Multimodal Fusion Strategies for Multivariate Time Series Classification

Samuel Thomaz Bastos[1], **Rafael da Costa Silva**[1], **Diego Furtado Silva**[1]

[1]Instituto de Ciências Matemáticas e Computação (ICMC) - Universidade de São Paulo (USP)
São Carlos - SP - Brazil

samuelthomaz@usp.br, rafael.csilva@usp.br, diegofsilva@icmc.usp.br

Abstract. *Time series classification (TSC) is relevant in various domains, including finance, healthcare, and human activity recognition. While deep learning methods have achieved state-of-the-art performance in TSC, most advancements focus on univariate or straightforward adaptations for multivariate time series, neglecting different inter-variable dependencies. This problem increases in scenarios where the data comes from multiple sensors. In this case, the classifier does not consider the particularities of each source. For instance, consider a motion recognition dataset containing data from accelerometers and gyroscopes. A simple architecture for multivariate TSC applies the same transformations, e.g., kernel convolutions, to both sensors, even if they naturally present different characteristics. This paper argues that treating the variables of distinct sources as separate modalities, akin to sensor data, and employing specialized models for each source can enhance the classification performance. To evaluate this claim, we propose treating sensor inputs as multimodal data, applying fusion techniques to integrate multi-source multivariate time series classifiers. Experimental analysis shows that our approach outperforms traditional single-model architectures, particularly in complex tasks. In these worst scenarios for the single-modality models, our proposal achieves accuracy up to twelve percentage points higher, achieving the best results in the literature. It highlights the importance of considering multi-source dependencies in TSC, highlighting the potential of fusion-based strategies to advance the multivariate time series classification field.*

1. Introduction

Time series classification (TSC) has implications across various domains, including finance, healthcare, industrial monitoring, and human activity recognition. Given its importance, numerous algorithms have been proposed to tackle this task effectively [Yeh et al. 2018, Mohammadi Foumani et al. 2024].

Deep learning techniques for TSC have emerged prominently over the past decade. The seminal work of Wang et al. [Wang et al. 2017] established the first focused study on developing "standard" deep architectures for time series, presenting insights into the strengths and weaknesses of diverse approaches, including the classical Multi-Layer Perceptron and different convolutional-based networks. While InceptionTime [Ismail Fawaz et al. 2020] and its more recent variations [Ismail-Fawaz et al. 2022, Ismail-Fawaz et al. 2025] achieve the state-of-the-art of

deep learning for TSC [Middlehurst et al. 2024], other architectures, such as Fully Convolutional Network (FCN), remain a cornerstone among the proposed architectures due to their cost-benefit trade-off [Medeiros Júnior et al. 2024].

Parallel to these efforts, the related literature has incorporated many relevant topics to the TSC development, such as Neural Architecture Search [Rakhshani et al. 2020], adversarial attacks [Fawaz et al. 2019], ensemble methods [Ismail Fawaz et al. 2019], self-supervised learning [Shi et al. 2021], and knowledge distillation [Ay et al. 2022]. Despite these advancements, most fundamental developments in the field only concern univariate time series. Consequently, most adaptations of TSC architectures for multivariate time series involve straightforward modifications of univariate methods. For instance, state-of-the-art approaches, primarily based on convolutional neural networks, often apply 1D convolution layers across dimensions or employ 2D convolutions that aggregate all dimensions simultaneously.

However, these straightforward adaptations have limitations. Notably, they disregard the dependency between the variables. They treat all dimensions as independent (e.g., 1D convolutions) or dependent (e.g., 2D convolutions that comprise all variables in the same operation). Recognizing these constraints, some studies have proposed architectures tailored specifically for multivariate time series. Notable examples include the attention-based multivariate convolutional neural network (AT-MVCNN) [Tripathi and Baruah 2020] and the Disjoint-CNN [Foumani et al. 2021], which introduce varied mechanisms to address dependencies between variables more effectively. Besides, InceptionTime and the ideas behind its proposal support the proposal of several recent models [Mohammadi Foumani et al. 2024]. It incorporates the multivariate scenario into its design by leveraging Inception modules that combine 1D convolutions to capture temporal patterns within each variable and bottleneck 1D convolutions along the variable axis to model inter-variable relationships. These operations are performed in parallel and concatenated for subsequent layers, offering a robust mechanism for handling multivariate data.

