

Loshchilov, I. and Hutter, F. SGDR: Stochastic gradient descent with warm restarts. In *International Conference on Learning Representations*, 2016.

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

Malekzadeh, M., Clegg, R. G., Cavallaro, A., and Haddadi, H. Mobile sensor data anonymization. In *Proceedings of the international conference on internet of things design and implementation*, pp. 49–58, 2019.

Middlehurst, M., Schäfer, P., and Bagnall, A. Bake off redux: a review and experimental evaluation of recent time series classification algorithms. *Data Mining and Knowledge Discovery*, pp. 1–74, 2024.

Naeini, M. P., Cooper, G., and Hauskrecht, M. Obtaining well calibrated probabilities using Bayesian binning. *Proceedings of the AAAI conference on artificial intelligence*, 29(1), 2015.

Nie, Y., H. Nguyen, N., Sinthong, P., and Kalagnanam, J. A time series is worth 64 words: Long-term forecasting with transformers. In *International Conference on Learning Representations*, 2023.

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

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21 (140):1–67, 2020.

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

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. u., and Polosukhin, I. Attention is all you need. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

Wang, C. Calibration in deep learning: A survey of the state-of-the-art. *arXiv preprint arXiv:2308.01222*, 2023.

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

Wen, S., Feofanov, V., and Zhang, J. Measuring pre-training data quality without labels for time series foundation models. *arXiv preprint arXiv:2412.06368*, 2024.

Woo, G., Liu, C., Kumar, A., Xiong, C., Savarese, S., and Sahoo, D. Unified training of universal time series forecasting transformers. In Salakhutdinov, R., Kolter, Z., Heller, K., Weller, A., Oliver, N., Scarlett, J., and Berkenkamp, F. (eds.), *Proceedings of the 41st International Conference*

on Machine Learning, volume 235 of *Proceedings of Machine Learning Research*, pp. 53140–53164. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/woo24a.html.

Xie, R., Odonnat, A., Feofanov, V., Deng, W., Zhang, J., and An, B. MANO: Exploiting matrix norm for unsupervised accuracy estimation under distribution shifts. *arXiv preprint arXiv:2405.18979*, 2024.

Zadrozny, B. and Elkan, C. Transforming classifier scores into accurate multiclass probability estimates. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 694–699, 2002.

Zhang, X., Zhao, Z., Tsiligkaridis, T., and Zitnik, M. Self-supervised contrastive pre-training for time series via time-frequency consistency. In Koyejo, S., Mohamed, S., Agarwal, A., Belgrave, D., Cho, K., and Oh, A. (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 3988–4003. Curran Associates, Inc., 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/194b8dac525581c346e30a2cebe9a369-Paper-Conference.pdf.

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

A Methodology – Additional Material

A.1 Pre-training datasets

Our pre-training dataset combines various public datasets, including UCR (Dau et al., 2019), UEA (Bagnall et al., 2018) (all except EigenWorms and InsectWingbeat), ECG (Clifford et al., 2017), EMG (Goldberger et al., 2000), Epilepsy (Andrzejak et al., 2001), FD-A and FD-B (Lessmeier et al., 2016), Gesture (Liu et al., 2009), HAR (Anguita et al., 2013), SleepEEG (Kemp et al., 2000). We ensure that test sets used for evaluation in Section 4 are not part of the pre-training dataset. The total pre-training consists of 1.89 million time series examples.

B Experimental Results – Additional Material

B.1 Baseline Details

We provide a brief overview of the four baseline models used in our experiments:

UniTS (Gao et al., 2024). UniTS is a lightweight multi-task foundation model designed for time series data. It has approximately 1 million parameters and is trained in a supervised fashion on 38 labeled datasets from UCR and UEA. Although originally built for multi-task learning, it serves as a competitive baseline for zero-shot and fine-tuning evaluations due to its broad exposure to diverse tasks during training.

GPT4TS (Zhou et al., 2023). GPT4TS adapts a pre-trained GPT-2 model to time series by introducing a learnable tokenization layer. The total number of parameters is around 80 million. We follow the authors' recommendations for partial fine-tuning.

NuTime (Lin et al., 2024). NuTime is a transformer-based foundation model for time series classification. It shares the same pre-training dataset as Mantis, allowing for a fair comparison. The model emphasizes efficiency and generalization from a relatively compact architecture.

