COCALITE: A Hybrid Model COmbining CAtch22 and LITE for Time Series Classification

Oumaima Badi
Université de Haute Alsace
IRIMAS
Mulhouse, France
oumaima.badi.contact@gmail.com

Maxime Devanne
Université de Haute Alsace
IRIMAS
Mulhouse, France
maxime.devanne@uha.fr

Ali Ismail-Fawaz
Université de Haute Alsace
IRIMAS
Mulhouse, France
ali-el-hadi.ismail-fawaz@uha.fr

Javidan Abdullayev
Université de Haute Alsace
IRIMAS
Mulhouse, France
javidan.abdullayev@uha.fr

Vincent Lemaire
Orange innovation
Lannion, France
vincent.lemaire@orange.com

Stefano Berretti
University of Florence
MICC
Florence, Italy
stefano.berretti@unifi.it

Jonathan Weber
Université de Haute Alsace
IRIMAS
Mulhouse, France
jonathan.weber@uha.fr

Germain Forestier
Université de Haute Alsace
IRIMAS
Mulhouse, France
Monash University, DSAI
Melbourne, Australia
germain.forestier@uha.fr

***Abstract*—Time series classification has achieved significant advancements through deep learning models; however, these models often suffer from high complexity and computational costs. To address these challenges while maintaining effectiveness, we introduce COCALITE, an innovative hybrid model that combines the efficient LITE model with an augmented version incorporating Catch22 features during training. COCALITE operates with only 4.7% of the parameters of the state-of-the-art Inception model, significantly reducing computational overhead. By integrating these complementary approaches, COCALITE leverages both effective feature engineering and deep learning techniques to enhance classification accuracy. Our extensive evaluation across 128 datasets from the UCR archive demonstrates that COCALITE achieves competitive performance, offering a compelling solution for resource-constrained environments.**

***Index Terms*—Time Series Classification, Deep Learning, Feature Engineering, LITE Model, Catch22, Hybrid Model**

I. Introduction

Time Series Classification (TSC) is a crucial research area with broad applications. It is used in healthcare [1], finance [2], and human activity recognition [3], as well as other key domains. The UCR Time Series Archive [4] has been pivotal in advancing research by providing a comprehensive repository of benchmark datasets.

Over time, various approaches have been developed for TSC, ranging from distance-based and feature-based methods to interval-based, dictionary-based, and shapelet-based techniques. More recent innovations include convolutional and deep learning models, as well as hybrid approaches that integrate these methods for improved performance.

Despite these advancements, TSC models often face the challenge of balancing high performance with computational cost. Hybrid approaches, such as HIVE-COTE v2.0 (HC2) [5], achieve state-of-the-art performance by independently training and combining multiple classifiers. Similarly, deep learning models like InceptionTime [6] excel due to their superior performance, leveraging deep neural networks. However, these models are resource-intensive, which limits their practical use in environments with constrained computational resources.

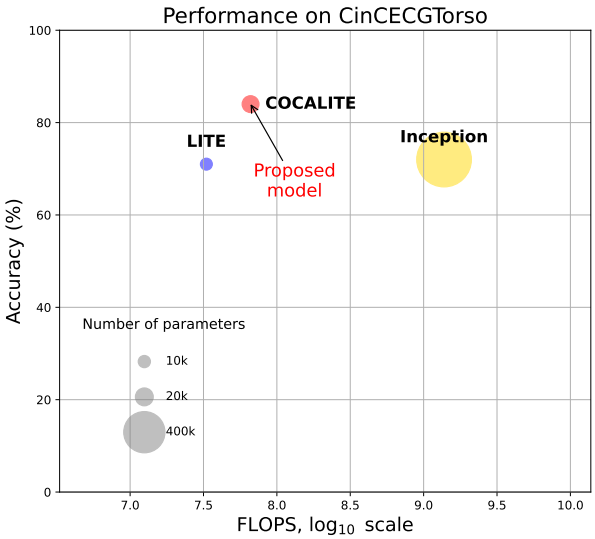

Fig. 1. Comparison of multiple models on the CinCECGTorso dataset, evaluating accuracy versus computational efficiency. The y-axis represents accuracy, while the x-axis denotes the $\log_{10}$ of Floating-point Operations Per Second (FLOPS). Circle sizes correspond to the number of trainable parameters. COCALITE, with nearly 20k parameters and a $\log_{10}$ FLOPS value of 7.8, achieves the highest accuracy on the test set in this comparison.

To address the limitations of current TSC models, we propose a hybrid approach that integrates deep learning models with feature-based methods. Our approach aims to combine the strengths of deep learning's representation capabilities with the efficiency and interpretability of feature-based methods.

As part of this approach, efficient deep learning models

DOI: https://doi.org/10.1109/BigData62323.2024.10825872‘

like LITE [7] represent a key advancement. LITE utilizes DepthWise Separable Convolutions and other techniques to significantly reduce the number of parameters, making it a viable solution for constrained environments, with parameters reduced to just 2.34% of those in InceptionTime.

