Training-Free Time Series Classification via In-Context Reasoning with LLM Agents

Songyuan Sui[1], Zihang Xu[1], Yu-Neng Chuang[1],
Kwei-Herng Lai[2], Xia Hu[1]
[1]Rice University
{Songyuan.Sui, Zihang.Xu, Yu-Neng.Chuang, Xia.Hu}@rice.edu
[2]Microsoft
henrylai@microsoft.com

Abstract

Time series classification (TSC) spans diverse application scenarios, yet labeled data are often scarce, making task-specific training costly and inflexible. Recent reasoning-oriented large language models (LLMs) show promise in understanding temporal patterns, but purely zero-shot usage remains suboptimal. We propose FETA, a multi-agent framework for training-free TSC via exemplar-based in-context reasoning. FETA decomposes a multivariate series into channel-wise subproblems, retrieves a few structurally similar labeled examples for each channel, and leverages a reasoning LLM to compare the query against these exemplars, producing channel-level labels with self-assessed confidences; a confidence-weighted aggregator then fuses all channel decisions. This design eliminates the need for pretraining or fine-tuning, improves efficiency by pruning irrelevant channels and controlling input length, and enhances interpretability through exemplar grounding and confidence estimation. On nine challenging UEA datasets, FETA achieves strong accuracy under a fully training-free setting, surpassing multiple trained baselines. These results demonstrate that a multi-agent in-context reasoning framework can transform LLMs into competitive, plug-and-play TSC solvers without any parameter training. The code is available at https://github.com/SongyuanSui/FETATSC.

1 Introduction

Time series classification (TSC) is a long-standing and fundamental task with wide applications ranging from healthcare (Wang et al., 2024a), finance (Liang et al., 2023) and human activity recognition (Ghorrati et al., 2024) to industrial monitoring (Farahani et al., 2025). Dominant solutions learn a trainable mapping from raw sequences to labels, either end-to-end with deep neural networks (Karim et al., 2019; Zerveas et al., 2020) or in two stages via representation learning that pretrains an encoder and then learns a downstream task-specific model (Eldele et al., 2021; Liu and Chen, 2023). While effective, these pipelines require dataset-specific training or fine-tuning, incurring nontrivial computational cost, hyperparameter tuning, and domain adaptation effort. Recently, Large Language Models (LLMs) have shown remarkable performance across a large variety of tasks beyond natural language processing (NLP) (Yang et al., 2023). Studies that leverage LLMs for time series often still relies on additional training like fine-tuning (Xue and Salim, 2023) and tokenization into patches (Nie et al., 2023; Chang et al., 2025), limiting their generalization ability and plug-and-play usability. Moreover, the scarcity of labeled time series often makes such training insufficiently robust. For instance, Zhou et al. (2025b) introduces a method to augment training data for time series representation learning, highlighting the difficulty of obtaining broadly generalizable models when data is limited.

More recently, reasoning-oriented LLMs such as GPT-o1 (OpenAI et al., 2024), DeepSeek-R1 (DeepSeek-AI et al., 2025), and Qwen3 (Yang et al., 2025) have demonstrated the ability to capture temporal patterns such as trends, seasonality, and fluctuations, suggesting that they can process time series data through analogical reasoning (Chen et al., 2024; Zhou et al., 2025a). However, existing studies also indicate that applying them in a purely zero-shot or fixed-prompt manner often yields suboptimal performance (Merrill et al., 2024; Liu et al., 2025). This gap highlights an equally important but overlooked capability of LLMs in the time series domain: *in-context learning*. Unlike conventional time series models, LLMs offer novel paradigms of learning and knowledge representation, particularly through in-context learning, which enables them to dynamically adapt their output based on exemplars without any parameter updates (Mohamed et al., 2025).

We thereby ask a different question: *Instead of*

further expanding or retraining models, can we perform time series classification without any training? We answer in the affirmative with **FETA** (training-Free timE series classificaTion with LLM Agents), a multi-agent framework that replaces model fitting with exemplar-based, in-context reasoning.