MOMENT (Goswami et al., 2024). MOMENT is a large-scale time series foundation model based on the T5 architecture (Raffel et al., 2020), with approximately 385 million parameters. It is pre-trained on an extensive dataset containing 1.13 billion time series samples across multiple tasks. While highly expressive, its size results in longer inference times and higher computational demands.

Summary. These baselines provide a range of model sizes (from 1M to 385M parameters) and design philosophies (lightweight supervised models, large pre-trained transformers, and hybrid approaches). This diversity allows us to benchmark Mantis under varied conditions and usage settings.

B.2 Dataset Details

UCR Collection. The UCR Time Series Archive (Dau et al., 2019) consists of 128 univariate datasets spanning a wide range of domains such as sensor readings, ECG signals, motion capture, and simulated data. We use the entire collection for the zero-shot evaluation. For fine-tuning, 15 datasets are excluded due to memory limitations (e.g., long sequences or large sample sizes).

UEA Collection. The UEA archive (Bagnall et al., 2018) includes 30 multivariate time series datasets. We exclude three datasets with insufficient test samples or extremely short sequence lengths and downsample *InsectWingbeat* for efficiency. This yields 27 datasets, denoted `UEA-27`. For fine-tuning and calibration, an additional 11 datasets from UEA are excluded due to GPU memory constraints or excessive channel count. For ablation study of Mantis, 7 datasets from UEA are excluded due to excessive channel count.

Additional Datasets. We also include four real-world datasets used in prior time series foundation model evaluations:

- *Blink* (Chicaiza & Benalcázar, 2021): Eye-blink detection using EEG signals.
- *MotionSenseHAR* (Malekzadeh et al., 2019): Human activity recognition from smartphone motion sensors.
- *EMOPain* (Egede et al., 2020): Multimodal pain detection in physical therapy contexts.
- *SharePriceIncrease* (Middlehurst et al., 2024): Binary classification of share price movements.

These datasets are included in zero-shot evaluation, and two of them are included in fine-tuning and calibration experiments (excluding EMOPain and MotionSenseHAR due to memory constraints). For ablation study of Mantis, EMOPain is excluded due to eccessive number of channels.

Dataset Splits. We use the official train/test splits provided in the respective archives for all datasets. For fine-tuning and calibration experiments, 20% of the training set is further held out as a validation set for hyperparameter selection.

Summary.

- `159-D`: Used in zero-shot evaluation. Includes full UCR, `UEA-27`, and 4 additional datasets.
- `131-D`: Used in fine-tuning and calibration. Excludes datasets based on computational constraints of all models.
- `151-D`: Used for ablation study. Excludes datasets based on computational constraints of Mantis only – those with a large number of channels.

B.3 Full fine-tuning

In the full fine-tuning setup, we use the same procedure for all models, ensuring a fair comparison. More specifically, for each task, we append a prediction head after an encoder and fine-tune certain layers on the training data. For all the models, we fix a fine-tuning scheme: we minimize the cross-entropy loss for 100 epochs with a batch size equal to 256, using an AdamW optimizer (Loshchilov et al., 2017) with a weight decay of 0.05. For each model and each dataset, we test 3 learning rates $10^{-4}, 2 \cdot 10^{-4}, 10^{-3}$ and select the best using the validation set (with 80% - 20% as train - validation split). For GPT4TS, we follow their paper and fine-tune the layers specified by the authors, and fine-tuning the whole architecture for all the other models.

B.4 Calibration

We study the calibration properties of Mantis and other time series classification foundation models, and introduce here some formal definition. We say a model is calibrated if, for a given confidence level α, the probability that the model predicts the true class is equal to α (Guo et al., 2017):

$$\mathbb{P}(Y = \hat{y} \,|\, \text{conf}(\mathbf{X}) = \alpha) = \alpha, \quad \forall \alpha \in [0, 1],$$

where $\hat{y}$ and $\text{conf}(\mathbf{X})$ denote the predicted label and model's confidence of a random variable $\mathbf{X}$, respectively, which were formally defined in Section 2.1. We consider the Expected Calibration Error (ECE, (Naeini et al., 2015)) as a metric to evaluate the calibration on a test set $\{\mathbf{x}_i, y_i\}_{i=1}^n$. We discretize the interval $[0, 1]$ into m disjoint bins: $B_j \cap B_k = \emptyset$, $\forall j \neq k, 1 \leq j, k \leq m$ and $\cup_{j=1}^m B_j = [0, 1]$. As commonly done in the literature, we split the interval into 10 equally spaced bins. Then, denoting the predicted label for $\mathbf{x}_i$ by $\hat{y}_i$, the ECE is defined as follows:

$$\text{ECE} = \frac{1}{n} \sum_{j=1}^{m} \left| \sum_{\substack{1 \leq i \leq n \\ \text{conf}(\mathbf{x}_i) \in B_j}} \mathbb{I}(\hat{y}_i = y_i) - \text{conf}(\mathbf{x}_i) \right|.$$