In addition to LITE, automatic feature engineering has proven to be an effective approach for TSC [8]. TSFresh and hctsa are among the leading tools in this domain, with the Catch22 feature set offering a notable advancement. Introduced by [9], Catch22 comprises 22 computationally efficient features selected for their strong classification performance and minimal redundancy from the 7,700 features available in the hctsa toolbox [10]. These features provide a lightweight complement to the LITE model's 32 latent features. In contrast, using a larger feature set, such as the 777 features from TSFresh, could overwhelm the classifier, potentially skewing its focus towards external features and diminishing the effectiveness of the model's learned representations.

Combining these elements, we introduce COCALITE, a hybrid model that integrates the base LITE architecture with an augmented version incorporating Catch22 features during training. This approach seeks to balance high performance with computational efficiency by leveraging the complementary strengths of both models. Rather than merely integrating LITE with Catch22 features as initially suggested, we have extended our approach to employ an ensemble strategy to fully exploit their respective advantages. The rationale behind this strategy will be elaborated in Section IV, where we analyze how this strategy contributes to superior performance and efficiency.

Our experimental results, evaluated using datasets from the univariate UCR Time Series Archive [4], show that COCALITE not only improves performance but also has significantly lower complexity compared to the state-of-the-art Inception model [6]. This is exemplified by the CinCECG-Torso dataset, as illustrated in Figure 1.

Our main contributions in this work are:

- We propose COCALITE, a hybrid model that achieves only 4.7% of the parameters of the Inception model, while maintaining competitive performance.
- We conduct a detailed ablation study to explore various strategies within COCALITE, clarifying the model's motivation and demonstrating its effectiveness.
- We present a reduced version of COCALITE that maintains similar performance, while achieving approximately half the complexity.

The paper is organized as follows: Section II reviews related work, including recent advancements in feature engineering, deep learning and hybrid approaches for TSC. Section III provides a detailed explanation of our proposed model. Section IV analyzes and discusses the results. Finally, Section V summarizes the key findings and suggests future research directions.

II. Background and Related Work

A. Definition

A univariate time series of length L is an ordered set of real values over time, denoted as:

$$\mathbf{X} = (x_1, x_2, \ldots, x_L). \tag{1}$$

Given a dataset $\mathcal{D} = \{(\mathbf{X}_i, \mathbf{Y}_i)\}_{i=1}^{N}$, consisting of N time series $\mathbf{X}_i$ with corresponding one-hot encoded labels $\mathbf{Y}_i$, where $\mathbf{Y}_i$ is a vector of length K with each element $j \in [1, K]$ set to 1 if $\mathbf{X}_i$ belongs to class j and 0 otherwise. The goal is to learn a function $f : \mathbf{X} \rightarrow \mathbf{Y}$ that accurately classifies each input time series into one of the predefined categories.

B. Feature Engineering for Time Series Classification

Feature engineering is essential in TSC, as it extracts meaningful characteristics from raw time series data to improve classification. Several approaches have been developed, each offering different advantages and trade-offs in terms of accuracy and computational efficiency.

One comprehensive approach is the Highly Comparative Time-Series Analysis (HCTSA) toolbox [10], which provides over 7,700 features, offering a powerful but computationally intensive solution. To address this, the Canonical Time Series Characteristics (Catch22) [9] feature set was developed, containing 22 carefully selected features chosen for strong classification performance and minimal redundancy. The Catch22 set significantly reduces computation time by a factor of 1000 while achieving a mean classification accuracy of 71.7%, only 7.5% less than the full HCTSA set. Despite Catch22's effectiveness, its limited feature set may struggle with datasets where subtle class distinctions necessitate advanced extraction techniques to capture complex temporal dynamics beyond its capabilities. This suggests that integrating Catch22 with lightweight deep learning models could effectively address their respective limitations without significantly increasing computational cost.

Another popular approach, Time Series Feature Extraction based on Scalable Hypothesis Tests (TSFresh) [11], provides nearly 800 features spanning time, frequency, and wavelet domains, using hypothesis testing to retain the most relevant ones. However, TSFresh can be computationally demanding, particularly on large datasets.

C. Deep Learning for Time Series Classification

Deep learning has significantly advanced the field of TSC by enabling the automatic extraction of complex patterns from raw data. Unlike traditional machine learning approaches that relied on handcrafted features and required domain expertise and extensive preprocessing. As mentioned in the review [12], early deep learning models for TSC, such as Multi-Layer Perceptrons (MLP), demonstrated the potential of neural networks but were limited in their ability to capture temporal dependencies due to their fully connected nature. On the other hand, Fully Convolutional Networks (FCNs) [13] improved upon traditional architectures by using convolutional layers

followed by Batch Normalization and ReLU activation, avoiding local pooling to preserve the time series length and capture temporal relationships within the data. ResNet architectures, also introduced by the same authors [13], utilized shortcut connections to maintain information across layers and mitigate the vanishing gradient problem.