In this paper, we highlight that many real-world applications extend beyond simple multivariate data to scenarios involving multi-source multivariate data. For example, human activity recognition often relies on multiple sensors, such as accelerometers and gyroscopes, while patient monitoring may involve diverse physiological signals. We argue that treating subsets of variables, considering each source as a separate modality, can yield better performance by allowing for specialized models for each source, which are subsequently aggregated into a unified decision. Although these observations may seem self-evident, the literature usually disregards this assumption.

To evaluate our hypothesis, we propose leveraging simple fusion techniques to address multi-source time series. Our approach treats sensor data as multimodal inputs, applying fusion strategies to integrate them effectively, such as the framework illustrated in Figure 1.

A comparative analysis against off-the-shelf deep learning architectures demonstrates that our method outperforms single-model approaches, particularly as task complexity increases. Particularly, we demonstrate that simple and well-established fusion strategies can improve accuracy and stability in the most challenging datasets.

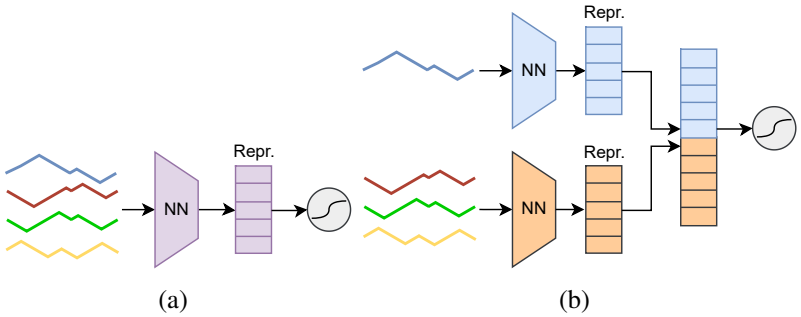


Figure 1. Consider a scenario of data collected by two sensors, where one sensor gathers an unimodal time series, and the other collects a three-dimensional time series. In a traditional approach (a), the neural model (NN) would deal with all these dimensions as a single multivariate data, attempting to model their relations in a single classification flow. This work evaluates how modeling the data from separate sensors by separate networks and fusing their representations (Repr.) to serve as input to a single decision layer (b) affects the performance.

2. Background and Related Work

In this section, we introduce the main definitions and related work that support our proposal. A `time series` can be informally defined as a set of observations over a period of time. Thus, a time series $\mathbf{S}$ of size m can be defined as an ordered sequence of observations:

$$\mathbf{S} = (s_1, \ s_2, \ \ldots, \ s_m) \ \ s_i \in \mathbb{R}^d \ \ \forall \ \ 1 \leq i \leq m \tag{1}$$

where s_i represents an observation of the series at the i^{th} moment and d represents the number of dimensions in that observation. Therefore, the univariate time series can be defined with $d = 1$, and the multivariate time series can be defined with any value greater than 1, increasing the number of dimensions in observation in the series.

Due to the complexity and particularities of this type of data, the main challenge in time series classification lies in effectively capturing temporal dependencies and patterns within the data to achieve accurate classification results. The literature in TSC has developed diverse algorithms to overtake this challenge with considerably different approaches, such as distance-based, feature extraction, dictionary extraction, random kernels, and deep learning [Middlehurst et al. 2024]. The more formal and focused study of neural networks for time series classification is relatively recent. A 2017 paper is considered a pioneer in this domain [Wang et al. 2017], as it proposed neural architectures for the task without a specific application. This study introduces a simple yet robust baseline using artificial neural networks, such as the Fully Convolutional Network (FCN). In the last few years, several architectures were proposed for the task. Particularly, Convolution-based networks (CNNs) dominate the state-of-the-art in deep learning-based time series classification [Mohammadi Foumani et al. 2024].