C Ablation Study

In this section, we perform several ablation experiments for a more thorough analysis of Mantis. First, we validate some of our architecture choices. More specifically, we test whether the incorporation of features extracted from the differential of a time series, proposed in Section 2.2, improves the quality of embeddings. For this, we have pre-trained another version of our foundation model where the differential feature extraction part was removed from the Token Generator Unit. In Figure 6, we plot the average zero-shot feature extraction performance of the two versions of Mantis. One can see that the incorporation of the differential block noticeably improves the accuracy score. When comparing Figure 2 and Figure 6, it is interesting to notice that Mantis without the differential block still slightly outperforms NuTime and MOMENT in the case of zero-shot feature extraction.

In the second part of our ablation study, we raise the following questions: (a) is pre-training really useful, or it is sufficient to train Mantis from scratch on each dataset?, (b) should we fine-tune the head or the whole model?, (c) can Mantis be

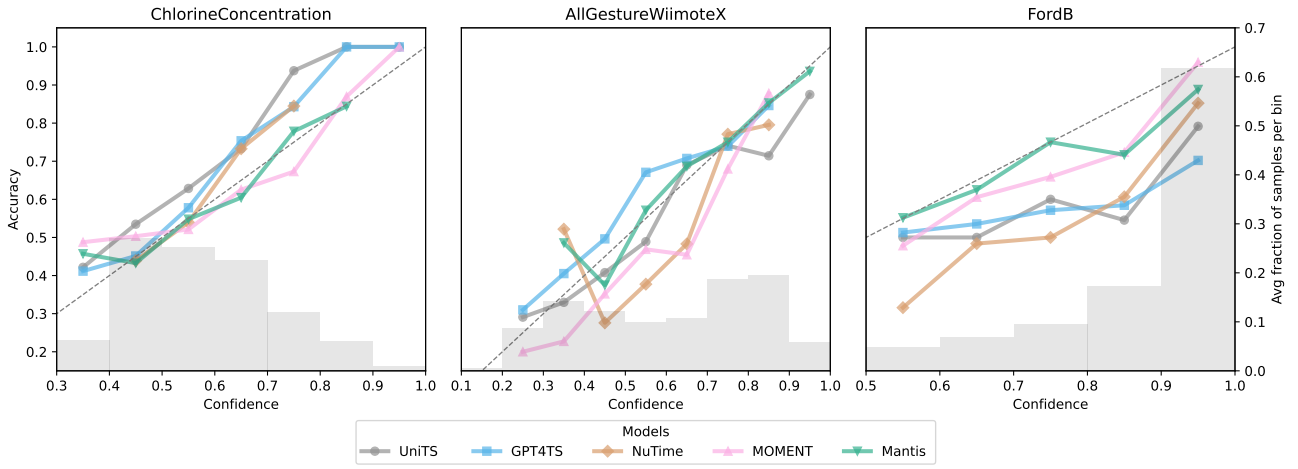

Figure 5: The reliability diagram on the test set after *post-hoc* temperature scaling on the validation set for three different datasets. The gray histogram illustrates the distribution of the confidence score across the test set.

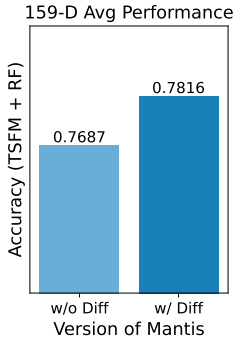

Figure 6: Averaged accuracy of Mantis w/ and w/o the differential block over 3 random seeds and `159-D` datasets.

fine-tuned with default hyperparameters, and how much we can gain from the model selection? To answer these questions, we empirically compare the following fine-tuning regimes:

- The setup which we further call RF, and which we used for the zero-shot feature extraction experiment: the encoder of Mantis is frozen, and embeddings are used as features to train a Random Forest classifier.
- The encoder of Mantis is frozen, and the task is solved by fine-tuning a classification head, which, in our case, is layer normalization step + linear layer. The purpose of comparing this strategy, further called `Head`, with `RF` is to see the impact of the choice of a classifier (linear vs non-linear, differentiable vs non-differentiable) on the overall performance.
- The encoder is randomly initialized and fine-tuned together with a classification head. This baseline, which we further call `Scratch`, is introduced to see whether or not pre-training of Mantis is useful.
- Finally, the strategy further called `Full` consists in fine-tuning the pre-trained encoder together with a classification head.