InceptionTime [6], inspired by the Inceptionv4 architecture [14], advances TSC by employing an ensemble of Inception models, each using convolutional kernels of varying sizes to capture diverse temporal patterns. This approach has achieved state-of-the-art results across multiple benchmark datasets. Building on this success, researchers developed Hybrid Inception (H-Inception) [15], which enhances the original Inception architecture by integrating handcrafted convolutional filters. An extension of this concept, Hybrid InceptionTime (H-InceptionTime), further improves classification performance by ensembling five H-Inception models, thus combining diverse feature representations to refine accuracy in TSC tasks.

Advancing this field, Light Inception with Boosting Techniques (LITE), proposed in [7], presents a parameter-efficient alternative, utilizing only 2.34% of InceptionTime's parameters while maintaining competitive performance. This efficiency arises from DepthWise Separable Convolution (DWSC) and techniques such as multiplexing [6], hand-crafted filters [15], and dilated convolutions [16]. Consequently, LITE is 2.78 times faster in training and consumes 2.79 times less power than InceptionTime, making it suitable for resource-constrained environments. While LITE's limited parameters contribute to its efficiency, they may also increase sensitivity to variance during training; however, this can be effectively addressed by employing an ensemble approach, known as LITETime.

D. Hybrid Approaches in Time Series Classification

Hybrid approaches in TSC combine multiple methodologies to enhance performance and robustness. The Hierarchical Vote Collective of Transformation-Based Ensembles (HIVE-COTE) [17] has evolved through several versions, starting with HIVE-COTEα (HCα) that combined classifiers like Elastic Ensemble (EE) and Time Series Forest (TSF). HIVE-COTE v1.0 (HC1) [18] simplified the model by reducing the number of classifiers, while the latest version, HIVE-COTE v2.0 (HC2) [5], introduced new classifiers, including the Transform-Based Ensemble (TDE) and DrCIF, as well as an ensemble of ROCKET classifiers [16] known as Arsenal. Despite its advancements, HIVE-COTE faces challenges related to computational complexity and training times.

Hydra-MultiRocket [19] is another notable hybrid model that combines randomly generated convolutional kernels organized into competitive groups with MultiRocket's [20] diverse pooling strategies. This integration captures various temporal patterns, enhancing feature extraction and classification performance. While Hydra-MultiRocket enables rapid training and inference, its implementation can be complex and resource-intensive for large datasets.

E. Complexity and Computational Cost

Our approach addresses the limitations of existing time series classification methods, which often struggle to balance performance and computational efficiency. We introduce a unified framework that combines deep learning with automatic feature extraction, as illustrated in Figure 2. Specifically, we integrate the lightweight LITE model with the efficient Catch22 feature set to develop a time series classification model suitable for resource-limited environments. While this work focuses on one instantiation, our adaptable framework encourages exploration of other promising combinations in future research.

III. Proposed Approach

A. Backbone Deep Learning Architecture

As a deep learning backbone, we employ the LITE model, originally proposed in [7]. It is designed to optimize the trade-off between performance and computational efficiency in TSC. The LITE architecture leverages multiples boosting techniques including:

- **Dilated Convolutions** [16]: By introducing gaps between the kernel elements, dilated convolutions increase the receptive field of the model without adding additional parameters, allowing it to capture long-range dependencies within the time series data.
- **Multiplexing** [6]: By applying multiple convolutional layers with different kernel sizes in parallel, multiplexing enables the model to learn features at multiple scales simultaneously, enhancing its ability to capture complex temporal patterns in the data.
- **Depthwise Separable Convolutions (DWSC)** [7]: By decomposing the standard convolution operation into depthwise and pointwise convolutions, DWSC reduces computational cost and the number of parameters, while still capturing complex patterns in the time series data.
- **Hand-crafted Filters** [15]: Using hand-crafted filters to identify specific patterns in the data without additional learning enables the model to focus on learning more complex patterns.

The LITE architecture includes an initial layer with standard convolutions using hand-crafted filters and multiplexing, followed by dilated DWSC in subsequent layers. A Global Average Pooling (GAP) layer aggregates information across the time dimension, reducing dimensionality before the final fully connected layer, which outputs class probabilities.

B. Feature Extraction Method

As a feature extraction method, we employ the Catch22 feature set [9] due to its notable advantages:

- **Computational Efficiency**: With near-linear complexity, $O(N^{1.16})$, Catch22 significantly reduces computational demands compared to more resource-intensive methods.
- **Robust Performance**: It achieves a mean classification accuracy of 71.7%, which is only marginally lower than that the full HCTSA feature set [9].