InceptionTime [Ismail Fawaz et al. 2020] is an outstanding architecture based on an ensemble of deep Convolutional Neural Networks (CNN) that has become well-established in the field. It performs similarly to the ensemble-based HIVE-COTE in accu-

racy while being significantly more scalable [Middlehurst et al. 2024]. Some of its evolutions also called the community's attention, such as LITE [Ismail-Fawaz et al. 2025], a lightweight time series classification model with only 2.34% of InceptionTime's parameters, utilizing DepthWise Separable Convolutions (DWSC) to maintain performance. Enhanced by multiplexing, custom filters, and dilated convolutions, LITE is 2.78 times faster and significantly more energy-efficient, reducing CO2 and power consumption by 2.79 times compared to InceptionTime.

Considering this scenario, this section proceeds with the definition of CNNs and the mechanisms cited in the architectures that will serve as the foundation for this work.

2.1. CNNs and Mechanisms Used in TSC

A Convolutional Neural Network layer is a fundamental building block of CNNs, designed to extract spatial and temporal features from data, such as images or time series. It uses a convolution operation, where a filter (or kernel) is applied to the input data to produce feature maps that highlight important patterns like edges, textures, or movements. Dilated convolutions expand the receptive field of the filter by introducing gaps (dilation) between its elements, allowing the network to capture a larger context without increasing the number of parameters. Convolutional neural network layers can include custom filters with fixed weights that do not change during training, enabling the network to extract specific predefined features.

Many architectures developed to deal with multivariate time series use a bottleneck layer. It consists of a convolution layer with a kernel size of 1, used to reduce the dimensionality of the input before applying larger convolutions, reducing the computational cost.

As cited, LITE (as well as other architectures) uses depthwise separable convolutions. Depthwise, pointwise, and depthwise separable convolutions are efficient variations of standard convolutional operations designed to reduce computational complexity while maintaining performance. Depthwise convolution applies a single convolutional filter to each input channel independently, extracting spatial features within each channel but not combining information across channels. Pointwise convolution, typically implemented as a 1x1 convolution, operates on the depth (channel dimension) by linearly combining the output of the depthwise convolution across channels, enabling inter-channel feature interactions. Combining these two operations sequentially forms a DepthWise Separable Convolution (DWSC), where the depthwise step captures spatial information and the pointwise step fuses channel information.

Multiplexing involves applying multiple convolutional filters with different kernel sizes to the time series data, each designed to focus on different temporal patterns, trends, or periodicities. The outputs of these filters, known as feature maps, are combined into a multidimensional tensor, effectively encoding the integrated information.

Ensembling is a technique that combines predictions from multiple models to improve accuracy and robustness by leveraging their diverse strengths. In deep learning, ensembling models with different initial weights enhances generalization, reduces overfitting, and captures varied data patterns. This approach has been critical for time series classification, as it improves reliability and performance [Ismail Fawaz et al. 2019]. State-of-the-art deep learning models like LITE and InceptionTime have successfully uti-

lized ensembles, demonstrating significant accuracy gains across benchmark datasets. These examples highlight ensembling as a key strategy in advanced time series classification.

2.2. Fusion Strategies

Multimodal learning data fusion involves integrating information from multiple sources or modalities, such as text, images, audio, and sensor data, to improve the performance of machine learning models. This approach leverages the complementary strengths of different data types to provide a more comprehensive understanding of the underlying patterns and relationships. By combining diverse data sources, multimodal learning can enhance the robustness and accuracy of models, facilitate better generalization to new tasks, and enable more sophisticated decision-making processes [Lahat et al. 2015].

In multimodal deep learning, early, intermediate, and late fusion integrate different data types to enhance model performance. Early fusion combines raw data at the input level, capturing low-level interactions but struggling with diverse data. Intermediate fusion processes each modality separately before merging, offering specialized handling but potentially missing early interactions. Late fusion combines outputs from separate models, preserving modality-specific strengths but foregoing early joint representations. Each strategy presents limitations and strengths and is selected based on the task's requirements [Lahat et al. 2015]. Figure 2 presents a schematic representation of architectures utilizing fusion.