In order to answer question (c), for every method, we report two scores: the test accuracy of the model at the last epoch (i.e., at the end of fine-tuning), and the best test accuracy of the model across all fine-tuning epochs. While the last epoch score gives a pessimistic evaluation of the performance as no model selection is performed, the best epoch score is an optimistic evaluation as it indicates the best achievable performance. In this experiment, we fix the fine-tuning scheme: the number of

epochs is 100, the batch size is 256, the optimizer is AdamW with the weight decay of 0.05 and the learning rate set to $2 \cdot 10^{-4}$ and adjusted following the cosine annealing decay strategy (Loshchilov & Hutter, 2016) with 10 warm-up epochs.

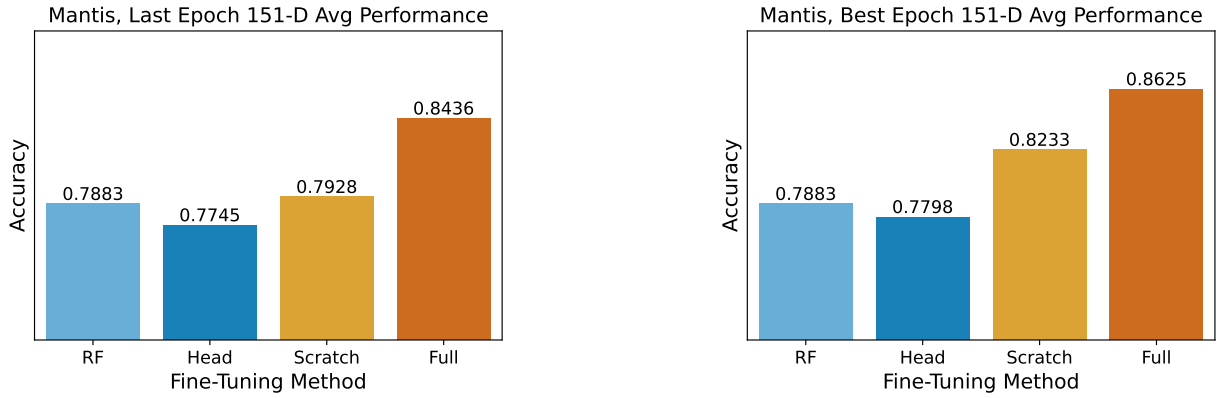

Figure 7: Performance of different fine-tuning methods averaged over 3 random seeds and `151-D` datasets.

Figure 7 depicts the average performance over the `151-D` benchmark both for the last and best epoch. First, we can see that the pre-training of Mantis is visibly useful as `Full` outperforms `Scratch` by 5.08% and 3.92% in the last and best epoch score, respectively. On the other hand, using a frozen pre-trained encoder is rather suboptimal in terms of performance, and full fine-tuning is preferred. This observation points out an important direction of future work, namely, to advance time series classification foundation models in terms of zero-shot performance.

Another observation from Figure 7 is that fine-tuning the head with a frozen encoder may be suboptimal, and training a Random Forest on embeddings gives a slightly better accuracy score. Finally, by comparing the performance results for the last and best epoch, we conclude that although model selection is important to improve the overall performance (+1.89% for `Full`), using directly the model from the last epoch gives already good performance. This shows that Mantis can be applied in practice without the need to tediously search for optimal values of hyperparameters.

D Adapters

D.1 Introduction and proposed approach

Handling multivariate time series is a key challenge for foundation models, since different classification tasks involve varying channel counts. Like other foundation models (Goswami et al., 2024; Lin et al., 2024), Mantis is pre-trained on univariate data and applied to multivariate settings by treating channels independently. However, this approach may demand excessive resources and ignore channel correlations during the feature extraction.

