A Strongly Typed Genetic Programming Approach to Feature Learning in Time Series Classification

Xuanhao Yang, Bing Xue, Mengjie Zhang
Centre for Data Science and Artificial Intelligence & School of Engineering and Computer Science
Victoria University of Wellington
PO Box 600, Wellington 6140, New Zealand
xuanhao.yang@vuw.ac.nz, {bing.xue, mengjie.zhang}@ecs.vuw.ac.nz

***Abstract*—Learning discriminative yet interpretable features remains a challenge in time series classification (TSC). Unlike traditional methods that rely on fixed hand-crafted features or deep neural network methods that lack interpretability, this paper proposes a strongly typed genetic programming approach to learning discriminative and interpretable features in TSC. Specifically, a new program structure is introduced, comprising three layers: domain transformation, feature extraction, and feature concatenation. The layered design enforces an explicit order of operations that mimics human expert logic, while still permitting flexible compositions. This enables the evolutionary search to learn discriminative representations tailored to the specific data. Experiments on 31 UCR datasets demonstrate the effectiveness of the learned features. In addition, visualisation of the evolved programs highlights the interpretability of the resulting solutions.**

***Index Terms*—genetic programming, feature learning, classification**

I. Introduction

Time series data, which record dynamic variations over continuous time intervals, have become increasingly prevalent with the rapid proliferation of digital services, and Internet of Things (IoT) devices. Downstream time series analysis tasks include classification [1], clustering [2], forecasting [3], anomaly detection [4]. Among these, time series classification (TSC) has been widely used in real-world applications such as smart healthcare [5] and industrial monitoring [6]. However, the complexity of real-world time series data, together with the increasing demand for interpretability, poses substantial challenges. As a result, considerable effort has been devoted to developing effective TSC methods. Representative categories include distance-based approaches [7], which measure pairwise similarity between time series using elastic distance functions; shapelet-based approaches [8], which identify class-discriminative subsequences and classify a series based on how well it matches these local patterns; and feature-based approaches [9], which learn a set of discriminative features guided by an explicit objective function for classification. Given the inherent potential for interpretability in feature-based methods, this paper focuses on feature-based TSC.

An important branch of feature-based TSC relies on domain expertise to design and extract hand-crafted features. These approaches typically extract a comprehensive set of predefined properties, such as statistical moments (e.g., mean, variance), shape descriptors (e.g., slopes), or autocorrelation features, to summarize the raw series [9]. A key advantage of such methods lies in their high interpretability. This is particularly important in decision-critical domains such as healthcare and finance, where practitioners need to understand why a prediction is made and assess whether it is trustworthy [10]. However, applying the exact same set of predefined features across different datasets restricts the flexibility of the learned representations. Recent advancements in feature-based methods have shifted from traditional hand-crafted feature engineering to neural-network-based representation learning [11]. A common paradigm is to pre-train a neural network encoder with a self-supervised objective to obtain class-discriminative representations, and then adapt it to the downstream classification task by attaching a classifier head and optimizing the model with supervised learning [12]. Owing to their large parameter space, these approaches are often more effective for modelling complex data, as they can learn flexible representations that vary across datasets. However, these approaches face significant limitations. The inherent black-box nature of deep neural networks means that the learned features lack interpretability, offering little insight into the underlying decision-amking process [13].

Strongly typed genetic programming (STGP), originally proposed by Montana [14], is an enhanced variation of standard genetic programming GP that introduces type constraints into the evolutionary process. STGP provides a promising alternative to learn flexible features for classification by automatically evolving tree-structured programs. In stark contrast to the black-box nature of deep neural networks, STGP offers potentially interpretable solutions through its tree-based representation, allowing practitioners to explicitly inspect and understand the learned features [15]. Recent STGP-based feature learning studies demonstrate the effectiveness of such structured representations across diverse domains, including image classification [16], multimodal learning [17], and industrial fault diagnosis [18]. The achievements in these domains demonstrate that STGP is capable of learning features that are not only discriminative but also interpretable. However, to the best of our knowledge, STGP-based feature learning methods specifically designed for general time series classification remain very limited. Accordingly, this work investigates the potential of STGP to learn effective and interpretable features

for TSC. The objectives of this study are summarised as follows:

- Develop a STGP-based program structure for TSC, enabling evolutionary search to determine how to transform input time series data into discriminative features.
- Investigate the performance of the evolved features by coupling them with a classifier and comparing with other methods.
- Visualise the evolved feature learning program to demonstrate the interpretability of the solution and gain insights into the decision-making process.