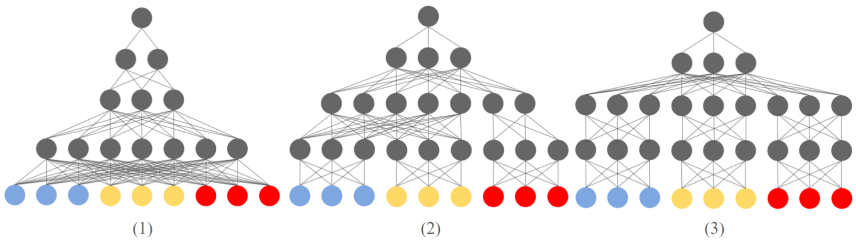


Figure 2. In the figure, each color represents a different modality. In (1), we have an early fusion, where the information is merged in the first hidden layer. In (2), we have an intermediate fusion, where the modalities are combined in an intermediate layer. Finally, in (3), we have a late fusion, where the information is merged only in the last dense layer.

2.3. Used Architectures

To ensure a representative and meaningful comparison, we selected three well-established architectures for time series classification. The Fully Convolutional Network (FCN) was included due to its simplicity and consistently competitive performance reported in the literature, making it a strong baseline. InceptionTime was chosen as it is widely regarded as one of the most accurate and robust architectures in this domain. Finally, we included LITETime, a more recent model that builds upon InceptionTime with the goal of improving efficiency while maintaining or enhancing classification performance. All parameters of all networks were kept similar to the original papers.

2.3.1. Fully Convolutional Network (FCN)

The FCN [Wang et al. 2017] architecture is composed of three blocks. Each block comprises a convolutional layer and a batch normalization layer, and applies ReLU activation. The convolution operations utilize three 1-D kernels of sizes 8, 5, 3 without striding, ensuring all input features are fully covered. The network is built by stacking three such blocks, with the number of filters 128, 256, 128 in each block. Batch normalization is incorporated to accelerate convergence and enhance generalization. After the convolutional blocks, a global average pooling layer projects the resulting representation to a 128-dimensional space, used by the final softmax decision layer. This architecture was used because it is a very common benchmark due to its simplicity and performance.

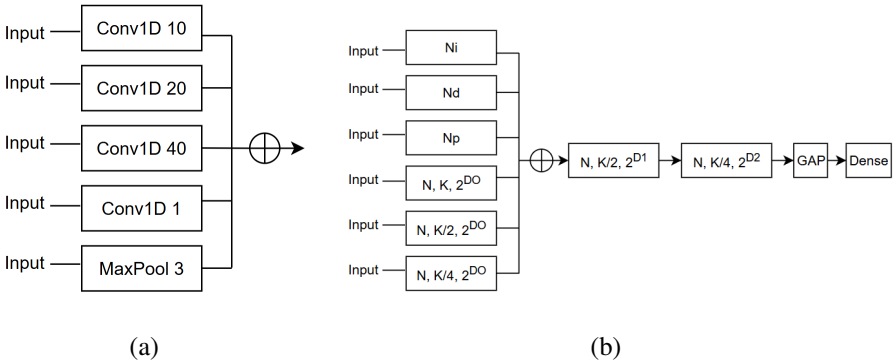
2.3.2. InceptionTime

InceptionTime [Ismail Fawaz et al. 2020] is a deep learning model designed specifically for time series classification tasks, inspired by the Inception module from computer vision [Szegedy et al. 2017]. InceptionTime incorporates multi-scale convolutions to effectively extract features from time series data at different temporal resolutions. The architecture is built using multiple parallel convolutional filters of varying lengths, allowing the model to capture both short-term and long-term patterns.

The Inception module consists of several components that extract and combine features from multiple temporal scales. It features multiple 1D convolutional layers with varying kernel sizes, such as 10, 20, and 40, that operate in parallel. This design enables the network to learn features across different time scales simultaneously. To reduce computational costs, a bottleneck layer, implemented as a 1×1 convolution, is used to reduce the input dimensionality before applying larger convolutional filters. In this configuration, max pooling is also incorporated as an additional branch to capture global patterns. The outputs of all convolutional branches are then concatenated along the channel dimension, combining the features learned at different scales. A residual connection is added from the input to the output of the module to improve gradient flow and stabilize training after three inception blocks. 5 models were used in an ensemble with 8 Inception blocks each. The ensemble of 5 Inception networks is called InceptionTime. The block's configuration, shown in Figure 3 (a).