To address these concerns, we study a simple approach of using a channel-level adapter $a : \mathbb{R}^d \to \mathbb{R}^{d_{\text{new}}}$ that precedes the foundation model and transforms the original d channels into new d_{new} ones. We consider this definition of an adapter due to its high flexibility: (a) not only Mantis but any foundation model can be plugged in, (b) channel-level transformation prevents from disrupting the temporal structure, (c) adaptation to the computation budget by determining the number of encoded channels.

First adapters that we introduce are classical dimension reduction approaches like Principal Component Analysis that are computationally efficient and unsupervised. However, they are normally limited to 2D data matrices, so we train them on the data reshaped to $(n \times t,\ d)$, where n is the number of training examples. This allows us to focus on correlations between channels over all time steps, learning the rotation matrix $W \in \mathbb{R}^{d_{\text{new}} \times d}$ that linearly combines the original channels into new ones. Overall, these four adapters rely on this strategy:

- *Principal Component Analysis* (*PCA*) seeks to find an orthogonal basis of principal components where few components capture most of the data variance.

- *Truncated Singular Value Decomposition* (*SVD*, (Halko et al., 2009)) is similar to PCA, but it applies SVD decomposition directly on the non-centered data matrix.
- *Random Projection (Rand Proj)* is a trivial baseline that randomly generates the rotation matrix $W \in \mathbb{R}^{d_{\text{new}} \times d}$.
- *Variance-Based Channel Selection (Var Selector)* performs feature selection by keeping channels with the highest variance. This approach can be useful when channels with low variance can be discarded without affecting the overall performance.

Although these adapters are computationally very efficient, they are not supervised, which may lead to suboptimal performance. Therefore, we introduce one more adapter:

- *Differentiable Linear Combiner* (*LComb*) performs a linear projection to D channels by learning it via backpropagation together with the encoder and head.

D.2 Experimental results

Table 1: Performance comparison between different adapters when Mantis is fully-fine-tuned to a multi-channel time series classification task. The `UEA-27` data collection is considered in this experiment, and the results are averaged over 3 random seeds. The number of selected channels is fixed to be ≤ 10. `No Adapter` means that all channels are independently fed to the TSFM, and `NaN` in its performance results marks the cases when the model has not fitted to a single V100-32GB GPU card's memory. Best Result summarizes the performance when the best strategy per dataset is chosen.