The rest of this article is organized as follows. Section II reviews related work on feature-based TSC methods. Section III presents the proposed STGP method and introduces the proposed program structure. Section IV describes the experimental design, and Section V reports the experimental results. Finally, Section VI concludes the paper and outlines future directions.

Algorithm 1: Overall Framework of the Proposed GP Method

Input: Training dataset $\mathcal{D}_{train}$, Population size m, Maximum generations G, Tournament size τ, Elitism rate p_e, Crossover rate p_c, Mutation rate p_m.
Output: The best feature learning program Ind_{best}.

1 $P_0 \leftarrow$ Initialize population using ramped half-and-half strategy under structural constraints;
2 Evaluate fitness of all individuals in P_0;
3 $Ind_{best} \leftarrow$ Best individual in P_0;
4 **for** $g = 1$ **to** G **do**
5 $\quad n_e \leftarrow \lfloor p_e \cdot m \rfloor$;
6 $\quad P_{elites} \leftarrow$ Select n_e individuals from P_{g-1} via elitism;
7 $\quad P_{parents} \leftarrow$ Select parents from P_{g-1} using tournament selection (size τ);
8 $\quad P_{offspring} \leftarrow$ Generate offspring by performing crossover and mutation operators on $P_{parents}$ according to the predefined probabilities p_c and p_m;
9 $\quad P_g \leftarrow P_{elites} \cup P_{offspring}$;
10 $\quad$ Evaluate fitness of all individuals in P_g;
11 $\quad$ Update Ind_{best} if a better individual is found;
12 **return** Ind_{best};

II. Related Work

Feature-based TSC methods aim to map an input time series to a discriminative feature vector, which can improve the performance of downstream classifier.

An important line of research relies on domain knowledge to design and extract hand-crafted features. A representative example is Catch22 [9], which distils a compact set of 22 features from thousands of candidate features through a data-driven selection process. These features capture diverse dynamical properties, such as autocorrelation structure and distributional statistics, while remaining minimally redundant and maintaining strong classification performance. Another widely used approach is TSFEL [19], which computes more than 60 features spanning temporal, statistical, and spectral domains. Despite their interpretability and ease of use, hand-crafted feature extraction methods are inherently limited by the fixed nature of predefined feature sets. Applying the same transformations across datasets can restrict representational flexibility and may fail to capture complex, dataset-specific information. This motivates adaptive approaches that construct task-specific features rather than relying on a static, manually designed feature set.

More recently, feature-based TSC has increasingly shifted toward deep neural networks. Han et al. [12] propose a masked time series modeling method called CBD, which augments the standard temporal reconstruction decoder with an auxiliary frequency decoder to explicitly address feature homogenization and spectral energy imbalance during representation learning. Roschmann et al. [20] propose an approach called TiViT, which transforms time series data into 2D image-like representations and repurposes large pretrained vision transformers as frozen feature extractors. The hidden-layer representations are then used to train a lightweight classifier for TSC. Although these deep feature learners are highly flexible and often achieve strong accuracy, their learned representations are typically difficult to interpret, motivating alternative approaches that can learn adaptive yet interpreteble features.