2.3.3. LITETime

The LITE [Ismail-Fawaz et al. 2025] architecture aims to reduce the number of parameters while maintaining the performance of InceptionTime by leveraging custom kernels, multiplexing, dilation, and depthwise separable convolutions (DWSC). First, custom filters (increasing, decreasing, and peak filters) operate in parallel to the primary convolutional layer to extract distinct patterns from the input. Second, multiplexing convolutions are used in the first layer, employing three parallel convolutional layers to detect patterns of varying characteristics. Third, the second and third layers incorporate dilated convolutions, allowing the network to capture broader temporal contexts without increasing the kernel size.



(a) (b)

Figure 3. Figure (a) shows a representation of the Inception block, while Figure (b) shows a representation of the LITE block.

In the original, unidimensional proposal, standard convolutions are used in the first layer instead of DWSC because the input time series is univariate, and DWSC would only learn a single filter in this scenario. Subsequent layers, however, utilize DWSC to reduce parameters efficiently. However, LITE can be used for multivariate time series by allowing the model to learn a filter per channel and then learn how to combine in the PointWise Convolution step.

The block's configuration, shown in Figure 3 (b), uses the following parameter setup: For increasing filters (N_i) with kernel sizes: [2, 4, 8, 16, 32, 64]; for decreasing filters (N_d) with kernel sizes: [2, 4, 8, 16, 32, 64]; for peak filters (N_p) with kernel sizes: [2, 4, 8, 16, 32, 64]; the number of filters per convolutional layer is $N = 32$, with $K = 40$ iterations; Dilation starts at $D_0 = 1$ in the initial layer and increases progressively with depth, reaching $D_1 = 2$ and $D_2 = 4$. This layered approach, combined with the use of dilation and multiplexing, ensures robust pattern detection while maintaining a lightweight and efficient architecture. The ensemble of 5 LITE networks is called LITETime.

3. Proposed Method

In our work, we propose adapting well-established architectures for time series classification, adopting a multimodal approach that processes each modality independently. Specifically, we adapt Fully Convolutional Network (FCN) [Wang et al. 2017], InceptionTime [Szegedy et al. 2017], and Light Inception with boosTing tEchniques (LITE) [Ismail-Fawaz et al. 2025] architectures better to suit the characteristics of multivariate time series data. By treating each modality as a distinct input, our approach aims to leverage the unique features of each modality while maintaining the flexibility and robustness of these proven models. This design choice allows us to capture individual modality patterns and cross-modal relationships, improving classification performance in complex time series tasks.

3.1. Multimodal Adaption

This section will detail how the architectures were adapted to the multimodal context in different phases of the neural network. In our approach, each modality is first processed independently through separate input layers, allowing the network to learn modality-specific features. After a series of layers, the outputs are concatenated and passed through

the rest of the network. The concatenation point determines whether the fusion occurs in the late or intermediate phase, as explained in the following sections.

3.1.1. Late Fusion

In late fusion, the modality-specific features are concatenated in the last layer before the output. This approach processes each modality independently throughout most of the network, combining them only at the end to make the final prediction, allowing for better handling of heterogeneous data. The late fusion of the data in all the architectures occurs before the last dense layer, as shown in Figure 4 (b), exemplified by the FCN with a dataset with three different modalities.

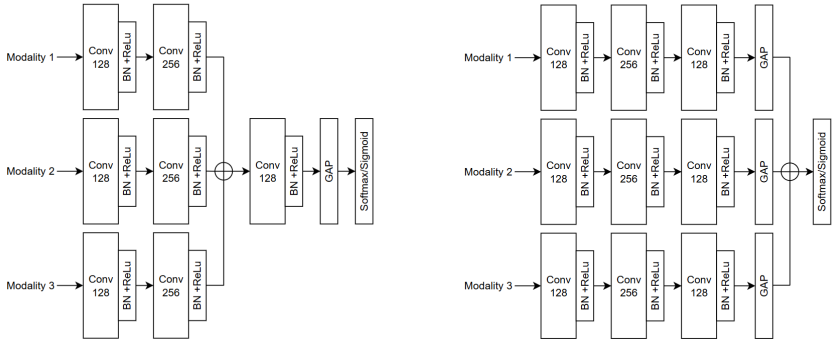


Figure 4. In Figure (a), we present a representation of an FCN adapted for intermediate multimodality, while in Figure (b), we show the same architecture adapted for late multimodality.