	d	No Adapter	Standalone Adapter				Diff. Adapter	Best Result
			PCA	SVD	Rand Proj	Var Selector	LComb	
ArticularyWordRecognition	9	**0.9933**$_{\pm 0.0}$	0.9922$_{\pm 0.0019}$	0.9878$_{\pm 0.0038}$	0.9811$_{\pm 0.0038}$	**0.9933**$_{\pm 0.0}$	0.9744$_{\pm 0.0069}$	0.9933$_{\pm 0.0}$
BasicMotions	6	**1.0**$_{\pm 0.0}$	**1.0**$_{\pm 0.0}$	**1.0**$_{\pm 0.0}$	**1.0**$_{\pm 0.0}$	**1.0**$_{\pm 0.0}$	**1.0**$_{\pm 0.0}$	1.0$_{\pm 0.0}$
CharacterTrajectories	3	0.9928$_{\pm 0.0004}$	**0.9947**$_{\pm 0.0004}$	0.9923$_{\pm 0.0012}$	0.9912$_{\pm 0.0016}$	0.9928$_{\pm 0.0004}$	0.993$_{\pm 0.0018}$	0.9947$_{\pm 0.0004}$
Cricket	6	**1.0**$_{\pm 0.0}$	0.9861$_{\pm 0.0}$	0.9769$_{\pm 0.008}$	0.9722$_{\pm 0.0}$	**1.0**$_{\pm 0.0}$	0.9907$_{\pm 0.008}$	1.0$_{\pm 0.0}$
DuckDuckGeese	1345	NaN	0.5733$_{\pm 0.0115}$	**0.6**$_{\pm 0.02}$	0.54$_{\pm 0.0872}$	0.5133$_{\pm 0.0231}$	0.54$_{\pm 0.0693}$	0.6$_{\pm 0.02}$
ERing	4	**0.9926**$_{\pm 0.0074}$	0.9778$_{\pm 0.0064}$	0.9753$_{\pm 0.0113}$	0.9642$_{\pm 0.0043}$	**0.9926**$_{\pm 0.0074}$	0.9778$_{\pm 0.0064}$	0.9926$_{\pm 0.0074}$
EigenWorms	6	0.8372$_{\pm 0.0044}$	0.8448$_{\pm 0.0117}$	**0.8601**$_{\pm 0.0192}$	0.8117$_{\pm 0.0233}$	0.8372$_{\pm 0.0044}$	0.8066$_{\pm 0.0384}$	0.8601$_{\pm 0.0192}$
Epilepsy	3	**1.0**$_{\pm 0.0}$	0.9976$_{\pm 0.0042}$	0.9976$_{\pm 0.0042}$	0.9976$_{\pm 0.0042}$	**1.0**$_{\pm 0.0}$	**1.0**$_{\pm 0.0}$	1.0$_{\pm 0.0}$
EthanolConcentration	3	**0.4208**$_{\pm 0.0195}$	0.2928$_{\pm 0.0101}$	0.3029$_{\pm 0.0122}$	0.384$_{\pm 0.0503}$	**0.4208**$_{\pm 0.0195}$	0.4081$_{\pm 0.0275}$	0.4208$_{\pm 0.0195}$
FaceDetection	144	NaN	0.6026$_{\pm 0.0054}$	0.6064$_{\pm 0.0037}$	0.5638$_{\pm 0.0065}$	0.5696$_{\pm 0.0027}$	**0.6272**$_{\pm 0.013}$	0.6272$_{\pm 0.013}$
FingerMovements	28	NaN	0.5833$_{\pm 0.0058}$	0.57$_{\pm 0.06}$	0.5467$_{\pm 0.0289}$	**0.6167**$_{\pm 0.0058}$	0.58$_{\pm 0.0265}$	0.6167$_{\pm 0.0058}$
HandMovementDirection	10	0.4009$_{\pm 0.0206}$	**0.5135**$_{\pm 0.027}$	0.482$_{\pm 0.0546}$	0.4279$_{\pm 0.0512}$	0.4009$_{\pm 0.0206}$	0.4414$_{\pm 0.0624}$	0.5135$_{\pm 0.027}$
Handwriting	3	0.482$_{\pm 0.0157}$	0.4529$_{\pm 0.0129}$	0.4588$_{\pm 0.0224}$	0.4235$_{\pm 0.0418}$	0.482$_{\pm 0.0157}$	**0.5839**$_{\pm 0.0283}$	0.5839$_{\pm 0.0283}$
Heartbeat	61	NaN	0.7561$_{\pm 0.0098}$	0.7626$_{\pm 0.0123}$	0.7707$_{\pm 0.0098}$	**0.7951**$_{\pm 0.0129}$	0.774$_{\pm 0.0123}$	0.7951$_{\pm 0.0129}$
InsectWingbeatSubset	200	NaN	0.4703$_{\pm 0.0051}$	0.4733$_{\pm 0.0127}$	0.4803$_{\pm 0.0302}$	**0.591**$_{\pm 0.005}$	0.236$_{\pm 0.0125}$	0.591$_{\pm 0.005}$
JapaneseVowels	12	**0.9811**$_{\pm 0.0054}$	0.9802$_{\pm 0.0031}$	**0.9811**$_{\pm 0.0}$	0.9577$_{\pm 0.0271}$	0.9784$_{\pm 0.0047}$	0.9577$_{\pm 0.0128}$	0.9811$_{\pm 0.0054}$
LSST	6	**0.7109**$_{\pm 0.0015}$	0.6795$_{\pm 0.0027}$	0.6894$_{\pm 0.0021}$	0.6929$_{\pm 0.0207}$	**0.7109**$_{\pm 0.0015}$	0.6941$_{\pm 0.0053}$	0.7109$_{\pm 0.0015}$
Libras	2	0.9389$_{\pm 0.0}$	0.937$_{\pm 0.0032}$	0.9481$_{\pm 0.0064}$	0.8111$_{\pm 0.1392}$	0.9389$_{\pm 0.0}$	**0.9704**$_{\pm 0.0032}$	0.9704$_{\pm 0.0032}$
MotorImagery	64	NaN	0.5933$_{\pm 0.0351}$	0.5833$_{\pm 0.0208}$	0.5867$_{\pm 0.0379}$	**0.6**$_{\pm 0.01}$	0.59$_{\pm 0.0173}$	0.6$_{\pm 0.01}$
NATOPS	24	0.937$_{\pm 0.0116}$	0.9537$_{\pm 0.0064}$	**0.9611**$_{\pm 0.0}$	0.8926$_{\pm 0.0032}$	0.8981$_{\pm 0.0116}$	0.8796$_{\pm 0.017}$	0.9611$_{\pm 0.0}$
PEMS-SF	963	NaN	0.8536$_{\pm 0.0067}$	0.8304$_{\pm 0.0145}$	0.7457$_{\pm 0.0473}$	**0.9114**$_{\pm 0.0067}$	0.7476$_{\pm 0.0219}$	0.9114$_{\pm 0.0067}$
PhonemeSpectra	11	0.3421$_{\pm 0.0023}$	0.3351$_{\pm 0.009}$	0.3215$_{\pm 0.0052}$	0.3492$_{\pm 0.0033}$	0.344$_{\pm 0.0052}$	**0.3547**$_{\pm 0.0047}$	0.3547$_{\pm 0.0047}$
RacketSports	6	**0.9408**$_{\pm 0.0}$	0.9123$_{\pm 0.0152}$	0.9101$_{\pm 0.0076}$	0.9167$_{\pm 0.0038}$	**0.9408**$_{\pm 0.0}$	0.9254$_{\pm 0.0038}$	0.9408$_{\pm 0.0}$
SelfRegulationSCP1	6	0.9135$_{\pm 0.0071}$	**0.917**$_{\pm 0.0052}$	0.9113$_{\pm 0.0034}$	0.9135$_{\pm 0.0079}$	0.9135$_{\pm 0.0071}$	0.901$_{\pm 0.009}$	0.917$_{\pm 0.0052}$
SelfRegulationSCP2	7	0.5389$_{\pm 0.0096}$	0.5648$_{\pm 0.0449}$	**0.5685**$_{\pm 0.0251}$	0.5611$_{\pm 0.0455}$	0.5389$_{\pm 0.0096}$	0.5482$_{\pm 0.021}$	0.5685$_{\pm 0.0251}$
SpokenArabicDigits	13	0.987$_{\pm 0.0009}$	**0.9933**$_{\pm 0.0014}$	0.9906$_{\pm 0.0019}$	0.988$_{\pm 0.0027}$	0.9864$_{\pm 0.0028}$	0.9882$_{\pm 0.0016}$	0.9933$_{\pm 0.0014}$
UWaveGestureLibrary	3	**0.9438**$_{\pm 0.0108}$	0.8583$_{\pm 0.0079}$	0.8552$_{\pm 0.0095}$	0.8635$_{\pm 0.0737}$	**0.9438**$_{\pm 0.0108}$	0.9177$_{\pm 0.0048}$	0.9438$_{\pm 0.0108}$