III. Proposed Approach

A. Algorithm Overview

The overall framework of the STGP-based method is summarised in Algorithm 1. Lines 1 corresponds to the population initialization, where m strongly typed GP trees are generated using the ramped half-and-half strategy [21]. Each individual is constrained by the designed program structure (see Section III-B). Lines 2 and 10 correspond to the fitness evaluation stage, which provides the feedback signal for guiding the evolutionary search (details are given in Section III-D). In line 6, elitism carries the best n_e individuals into the next generation, which helps prevent the search from moving toward worse solutions. In Line 7, the parent selection step chooses a set of high-quality individuals from the current population to serve as parents to generate offspring for the next generation. We adopt tournament selection, a simple and widely used operator in GP. An important benefit of tournament selection is that it relies on local competition rather than a full population ranking. As a result, individuals that are not the best overall can still be selected if they perform well within a tournament, helping preserve diversity of population. Using the selected parents, crossover and mutation are applied to create the remaining individuals of the population for the next generation according to predefined probabilities (line 8). All operations strictly follow the constraints used during population initialization.

B. New Program Structure

Fig. 1 illustrates the proposed strongly typed GP program structure for feature learning in TSC. Each individual is a type-safe program tree that maps an input time series to a feature vector. Excluding the input and output layers, the structure comprises three functional layers: domain transform, feature extraction, and feature concatenation. This imposes a fixed

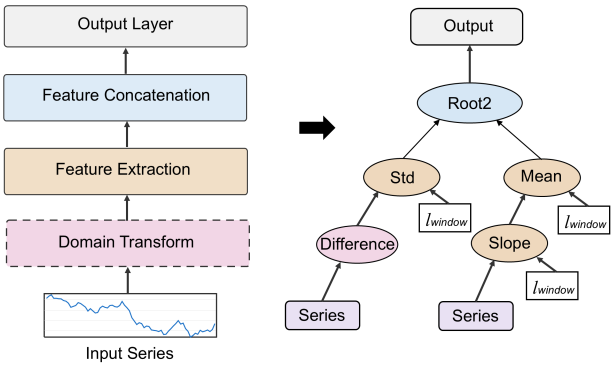

Fig. 1: Illustration of the proposed program structure and an example evolved program. An input time series is mapped to an alternative representation through the domain transform layer. The feature extraction layer subsequently derives a feature vector from the input series. Finally, the concatenation layer combines multiple feature vectors into a single output feature vector.

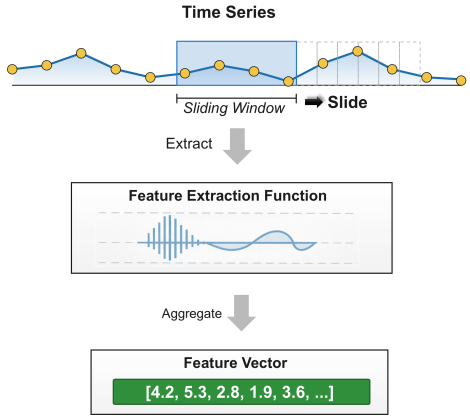

Fig. 2: Illustration of the sliding feature extraction process.

order on the feature learning operations to ensure the logical validity of evolved programs.

The proposed structure begins with the domain transform layer. Given a raw time series, this layer allows for a transformation into an alternative domain prior to feature extraction. While time domain representations are intuitive, salient class-discriminative patterns often manifest more clearly in alternative domains, such as the frequency domain [22] or the differencing domain [23]. Since discriminative information may be better captured in the time domain for certain tasks, this layer is designed to be optional.

The subsequent feature extraction layer serves as the core engine for pattern discovery. As shown in Fig. 2, the extraction operators slide a window along the series and aggregate the resulting measurements to capture information. Importantly, this layer is designed to be applied iteratively, thereby enabling evolved programs to capture richer information. For instance, in the right branch of the example program in Fig. 1, the model first extracts a slope-based shape feature vector from the input series, and then re-applies the extraction layer to compute a sliding mean over this vector, yielding a higher-level summary of local slope behaviour within each window. Details of the optional functions in this layer are provided in Section III-C.

The final feature vector is combined by the feature concatenation layer, which concatenates outputs from multiple branches to capture complementary views of the input (e.g., different domains, or different extraction methods). This layer provides a structured mechanism for building richer representations than any single branch can offer, while keeping the construction process explicit and traceable.

C. Function Set & Terminal Set

All functions used in the proposed method, together with their input types, output types, and brief descriptions, are summarised in Table 1.