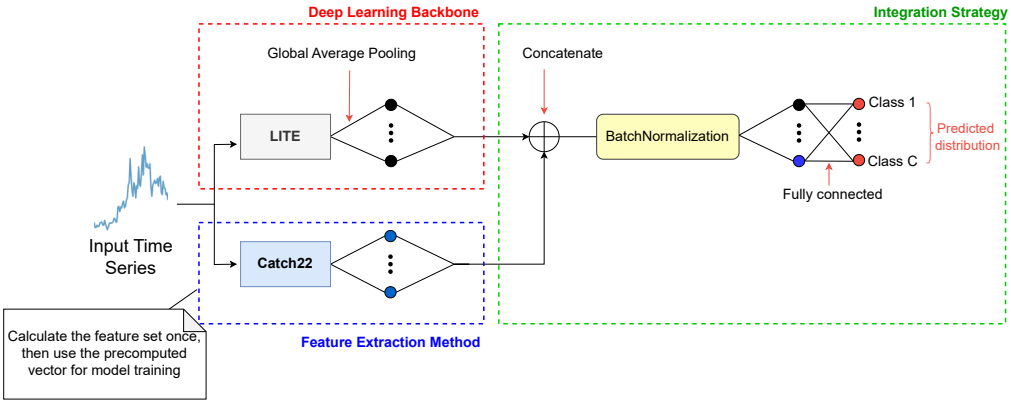

Fig. 2. Architecture of the LITE-Catch22 model (the second component of COCALITE), a specific implementation of our proposed Deep Learning-Feature Extraction integration strategy.

- **Lightweight and complementary**: The compact nature of the Catch22 feature set complements the latent features from the LITE model, ensuring a balanced and effective classification framework without over-reliance on either set.

C. Integration Strategy

To incorporate Catch22 features into LITE, we integrate these features into the model's latent space, enhancing its capacity to capture nuanced patterns (see Figure 2 for an overview). The process is outlined as follows:

1) Feature Extraction: Catch22 features $F_{\text{Catch22}}(\mathbf{X})$ are extracted once from the input series, precomputed for efficient reuse, avoiding repetitive calculations.

2) Latent Space Integration: At each training epoch, the latent features $F_{\text{LITE}}(\mathbf{X})$ of size $d_{\text{LITE}} = 32$ are concatenated with Catch22 features $F_{\text{Catch22}}(\mathbf{X})$ of size $d_{\text{Catch22}} = 22$, forming an integrated vector:

$$F_{\text{integrated}}(\mathbf{X}) = F_{\text{LITE}}(\mathbf{X}) \oplus F_{\text{Catch22}}(\mathbf{X}), \tag{2}$$

where $\oplus$ denotes concatenation, resulting in a 54-dimensional feature vector.

3) Batch Normalization: Although Catch22 features are precomputed and normalized before integration, combining them with the evolving latent features from the LITE model can lead to internal covariate shifts. To counter these shifts and maintain training stability, we apply batch normalization to $F_{\text{integrated}}(\mathbf{X})$, stabilizing the input distribution for the classifier.

The normalized features are then passed to a Dense Softmax classifier for final class prediction.

D. Proposed Hybrid Model: COCALITE

The integration strategy enhances LITE by allowing it to focus on complex patterns beyond those captured by Catch22 features. This suggests that a hybrid model could leverage distinct, complementary representations learned by both the original LITE and the Catch22-enhanced version. We propose COCALITE (COmbining CAtch22 and LITE), an ensemble model inspired by the ensemble learning approach of HC2 [5].

As shown in Figure 3, COCALITE comprises two models:

- **LITE model**: Trained solely on raw time series data, serving as the ensemble's baseline.
- **LITE-Catch22 model**: Utilizes the same LITE backbone with the same initialization but integrates pre-computed Catch22 features for a richer feature set.

The final prediction of COCALITE is obtained by averaging the class-wise predicted probabilities from both models:

$$P_{\text{COCALITE}}(c) = \frac{1}{2}\left(P_{\text{LITE}}(c) + P_{\text{LITE-Catch22}}(c)\right), \tag{3}$$

where $P_{\text{COCALITE}}(c)$ denotes the probability for class c in the COCALITE model, while $P_{\text{LITE}}(c)$ and $P_{\text{LITE-Catch22}}(c)$ represent the probabilities for class c from the LITE and LITE-Catch22 models, respectively.

To preserve each model's unique contributions, we avoid online training, which could dilute Catch22's impact and reduce learning efficiency. Separate training fully optimizes each model.

E. Ensemble of Hybrid Models: COCALITETime

In TSC ensemble learning, "Time" refers to models that average class-wise predicted probabilities across multiple instances (typically five) with different initializations, all using the same architecture and hyperparameters [6], [7], [15]. This approach enhances stability and performance by mitigating variability from random initialization. In contrast, without the "Time" designation, model performance is evaluated by averaging the accuracies of these instances, offering a simpler but less refined performance metric.

Building on the effectiveness of ensemble learning for TSC [21], we propose COCALITETime, an ensemble of COCALITE models designed to enhance performance and robustness while maintaining lower computational requirements than state-of-the-art models.