In this section, we study the performance of Mantis when it is combined with one of the adapters we introduced above. For `PCA`, `SVD`, `RandProj` and `Var Selector`, the new number of channels is equal to $D' = \min\{D, 10\}$, while for LComb is always fixed to $D' = 10$. Table 1 depicts the empirical comparison between the adapters and the case when we treat all channels independently (tagged as `No Adapter`). Below, we discuss the experimental results.

Dimension reduction: When the number of channels is too large, the full fine-tuning is limited as treating all channels independently is computationally costly. In Table 1, one can see that starting from 28 channels (fixing the batch size to 256),

the optimization of Mantis does not fit the memory of a single V100-32GB GPU card, which results in 7 `NaN` values in the `No Adapter` column. Nevertheless, applying one of the five adapters solves the issue, allowing the full fine-tuning and increasing the performance.

Channel interactions: Adapters also mix the original channels allowing them to interact before sending them to the encoder (Benechehab et al., 2024). Our experiments reveal that the utility of adapters from this perspective is rather dataset-dependent: while for datasets like RacketSports and UWaveGestureLibrary, adapters do not bring any improvement compared `No Adapter`, transforming original channels brings a large improvement for datasets like Handwriting and HandMovementDirection.

Channel selection: In situations where certain channels are not useful and can be simply discarded, `Var Selector` has the highest performance, which we can observe on FingerMovements and InsectWingbeat datasets.

Differentiable adapter: Although `LComb` yields high performance on some datasets, overall, it is less efficient than `PCA`, `SVD` and `Var Selector`. Since `LComb` is fine-tuned together with the encoder and the head, we suggest that the optimization of the adapter is intricate, so searching for a better optimization scheme can be a good future work.