The domain transform layer provides optional operators that convert the processed series into alternative representations. The **Difference** function applies first-order differencing, emphasising local changes and dynamic transitions. The **FFT** function transforms the time domain series into the frequency domain via the fast fourier transform and outputs the magnitude spectrum to capture periodic behaviour.

The feature extraction functions derive a feature vector from the input series or vector using a sliding-window mechanism governed by the terminal ℓ_{window}. The function **Mean**, **Median**, **Max**, and **Min** describe basic statistical properties of the series. The operators **Std** (standard deviation) and **IQR** (interquartile range) quantify dispersion. In addition, **Slope** provides information about local trends and shape patterns.

The feature concatenation layer combines outputs from multiple feature learning branches. Specifically, **Root2**, **Root3**, and **Root4** concatenate $2/3/4$ feature vectors, respectively, while **RootH** concatenates a feature vector with the output feature vector of **Root2**, **Root3** or **Root4**.

The terminal ℓ_{window} is used to control the sliding-window length of feature extraction function, allowing evolution to automatically select a suitable scale for the processed series or vector. Its search range is set from 2 to half of the length of the processed series or vector.

D. Fitness Evaluation

To improve generalisation of the learned features, we evaluate each GP individual using k-fold cross-validation. For a given candidate program, we first transform every training series into a feature vector. The resulting feature set is then divided into k folds. Each fold is used as the test set while the remaining $k-1$ folds are used for training an extremely randomized trees classifier [24]. This process is repeated k times so that each fold has exact one chance to be used for the test set. The fitness of an individual is defined as the average classification accuracy on the test set over the k rounds. Following common practice in GP-based feature learning, we set $k = 5$ [16].

TABLE I: Function Set of the Proposed Method

Layer	Function	Input	Output	Description
Domain Transform	Difference	Series	Series	Computes the first-order difference of the series.
	FFT	Series	Series	Computes the magnitude spectrum using FFT.
Feature Extraction	Mean	Series/Vector, ℓ_{window}	Vector	Average value via sliding window.
	Std	Series/Vector, ℓ_{window}	Vector	Standard deviation via sliding window.
	Min	Series/Vector, ℓ_{window}	Vector	Minimum value via sliding window.
	Max	Series/Vector, ℓ_{window}	Vector	Maximum value via sliding window.
	Median	Series/Vector, ℓ_{window}	Vector	Median value via sliding window.
	IQR	Series/Vector, ℓ_{window}	Vector	Interquartile range via sliding window.
	Slope	Series/Vector, ℓ_{window}	Vector	Linear slope via sliding window.
Feature Concatenation	RootH	Root2/3/4, Vector	Vector	Concatenates a new feature vector and the output feature vector of Root 2, Root 3 or Root 4.
	Root2	Vector, Vector	Vector	Concatenates two feature vectors.
	Root3	Vector, Vector, Vector	Vector	Concatenates three feature vectors.
	Root4	Vector, Vector, Vector, Vector	Vector	Concatenates four feature vectors.

IV. Experiment Design

A. Datasets

In the experiments, 31 time series datasets from the UCR Archive [25] is used, with detailed information reported in Table II. The number of classes ranges from 2 to 15, the total number of instances ranges from 40 to 4307, and the time series length ranges from 60 to 2000. These datasets cover diverse application scenarios (e.g., sensor measurements, motion/gesture recognition, and biomedical signals), providing a broad testbed for assessing the effectiveness of the learned features. All the datasets have been split into training and test set. The training set is used to guide feature learning and to train the classifier, while the test set is used to evaluate the final performance of the feature learning program together with the classifier.