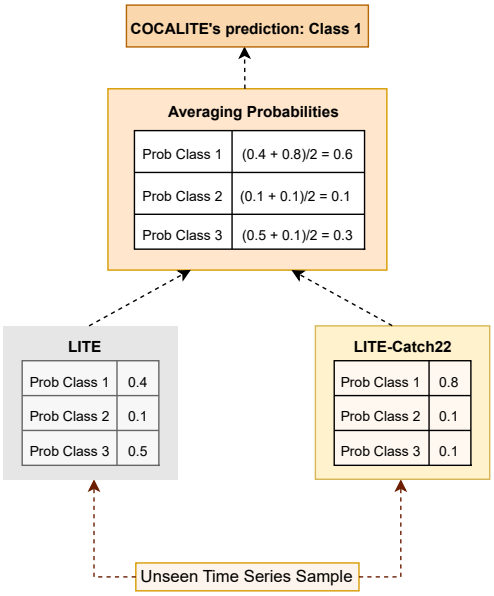

Fig. 3. Architecture of the proposed COCALITE hybrid model.

IV. Experimental Evaluation

A. Experimental Setup

1) Datasets: To evaluate our proposed model against state-of-the-art models, we use the UCR time series Archive [4], which contains 128 univariate time series datasets for classification, with the latest update in 2018.

For consistency with prior research, we applied the pre-processing methods outlined in [4]: each dataset was z-normalized, varying time series lengths were addressed through zero-padding, and missing values (NaNs) were imputed via linear interpolation.

2) Implementation details: We trained each model using the Adam optimizer [22], with a learning rate that decays based on the monitored training loss. Each model was trained with a batch size of 64 for 1500 epochs, and the version achieving the lowest training loss was selected for evaluation on the test set. To ensure consistency with established performance benchmarks, we utilized hyperparameters identified as optimal in the original LITE model's development, facilitating direct comparison with the base model's performance.

To account for the effects of random initialization, we repeated the training process five times with different random seeds. The results presented in this paper are averaged across these five runs, providing a robust assessment of model performance.

Feature extraction was performed using the *aeon* Python toolkit to compute Catch22 features [23]. All experiments were executed on an NVIDIA GeForce GTX 1080 GPU with 8 GB of memory, and the code will be made publicly available upon publication.

3) Comparison tool: The primary evaluation metric used is accuracy, which measures the proportion of correctly classified instances. To statistically validate our results, we employ the Multi-Comparison Matrix (MCM) evaluation tool [24], which is robust to the addition and removal of classifiers. Unlike traditional approaches that rely on average rank [25], the MCM uses *Mean-Accuracy*—the average accuracy of a classifier on the UCR datasets—as the ordering metric. Additionally, it includes a Mean-Difference value, which indicates the average difference in accuracy between two classifiers across all datasets. Positive Mean-Difference values suggest better performance of the classifier in the row, while negative values suggest the opposite.

The MCM also includes a Win/Tie/Loss metric, which captures the number of datasets, where the row classifier performs better, equally, or worse compared to the column classifier. To further assess the significance of these comparisons, the Wilcoxon signed-rank test [26] is applied, using a p-value threshold of 0.05. A p-value below this threshold indicates a statistically significant difference between classifiers, meaning the observed performance difference is unlikely due to chance. These significant p-values are highlighted in bold within the matrix.

B. Comparison with State-of-the-Art

1) Considered methods: We compare our COCALITETime model with leading approaches from the literature [27], including the top hybrid model HiveCote2 (HC2), the convolution-based method MultiRocket, and the best deep learning models, InceptionTime and H-InceptionTime. Additionally, we include the feature-based method combining Catch22 features with the Rotation Forest classifier (RotF) [28], which has proven superior for real-valued features [29].

This comparative study involves 127 datasets, excluding the Fungi dataset due to HC2's requirement for cross-validation to tune its parameters. Since the Fungi dataset contains only one training sample per class label, it is unsuitable for this comparison.

2) Comparative results: Comparative results are presented in the MCM shown in Figure 4. The proposed COCALITE-Time method ranks third in average accuracy across the 127 datasets, outperforming Catch22-RotF with a statistically significant difference indicated by a low p-value. When compared to more complex deep learning methods like InceptionTime and H-InceptionTime, COCALITETime demonstrates comparable performance, with high p-values suggesting no statistically significant differences. Notably, COCALITETime uses approximately 20k parameters per model, while InceptionTime uses 420k parameters, achieving similar performance with only 4.7% of the parameters. Figure 4 also indicates that our method has lower average performance than HC2 and MultiRocket, with statistically significant results. However, considering the trade-off between model complexity and computational resources, COCALITETime remains a competitive option with significantly fewer parameters and reduced computational demands.