3.1.2. Intermediate Fusion

Intermediate fusion in the context of multimodal learning refers to combining data from different modalities at a later stage in the network, after each modality has been processed independently through its specialized layers. This approach allows each modality to be handled according to its unique characteristics, enabling more specialized feature extraction. The intermediate fusion of the data occurred at the following stages in each architecture: FCN: after the second convolutional block, as shown in Figure 4 (a); Inception: after 4 inception blocks; LITE: after the 6 parallel convolutions.

4. Experimental Setup

4.1. Datasets

In this work, we are using a subset of 7 datasets that make sense for the multimodal context from the UEA multivariate time series classification archive [Bagnall et al. 2018], an effort to build an archive for multivariate time series classification similar to UCR's univariate archive. This archive contains 30 multivariate time series datasets with a wide range of cases, dimensions, and series lengths, including Human Activity Recognition,

Motion Classification, ECG Classification, EEG/MEG Classification, among others. The selected datasets contain subsets of variables that come from different sensors. The used datasets are: ArticularyWordRecognition (AWR), AtrialFibrillation (AF), BasicMotions (BM), Cricket (CR), NATOPS (NA), RacketSports (RS), StandWalkJump (SWJ). Accuracy was used as the main metric of this experiment because all the datasets are balanced.

4.2. Experimental Setup and Training Parameters

Our results were obtained on a subset of 7 datasets of the UEA multivariate archive using an RTX 3090 GPU with 24GB of VRAM. In the experiments, we accounted for the accuracy of each model. The model used for testing is the best model obtained in the training step, considering validation accuracy. The Adam optimizer was used with a Reduce on Plateau with an initial learning rate of 0.001 and reduced by a factor of 0.75 after 250 epochs with no improvement. The model was trained for 5000 epochs with a batch size of 32. All the code was written in Python using Pytorch [1] package. All models were trained 10 times to mitigate significant variations due to random weight initialization, and averages for each metric were calculated. The code used can be found in the repository[2].

5. Results and Discussion

This section presents the results obtained by the previously described multivariate architectures and their adaptations to the multi-source scenario by leveraging fusion techniques. We executed each experiment, i.e., each combination between dataset and architecture, ten times to avoid the results being significantly affected by variations caused by random initializations.

5.1. General Accuracy Performance

The FCN architecture performed exceptionally well, reaching accuracy levels close to 100% in most datasets regardless of the data fusion stage, with fusion leading to equal or better performance in four out of five cases and, when worse, showing only minor drops of up to 0.7 percentage points. In the more challenging AtrialFibrillation and StandWalkJump datasets, data fusion notably improved results, with intermediate fusion yielding gains of nearly 12 percentage points over the single-modal version. The LITETime architecture performed slightly below FCN on most datasets but also showed substantial accuracy improvements with fusion in the two hardest datasets. InceptionTime behaved somewhat differently, showing minimal changes with fusion on simpler datasets but underperforming compared to FCN and LITETime in more cases. Nonetheless, in the AtrialFibrillation and StandWalkJump datasets, InceptionTime outperformed the others with or without fusion, although late fusion led to a drop in accuracy for AtrialFibrillation, contrasting with the otherwise consistent improvements from fusion techniques.