TABLE II: Summary of the Datasets

ID	Dataset	Classes	Samples	Length
D1	ACSF1	10	200	1460
D2	Beef	5	60	470
D3	BirdChicken	2	40	512
D4	CBF	3	930	128
D5	Chlorine.	3	4307	166
D6	CinCECGTorso	4	1420	1639
D7	Coffee	2	56	286
D8	ECG200	2	200	96
D9	FacesUCR	14	2250	131
D10	GunPointAge.	2	451	150
D11	GunPointMale.	2	451	150
D12	GunPointOld.	2	451	150
D13	HouseTwenty	2	159	2000
D14	InsectEPGR.	3	311	601
D15	InsectEPGS.	3	266	601
D16	Lightning2	2	121	637
D17	Mallat	8	2400	1024
D18	MiddlePha.	3	554	80
D19	OliveOil	4	60	570
D20	ProximalCo.	2	891	80
D21	Refri.	3	750	720
D22	ScreenType	3	750	720
D23	SemgHandG.	2	900	1500
D24	SemgHandS.	5	900	1500
D25	SmallKit.	3	750	720
D26	SwedishLeaf	15	1125	128
D27	Symbols	6	1020	398
D28	SyntheticControl	6	600	60
D29	Trace	4	200	275
D30	UMD	3	180	150
D31	Yoga	2	3300	426

B. Comparison Methods

To demonstrate the effectiveness of the proposed GP-based feature learning method, comparisons are conducted against three categories of baselines.

The first category consists of traditional classifiers applied directly to the raw time series data. Four widely used models are considered: support vector machine (SVM), logistic regression (LR), random forest (RF), and extremely randomized trees (ET). The configurations of SVM and LR follow those in [26]. For RF and ET, the number of trees is set to 300.

The second category is designed to isolate the benefit of evolutionary search. Specifically, the same sliding-window extraction adopted in the proposed method is applied to compute a fixed set of predefined features from each series. An ET classifier is then trained on the transformed training set and evaluated on the transformed test set. This category comprises seven methods: Mean+ET, Std+ET, Median+ET, Iqr+ET, Min+ET, Max+ET, and Slope+ET. ET is chosen as the classifier to ensure a fair comparison, since it is also used in the proposed method. Comparing these with GP-evolved features helps assess whether evolutionary search yields additional gains beyond individual hand-crafted operators.

The third category includes three feature-based TSC approaches, selected to provide strong baselines. The first is Catch22 [9], which extracts a set of pre-defined time series characteristics. As Catch22 does not prescribe a specific classifier in the paper, ET with 300 trees is adopted for consistency. The second baseline is PWFTS [27], which is based on a fuzzy probabilistic representation. Its evaluation follows the protocol in the original paper, including the random forest classifier reported therein. The third approach is CBD [12], which is based on deep neural networks and is evaluated using the fine-tuning protocol described in its paper.

TABLE III: Parameter Settings of the Proposed GP Method

Parameter	Value	Parameter	Value
Maximal number of generations	50	Crossover Rate	0.80
Population Size	100	Mutation Rate	0.19
Tournament Size	3	Elitism Rate	0.01
Tree Minimum Depth	2	Tree Maximum Depth	5

TABLE IV: Overall comparison on the 31 UCR datasets

Method	Avg. Rank	#Top-1	#Top-3	Proposed vs. Baselines
SVM	10.06	3	6	$25(+)/2(\sim)/4(-)$
LR	8.87	5	7	$24(+)/3(\sim)/4(-)$
RF	7.19	3	5	$23(+)/7(\sim)/1(-)$
ET	5.87	3	5	$21(+)/9(\sim)/1(-)$
Mean+ET	8.84	3	3	$27(+)/3(\sim)/1(-)$
Std+ET	6.87	3	8	$24(+)/3(\sim)/4(-)$
Median+ET	8.42	4	5	$24(+)/4(\sim)/3(-)$
Iqr+ET	8.97	1	2	$26(+)/5(\sim)/0(-)$
Min+ET	10.74	2	3	$27(+)/4(\sim)/0(-)$
Max+ET	7.68	4	6	$24(+)/4(\sim)/3(-)$
Slope+ET	7.61	1	4	$25(+)/4(\sim)/2(-)$
Catch22	5.87	7	13	$21(+)/5(\sim)/5(-)$
PWFTS	4.29	**9**	19	$15(+)/3(\sim)/13(-)$
CBD	3.52	**9**	16	$15(+)/6(\sim)/10(-)$
Proposed	**3.16**	5	**23**	–

C. GP Settings

The GP settings of the proposed method are summarised in Table III. Considering the stochastic nature of evolution, 30 independent runs are performed, and the mean performance is reported as the final result.