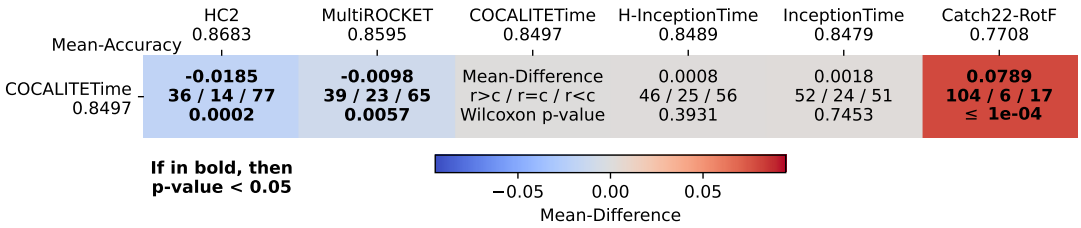

Fig. 4. The Multi-Comparison Matrix applied to show the performance of COCALITETime compared to state-of-the-art approaches.

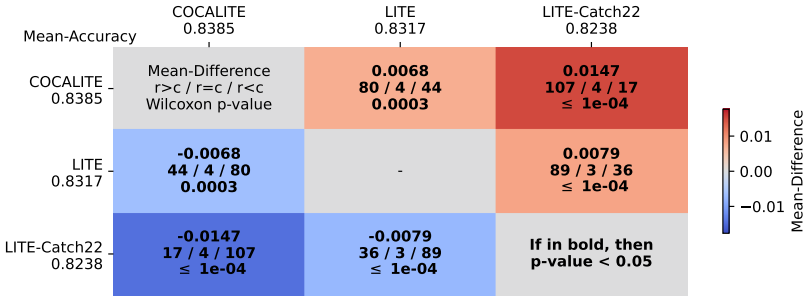

Fig. 5. The Multi-Comparison Matrix illustrates the one-vs-one performance comparisons between COCALITE and its components.

C. Ablation Study

1) Comparison Setup: To further assess the impact of our hybrid model on the original LITE model, we compare our COCALITE model with:

- The baseline LITE model.
- The LITE model enhanced with Catch22 features, referred to as LITE-Catch22.

2) Comparative Results: The results in Figure 5 indicate that the LITE model enhanced by our integration strategy (LITE-Catch22) does not outperform the original LITE model, suggesting that while the integration may help learn meaningful features for some datasets, it degrades classification performance on 89 datasets.

However, Figure 5 also highlights that COCALITE significantly outperforms the original LITE model, supporting our hypothesis that the features learned by both models are complementary. This enhances generalization and robustness, especially in noisy datasets. For a fair comparison, each LITE-Catch22 model shares initialization with its LITE counterpart to isolate the impact of Catch22 features.

To further illustrate our hypothesis, we compared the filters learned by the two models. The t-SNE scatter plot in Figure 6 visualizes the last convolutional layer's filters, showing distinct clustering of LITE (blue) and LITE-Catch22 (red) points. This suggests that incorporating Catch22 results in different filter representations, capturing a broad range of features.

D. Complexity Reduction in COCALITETime

Building on findings from [30], which showed that an ensemble of 10 LITE instances (LITETime-10) outperforms an ensemble of 5 instances (LITETime), we conducted a fair comparison by evaluating an ensemble of 5 COCALITE instances

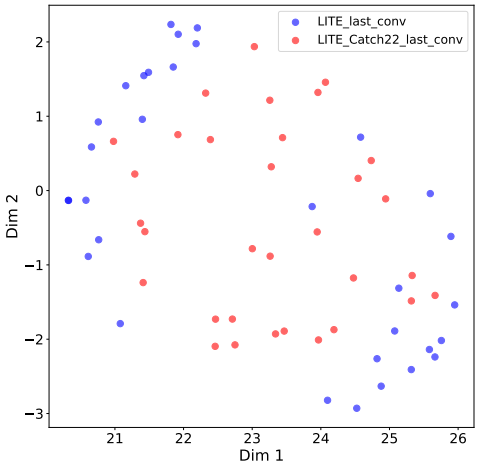

Fig. 6. t-SNE scatter plot of 2D filter representations from the last convolutional layer of the COCALITE components: LITE (blue) and LITE-Catch22 (red) models, trained on the FaceAll dataset.

(COCALITETime) against LITETime-10. Figure 7 indicates a low p-value, confirming the statistical significance of the performance differences and suggesting that COCALITETime's hybrid design and ensemble learning contribute to its superior performance in most scenarios.