We also compare these results with the related literature, considering the results published in a wide experimental evaluation [Ruiz et al. 2021] that pointed out MUSE [Schäfer and Leser 2017] as the state-of-the-art for the two most challenging datasets. The InceptionTime Intermediate model had a very similar accuracy of MUSE,

[1] https://pytorch.org/

[2] https://github.com/samuelthomaz7/multimodal_time_series_classification

which obtained 74% accuracy on the AtrialFibrillation dataset, achieving 75% accuracy. For the StandWalkJump dataset, all multimodal approaches surpassed MUSE, which achieved an accuracy of 38% on the benchmark. The lowest-performing multimodal model, LITETime Late, achieved 46.7% accuracy, while the best result was obtained by InceptionTime Late, with an accuracy of 66.7%. In a subsequent study conducted in 2022 [Pham et al. 2023], using the StandWalkJump dataset, an accuracy of 53.3% was achieved, which is also below the 66.7% obtained by InceptionTime Late.

In conclusion, for the more challenging datasets, intermediate data fusion demonstrated an improvement in accuracy across all architectures, while late-stage fusion showed no improvement in just one case. These findings emphasize the significant potential of data fusion to enhance the performance of classification models. By strategically applying data fusion techniques, particularly at intermediate stages, we can achieve notable improvements in the accuracy of time series classification models, highlighting the value of this approach in addressing complex data scenarios. All the results are shown in Table 1, the best accuracy is highlighted for each base architecture and dataset.

Table 1. Average max test accuracy per model and dataset in 10 runs. Abbreviations: Int. = Intermediate, IncT = InceptionTime.

Dataset	FCN	FCN Int.	FCN Late	LITE	LITE Int.	LITE Late	IncT	IncT Int.	IncT Late
AWR	99.6	**100.0**	**100.0**	98.7	99.1	**99.4**	**98.6**	98.5	98.4
AF	43.3	**55.0**	53.3	30.0	**46.7**	**46.7**	70.0	**75.0**	65.0
BM	100.0	100.0	100.0	**100.0**	98.8	97.5	95.0	91.2	**97.5**
CR	100.0	100.0	100.0	96.7	98.3	**98.6**	**99.4**	98.9	98.3
NA	**99.6**	99.0	98.9	**98.6**	97.8	94.7	**98.3**	97.8	97.4
RS	93.6	**95.4**	**95.4**	**93.0**	89.0	83.8	**93.6**	93.1	93.0
SWJ	50.0	**58.3**	**58.3**	41.7	**50.0**	46.7	56.7	65.0	**66.7**

5.2. Stability to Random Initialization

To ensure robustness beyond accuracy alone, we evaluated the stability of each model by measuring the standard deviation of accuracy across 10 runs with different random seeds, assessing their sensitivity to stochastic factors such as initialization and data shuffling. This analysis, which complements overall performance evaluation, revealed that five of the seven datasets showed near zero variability due to consistently high accuracy across all models. For the more challenging AtrialFibrillation and StandWalkJump datasets, where fusion models achieved the most significant accuracy gains, we observed a general decrease in variability of test data accuracy across most models. However, an exception was found in the LITETime Intermediate model, which showed increased variability. In contrast, the FCN Late model maintained the same level of variability as its non-multimodal version, as shown in Table 2.

6. Conclusion

This study explored the classification of multi-source multivariate time series using a multimodal approach, applying it to benchmark datasets from the UEA repository selected for their relevance to modality separation. Three established architectures, namely FCN, InceptionTime, and LITETime, were used to assess whether handling each modality independently could enhance performance. Results showed that while both conventional and

Table 2. Variation in the standard deviation of accuracy using multimodal models on AtrialFibrillation and StandWalkJump datasets, calculated over 10 runs with different random initializations.

Model Name	Variation over non-Multimodal
FCN Late	0%
FCN Intermediate	-3.2%
InceptionTime Late	-21.6%
InceptionTime Intermediate	-15.4%
LITETime Late	-9.9%
LITETime Intermediate	+15.4%

multimodal approaches achieved high accuracy on most datasets, the multimodal strategy outperformed in more complex cases where single modal models struggled, offering gains in accuracy, especially with intermediate fusion. In general, models using data fusion also demonstrated greater stability across different random initializations, indicating more consistent performance. These improvements often surpassed previously reported results, reinforcing the potential of multimodal modeling to handle the challenges and variability of multivariate time series, particularly when modality separation is important.

References

Ay, E., Devanne, M., Weber, J., and Forestier, G. (2022). A study of knowledge distillation in fully convolutional network for time series classification. In *2022 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE.