V. Results and Analysis

A. Overall Comparison

Table IV provides an overall comparison of all methods on the 31 UCR datasets using three indicators. Avg. Rank denotes the average ranking of a method across all datasets. Specifically, on each individual dataset, methods are ranked based on their mean test accuracy, and these ranks are subsequently averaged across the 31 datasets. #Top-1 and #Top-3 denote the numbers of datasets on which a method attains the best and top-three performance, respectively. $x(+)/y(\sim)/z(-)$ indicate the numbers of datasets where the proposed method is significantly better than, similar to, or significantly worse than the corresponding method, respectively, according to the Wilcoxon rank-sum test at the 5% significance level.

Overall, the proposed method achieves the best average rank (3.16) among all compared methods and attains the largest Top-3 count (23). Compared with ET, the proposed method achieves a better average rank and is significantly better than ET on 21 datasets and worse on only 1 dataset. This indicates that the learned features can provide consistent gains for the ET classifier. In addition, the proposed method also performs favourably against baselines that rely on a single fixed extraction operator, suggesting that evolutionary composition of operators offers additional modelling capacity beyond any individual hand-crafted operator.

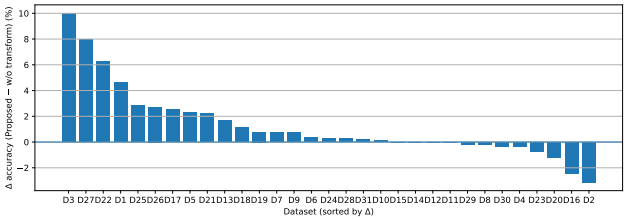

Fig. 3: Ablation on the domain transform layer across the 31 aforementioned UCR datasets. The bar chart reports the accuracy difference Δ = Proposed − w/o transform for each dataset, sorted in descending order. Positive values indicate that the domain transform layer improves performance.

B. Ablation on the Domain Transform Layer

To quantify the contribution of the domain transform layer, an ablation study is conducted by removing this layer and forcing all feature extraction operators to act directly on the time-domain (w/o transform). Fig. 3 reports the per-dataset comparison on 31 datasets. The full method attains higher accuracy on 19 out of 31 datasets, with several showing clear gains when the transform layer is enabled. For example, the performance of BirdChicken (D3) and Symbols (D27) are improved by 10 and 7.97 percentage points, respectively. These results suggest that mapping the raw series into alternative domains can make salient class-discriminative information easier to capture, which then benefits subsequent operators. In contrast, the full method performs worse on 10 datasets, likely because the time-domain representation is already sufficient for these problems and additional transforms may obscure discriminative information. Moreover, the performance drops are generally modest, with only two datasets declining by more than 2 percentage points.

C. Evolved Program Analysis

An evolved program from FacesUCR dataset is used for further analysis to illustrate how discriminative features are learned. Fig. 4 visualises the corresponding program tree, which follows the layered design in Section III-B. This program adopts **Root3**, yielding three feature learning branches. The left and middle branches perform feature extraction directly on the raw series: the left applies **Std** followed by **Min**, whereas the middle applies only **Min**. The right branch operates on a frequency-domain representation and applies the **Mean** operator. Across these branches, four distinct sliding-window lengths (9, 17, 43, and 51) are used to capture information at multiple scales. Such a branched composition increases the diversity of the learned features while preserving the interpretability of the evolved program. When the resulting features are fed into the ET classifier, the test accuracy reaches 89.22%, compared with 83.39% when ET is applied directly to the raw series.