To explore reducing COCALITETime's complexity, we halved the number of filters in each convolutional layer, creating COCALITETime-16-filters. This adjustment reduces the number of trainable parameters by approximately 50% (see Table I). As shown in Figure 7, COCALITETime-16-filters

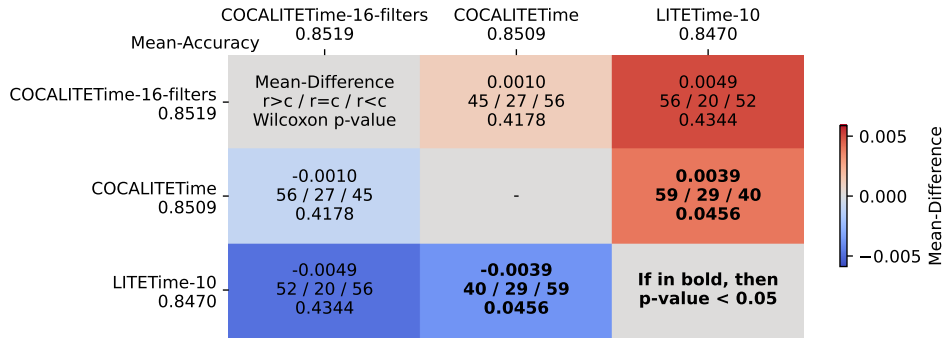

Fig. 7. The Multi-Comparison Matrix is used to show the performance of COCALITETime, COCALITETime-16-filters, and LITETime-10 in one-vs-one comparisons.

TABLE I
CONVOLUTIONAL LAYER FILTER COUNT AND TRAINABLE PARAMETERS BREAKDOWN BY MODEL AND COMPONENT.

Model	**Components**	**Nbr. Filters per Conv. Layer**	**Nbr. of Parameters per Component**	**Nbr. of Parameters per Model**
COCALITE	LITE	32	9 814	19 736
	LITE-Catch22	32	9 922	
COCALITE-16-Filters	LITE	16	4 070	8 216
	LITE-Catch22	16	4 146	

demonstrates no statistically significant performance difference compared to LITETime-10, indicating that the reduced-complexity model maintains comparable performance while substantially lowering the parameter count.

V. CONCLUSION

In this paper, we tackled the Time Series Classification challenge by blending statistical features and ensemble learning with deep learning models, specifically LITE. We introduced LITE-Catch22, which combines LITE's powerful feature extraction with Catch22's rich statistical insights, and COCALITE, a hybrid model that brings together the best of both models to boost performance.

Our experiments reveal that COCALITE often surpasses the standard LITE model by capturing the complementary features of each model. We also propose a reduced version, which delivers comparable performance to LITETime-10 but with about half the parameters, making it more efficient.

We aimed to find the right balance between performance and computational cost, and the results suggest we have made progress towards that objective. Future research could explore applying this approach to other deep learning models and experimenting with additional feature sets beyond Catch22.

In summary, our models provide a promising and efficient approach for time series classification, and we hope this work encourages further advancements in optimizing deep learning architectures.

ACKNOWLEDGMENT

This work was supported by the ANR DELEGATION project (grant ANR-21-CE23-0014) of the French Agence Nationale de la Recherche. The authors would like to acknowledge the High Performance Computing Center of the University of Strasbourg for supporting this work by providing scientific support and access to computing resources. Part of the computing resources were funded by the Equipex Equip@Meso project (Programme Investissements d'Avenir) and the CPER Alsacalcul/Big Data. The authors would also like to thank the creators and providers of the UCR Archive.

REFERENCES

[1] H. I. Fawaz, G. Forestier, J. Weber, L. Idoumghar, and P.-A. Muller, "Evaluating surgical skills from kinematic data using convolutional neural networks," *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pp. 214–221, 2018.

[2] L. Anghinoni, L. Zhao, Q. Zheng, and J. Zhang, "Time series trend detection and forecasting using complex network topology analysis," *International Joint Conference on Neural Networks*, pp. 1–7, 2018.

[3] S. Ji, W. Xu, M. Yang, and K. Yu, "3d convolutional neural networks for human action recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 1, pp. 221–231, 2012.

[4] H. A. Dau, A. Bagnall, K. Kamgar, C.-C. M. Yeh, Y. Zhu, S. Gharghabi, C. A. Ratanamahatana, and E. Keogh, "The ucr time series archive," in *IEEE/CAA Journal of Automatica Sinica*, vol. 6, no. 6, 2019, pp. 1293–1305.

[5] M. Middlehurst, J. Large, M. Flynn, J. Lines, and A. Bagnall, "Hc2: A new family of classifiers for time series classification," *ACM Transactions on Knowledge Discovery from Data*, vol. 14, no. 5, pp. 1–29, 2022.

[6] H. I. Fawaz, B. Lucas, G. Forestier, C. Pelletier, D. F. Schmidt, J. Weber, G. I. Webb, L. Idoumghar, P.-A. Muller, and F. Petitjean, "Inceptiontime: Finding alexnet for time series classification," *Data Mining and Knowledge Discovery*, vol. 34, no. 6, pp. 1936–1962, 2020.

[7] A. Ismail-Fawaz, M. Devanne, S. Berretti, J. Weber, and G. Forestier, "Lite: Light inception with boosting techniques for time series classification," in *Proceedings of the 2023 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, 2023.

[8] A. Renault, A. Bondu, V. Lemaire, and D. Gay, "Automatic feature engineering for time series classification: Evaluation and discussion," in *Proceedings of the 2023 International Joint Conference on Neural Networks (IJCNN)*, Gold Coast, Australia, June 2023, pp. 1–10.