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*.

Fawaz, H. I., Forestier, G., Weber, J., Idoumghar, L., and Muller, P.-A. (2019). Adversarial attacks on deep neural networks for time series classification. In *2019 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE.

Foumani, S. N. M., Tan, C. W., and Salehi, M. (2021). Disjoint-cnn for multivariate time series classification. In *2021 International Conference on Data Mining Workshops (ICDMW)*, pages 760–769. IEEE.

Ismail-Fawaz, A., Devanne, M., Berretti, S., Weber, J., and Forestier, G. (2025). Look into the lite in deep learning for time series classification. *International Journal of Data Science and Analytics*, pages 1–21.

Ismail-Fawaz, A., Devanne, M., Weber, J., and Forestier, G. (2022). Deep learning for time series classification using new hand-crafted convolution filters. In *2022 IEEE International Conference on Big Data (Big Data)*, pages 972–981. IEEE.

Ismail Fawaz, H., Forestier, G., Weber, J., Idoumghar, L., and Muller, P.-A. (2019). Deep neural network ensembles for time series classification. *arXiv e-prints*, pages arXiv–1903.

Ismail Fawaz, H., Lucas, B., Forestier, G., Pelletier, C., Schmidt, D. F., Weber, J., Webb, G. I., Idoumghar, L., Muller, P.-A., and Petitjean, F. (2020). Inceptiontime: Find-

ing alexnet for time series classification. *Data Mining and Knowledge Discovery*, 34(6):1936–1962.

Lahat, D., Adali, T., and Jutten, C. (2015). Multimodal data fusion: an overview of methods, challenges, and prospects. *Proceedings of the IEEE*, 103(9):1449–1477.

Medeiros Júnior, J. G. B., Mitri, A. G., and Silva, D. F. (2024). Semi-periodic activation for time series classification. In *Proceedings of the 34th Brazilian Conference on Intelligent Systems*.

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

Mohammadi Foumani, N., Miller, L., Tan, C. W., Webb, G. I., Forestier, G., and Salehi, M. (2024). Deep learning for time series classification and extrinsic regression: A current survey. *ACM Computing Surveys*, 56(9):1–45.

Pham, A.-D., Kuestenmacher, A., and Ploeger, P. G. (2023). Tsem: Temporally-weighted spatiotemporal explainable neural network for multivariate time series. In *Future of Information and Communication Conference*, pages 183–204. Springer.

Rakhshani, H., Fawaz, H. I., Idoumghar, L., Forestier, G., Lepagnot, J., Weber, J., Brévilliers, M., and Muller, P.-A. (2020). Neural architecture search for time series classification. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE.

Ruiz, A. P., Flynn, M., Large, J., Middlehurst, M., and Bagnall, A. (2021). The great multivariate time series classification bake off: a review and experimental evaluation of recent algorithmic advances. *Data Mining and Knowledge Discovery*, 35(2):401–449.

Schäfer, P. and Leser, U. (2017). Multivariate time series classification with weasel+ muse. *arXiv preprint arXiv:1711.11343*.

Shi, P., Ye, W., and Qin, Z. (2021). Self-supervised pre-training for time series classification. In *2021 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE.

Szegedy, C., Ioffe, S., Vanhoucke, V., and Alemi, A. (2017). Inception-v4, inception-resnet and the impact of residual connections on learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 31.

Tripathi, A. M. and Baruah, R. D. (2020). Multivariate time series classification with an attention-based multivariate convolutional neural network. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE.

Wang, Z., Yan, W., and Oates, T. (2017). Time series classification from scratch with deep neural networks: A strong baseline. In *2017 International joint conference on neural networks (IJCNN)*, pages 1578–1585. IEEE.

Yeh, C.-C. M., Zhu, Y., Ulanova, L., Begum, N., Ding, Y., Dau, H. A., Zimmerman, Z., Silva, D. F., Mueen, A., and Keogh, E. (2018). Time series joins, motifs, discords and shapelets: a unifying view that exploits the matrix profile. *Data Mining and Knowledge Discovery*, 32(1):83–123.