VI. Conclusions

The goal of this paper is to develop a new STGP-based feature learning method for TSC, which has been success-

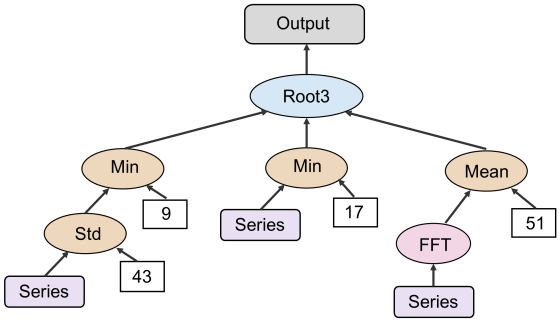

Fig. 4: An example evolved program on FacesUCR.

fully achieved by proposing a strongly typed GP framework with a novel layered program structure. Experiments on 31 datasets from the UCR Archive demonstrate the effectiveness of the proposed approach. Our method achieves the best average rank of 3.16 against fourteen baselines. Notably, the proposed method significantly outperform the ET classifier on 21 datasets, confirming that the evolved features provide substantial additional discriminative power. The ablation study also validate the contribution of the domain transform layer in capturing salient patterns that are difficult to detect in the raw time domain. Beyond accuracy, a key advantage of our method lies in its interpretability. The evolved program trees allow practitioners to explicitly trace and understand the decision-making process.

Although this study demonstrates the potential of STGP for feature learning in TSC, there remains substantial room for further research. One direction is to extend the function set in the feature extraction layer. Shape-related characteristics are crucial for TSC, yet the current implementation includes only a simple slope operator for capturing shape information. Another direction is to improve robustness to the noise segments that commonly occur in real-world time series data.

REFERENCES

[1] C. Ji, M. Du, Y. Wei, Y. Hu, S. Liu, L. Pan, and X. Zheng, "Time series classification with random temporal features," *Journal of King Saud University-Computer and Information Sciences*, vol. 35, no. 9, p. 101783, 2023.

[2] X. Cheng, M. Luo, K. Chen, J. Sun, and Y. Wu, "Intra-annual vegetation changes and spatial variation in china over the past two decades based on remote sensing and time-series clustering," *Environmental Monitoring and Assessment*, vol. 196, no. 7, p. 675, 2024.

[3] Y. Nie, N. H. Nguyen, P. Sinthong, and J. Kalagnanam, "A time series is worth 64 words: Long-term forecasting with transformers," in *The Eleventh International Conference on Learning Representations*, 2023. [Online]. Available: https://openreview.net/forum?id=Jbdc0vTOcol

[4] Z. Zamanzadeh Darban, G. I. Webb, S. Pan, C. Aggarwal, and M. Salehi, "Deep learning for time series anomaly detection: A survey," *ACM Computing Surveys*, vol. 57, no. 1, pp. 1–42, 2024.

[5] W. K. Wang, I. Chen, L. Hershkovich, J. Yang, A. Shetty, G. Singh, Y. Jiang, A. Kotla, J. Z. Shang, R. Yerrabelli *et al.*, "A systematic review of time series classification techniques used in biomedical applications," *Sensors*, vol. 22, no. 20, p. 8016, 2022.

[6] X. Bao, Y. Zheng, L. Chen, D. Wu, X. Chen, and Y. Liu, "Abnormal pattern recognition for online inspection in manufacturing process based on multi-scale time series classification," *Journal of Manufacturing Systems*, vol. 76, pp. 457–477, 2024.

[7] X. Xi, E. Keogh, C. Shelton, L. Wei, and C. A. Ratanamahatana, "Fast time series classification using numerosity reduction," in *Proceedings of the 23rd International Conference on Machine Learning*, 2006, pp. 1033–1040.

[8] C. Ji, Y. Wei, and X. Zheng, "Shapelet selection for time series classification," *Applied Soft Computing*, vol. 167, p. 112431, 2024.

[9] C. H. Lubba, S. S. Sethi, P. Knaute, S. R. Schultz, B. D. Fulcher, and N. S. Jones, "catch22: Canonical time-series characteristics: Selected through highly comparative time-series analysis," *Data mining and knowledge discovery*, vol. 33, no. 6, pp. 1821–1852, 2019.

[10] C. Rudin, "Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead," *Nature machine intelligence*, vol. 1, no. 5, pp. 206–215, 2019.