[9] C. H. Lubba, S. S. Sethi, P. Knaute *et al.*, "catch22: canonical time-series characteristics," *Data Mining and Knowledge Discovery*, vol. 33, no. 6, pp. 1821–1852, 2019.

[10] B. Fulcher and N. Jones, "hctsa: A computational framework for automated time-series phenotyping using massive feature extraction," *Cell Systems*, vol. 5, no. 5, pp. 527–531, 2017.

[11] M. Christ, N. Braun, J. Neuffer, and A. W. Kempa-Liehr, "Time series feature extraction on basis of scalable hypothesis tests (tsfresh–a python package)," *Neurocomputing*, vol. 307, pp. 72–77, 2018.

[12] H. I. Fawaz, G. Forestier, J. Weber, L. Idoumghar, and P.-A. Muller, "Deep learning for time series classification: a review," *Data Mining and Knowledge Discovery*, vol. 33, no. 4, pp. 917–963, 2019.

[13] Z. Wang, W. Yan, and T. Oates, "Time series classification from scratch with deep neural networks: A strong baseline," in *Proceedings of the 2017 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2017, pp. 1578–1585.

[14] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. Alemi, "Inception-v4, inception-resnet and the impact of residual connections on learning," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 31, 2017.

[15] A. Ismail-Fawaz, M. Devanne, J. Weber, and G. Forestier, "Deep learning for time series classification using new hand-crafted convolution filters," in *Proceedings of IEEE Big Data 2022*, 2022, pp. 1–8.

[16] A. Dempster, F. Petitjean, and G. I. Webb, "Rocket: Exceptionally fast and accurate time series classification using random convolutional kernels," *Data Mining and Knowledge Discovery*, vol. 34, no. 6, pp. 1454–1495, 2020.

[17] J. Lines, S. Taylor, and A. Bagnall, "Time series classification with hive-cote: The hierarchical vote collective of transformation-based ensembles," *ACM Transactions on Knowledge Discovery from Data*, vol. 12, no. 5, pp. 1–36, 2018.

[18] A. Bagnall, M. Flynn, J. Large, and et al., "On the usage and performance of hive-cote v1.0," in *Proceedings of the 5th Workshop on Advanced Analytics and Learning on Temporal Data*, 2020.

[19] A. Dempster, D. F. Schmidt, and G. I. Webb, "Hydra: Competing convolutional kernels for fast and accurate time series classification," *Data Mining and Knowledge Discovery*, vol. 37, no. 5, pp. 1779–1805, 2023.

[20] C. W. Tan, A. Dempster, C. Bergmeir *et al.*, "Multirocket: multiple pooling operators and transformations for fast and effective time series classification," *Data Mining and Knowledge Discovery*, vol. 36, no. 6, pp. 1623–1646, 2022.

[21] H. I. Fawaz, G. Forestier, J. Weber, L. Idoumghar, and P.-A. Muller, "Deep neural network ensembles for time series classification," in *Proceedings of the 2019 International Joint Conference on Neural Networks (IJCNN)*. Budapest, Hungary: IEEE, 2019, pp. 1–6.

[22] D. P. Kingma and J. B. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[23] M. Middlehurst, A. Ismail-Fawaz, A. Guillaume, C. Holder, D. G. Rubio, G. Bulatova, L. Tsaprounis, L. Mentel, M. Walter, P. Sch"afer *et al.*, "aeon: a python toolkit for learning from time series," *arXiv preprint arXiv:2406.14231*, 2024.

[24] A. Ismail-Fawaz, A. Dempster, C. W. Tan, M. Herrmann, L. Miller, D. F. Schmidt, S. Berretti, J. Weber, M. Devanne, and G. F. et al., "An approach to multiple comparison benchmark evaluations that is stable under manipulation of the comparate set," *arXiv preprint arXiv:2305.11921*, 2023.

[25] A. Benavoli, G. Corani, and F. Mangili, "Should we really use post-hoc tests based on mean-ranks?" *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 152–161, 2016.

[26] F. Wilcoxon, "Individual comparisons by ranking methods," in *Breakthroughs in Statistics*. Springer, 1992, pp. 196–202.

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

[28] J. J. Rodríguez, L. I. Kuncheva, and C. J. Alonso, "Rotation forest: A new classifier ensemble method," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, no. 10, pp. 1619–1630, 2006.

[29] A. Bagnall, M. Flynn, J. Large, J. Line, A. Bostrom, and G. Cawley, "Is rotation forest the best classifier for problems with continuous features?" *arXiv preprint arXiv:1809.06705v3 [cs.LG]*, 2020. [Online]. Available: https://arxiv.org/abs/1809.06705v3

[30] A. Ismail-Fawaz, M. Devanne, S. Berretti, J. Weber, and G. Forestier, "Look into the lite in deep learning for time series classification," *arXiv preprint arxiv:2409.02869*, 2024.