[11] Q. Ma, Z. Liu, Z. Zheng, Z. Huang, S. Zhu, Z. Yu, and J. T. Kwok, "A survey on time-series pre-trained models," *IEEE Transactions on Knowledge and Data Engineering*, vol. 36, no. 12, pp. 7536–7555, 2024.

[12] Y. Han, H. Wang, Y. Hu, Y. Gong, X. Song, and W. Guan, "Content-aware balanced spectrum encoding in masked modeling for time series classification," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 39, no. 16, 2025, pp. 17 059–17 067.

[13] A. B. Arrieta, N. Díaz-Rodríguez, J. Del Ser, A. Bennetot, S. Tabik, A. Barbado, S. García, S. Gil-López, D. Molina, R. Benjamins *et al.*, "Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai," *Information Fusion*, vol. 58, pp. 82–115, 2020.

[14] D. J. Montana, "Strongly typed genetic programming," *Evolutionary Computation*, vol. 3, no. 2, pp. 199–230, 06 1995.

[15] Z. Wu, B. Xue, and M. Zhang, "Multitree gp-based feature learning for multimodal medical image classification," in *2025 IEEE Congress on Evolutionary Computation*. IEEE, 2025, pp. 1–8.

[16] Y. Bi, B. Xue, and M. Zhang, "Genetic programming with image-related operators and a flexible program structure for feature learning in image classification," *IEEE Transactions on Evolutionary Computation*, vol. 25, no. 1, pp. 87–101, 2020.

[17] Z. Wu, B. Xue, and M. Zhang, "Multitree genetic programming for multimodal learning in multimodal medical image classification," *IEEE Transactions on Evolutionary Computation*, 2025.

[18] B. Peng, S. Wan, Y. Bi, B. Xue, and M. Zhang, "Automatic feature extraction and construction using genetic programming for rotating machinery fault diagnosis," *IEEE transactions on Cybernetics*, vol. 51, no. 10, pp. 4909–4923, 2020.

[19] M. Barandas, D. Folgado, L. Fernandes, S. Santos, M. Abreu, P. Bota, H. Liu, T. Schultz, and H. Gamboa, "Tsfel: Time series feature extraction library," *SoftwareX*, vol. 11, p. 100456, 2020.

[20] S. Roschmann, Q. Bouniot, and Z. Akata, "Time series representations for classification lie hidden in pretrained vision transformers," in *Recent Advances in Time Series Foundation Models Have We Reached the 'BERT Moment'?*, 2025. [Online]. Available: https://openreview.net/forum?id=niZwynKRi6

[21] J. R. Koza, *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. Cambridge, MA, USA: MIT Press, 1992.

[22] E. Fu and Y. Hu, "Frequency-masked embedding inference: A non-contrastive approach for time series representation learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 39, no. 16, 2025, pp. 16 639–16 647.

[23] E. J. Keogh and M. J. Pazzani, "Derivative dynamic time warping," in *SDM*, 2001. [Online]. Available: https://api.semanticscholar.org/CorpusID:6611383

[24] P. Geurts, D. Ernst, and L. Wehenkel, "Extremely randomized trees," *Machine Learning*, vol. 63, no. 1, pp. 3–42, 2006.

[25] H. A. Dau, E. Keogh, K. Kamgar, C.-C. M. Yeh, Y. Zhu, S. Gharghabi, C. A. Ratanamahatana, Yanping, B. Hu, N. Begum, A. Bagnall, A. Mueen, G. Batista, and Hexagon-ML, "The ucr time series classification archive," October 2018, https://www.cs.ucr.edu/~eamonn/time_series_data_2018/.

[26] Y. Bi, B. Xue, and M. Zhang, "An evolutionary deep learning approach using genetic programming with convolution operators for image classification," in *2019 IEEE Congress on Evolutionary Computation*, 2019, pp. 3197–3204.

[27] F. J. Erazo-Costa, P. C. L. Silva, and F. G. Guimarães, "A fuzzy-probabilistic representation learning method for time series classification," *IEEE Transactions on Fuzzy Systems*, vol. 32, no. 5, pp. 2940–2952, 2024.