FETA decomposes a multivariate time series into channel-wise subproblems, retrieves a few similar labeled sequences per channel using Dynamic Time Warping (DTW) (Berndt and Clifford, 1994), and leverages a reasoning LLM to compare the query against these exemplars, producing channel-level predictions with associated confidences. A confidence-weighted fusion then aggregates these outputs into the final decision. This design avoids pretraining or fine-tuning, improves efficiency by pruning irrelevant channels and controlling input length, and remains interpretable by grounding predictions in exemplars and confidence scores. The entire pipeline is built within a multi-agent framework, where each component operates independently yet contributes to a coherent, modular workflow.

To comprehensively evaluate the effectiveness of FETA, we conduct experiments across nine representative and challenging UEA benchmarks. The results demonstrate that FETA consistently achieves strong accuracy under a fully training-free setting, surpassing a variety of classical, deep learning, and representation learning baselines. These findings validate the effectiveness of our key architectural design: (i) decomposing multivariate sequences into channel-wise subproblems to focus reasoning on the most informative signals; (ii) retrieving structurally aligned exemplars to ground in-context reasoning; and (iii) aggregating per-channel decisions through confidence-weighted fusion. Together, these components enable robust and interpretable time series classification without any model training or parameter updates.

In summary, we propose FETA, a novel multi-agent collaboration framework for training-free time series classification via in-context reasoning with the following contributions:

- We introduce the first training-free multi-agent framework for multivariate time series classification that leverages LLMs' in-context reasoning ability, eliminating the need for pre-training, fine-tuning, or task-specific models.
- We design a multi-agent pipeline that integrates channel decomposition, exemplar retrieval, LLM in-context reasoning, and confidence-weighted aggregation, enabling efficient, interpretable, and modular classification.
- We conduct extensive experiments on nine UEA benchmarks with both reasoning and non-reasoning LLMs, showing that FETA not only achieves state-of-the-art accuracy with reasoning LLMs but also delivers competitive results with smaller non-reasoning models, highlighting the effectiveness of our exemplar-based in-context learning approach.

2 Preliminaries

In this section, we present the notations used throughout the paper, followed by the definitions of multivariate time series and the time series classification task. We then introduce the basic formulation of the multi-agent framework, which will be leveraged in the proposed method to enable training-free classification with large language models.

2.1 Multivariate Time Series Definition

A *multivariate time series* is defined as

$$\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_T] \in \mathbb{R}^{T \times C}, \quad \mathbf{x}_t \in \mathbb{R}^C, \quad (1)$$

where T is the number of time steps and C is the number of variables (channels). Each time step vector is written as

$$\mathbf{x}_t = [x_t^{(1)}, x_t^{(2)}, \ldots, x_t^{(C)}]^\top, \quad (2)$$

which denotes the measurements of all C variables at time step t. This work considers datasets where each time series $\mathbf{X}$ is associated with a ground-truth label $y \in \mathcal{Y}$.

2.2 Time Series Classification Task

Given a time series $\mathbf{X} \in \mathbb{R}^{T \times C}$, the time series classification task seeks a mapping

$$f : \mathbb{R}^{T \times C} \to \mathcal{Y}, \quad (3)$$

where $\mathcal{Y} = \{1, 2, \ldots, K\}$ is the finite set of K target classes. Traditional approaches parameterize f as a trainable model f_θ and learn parameters θ by minimizing a supervised loss on labeled training data. In our setting, f is instead implemented by a multi-agent framework that relies on LLMs' reasoning ability to directly predict labels without parameter training.

2.3 Multi-Agent Framework Basics

We define a multi-agent framework as a set of LLM-powered agents

$$\mathcal{A} = \{A_1, A_2, \ldots, A_M\}, \tag{4}$$

where each agent A_m takes an input I_m and produces an output

$$O_m = A_m(I_m). \tag{5}$$

Agents interact via a communication graph $\mathcal{G}_{\text{comm}} = (\mathcal{A}, \mathcal{E})$, where an edge $(A_i, A_j) \in \mathcal{E}$ indicates that the output of A_i is passed to A_j. The final prediction $\hat{y} \in \mathcal{Y}$ is obtained collaboratively as

$$\hat{y} = \mathcal{F}(\mathcal{A}, \mathcal{G}_{\text{comm}}), \tag{6}$$

with $\mathcal{F}$ denoting the overall decision function induced by agent interactions.

3 Related Work

We review previous work from three main aspects: Time Series Classification, Large Language Models for Time Series Analysis, and Multi-Agent Collaboration.

Time Series Classification. Time series classification is a long-standing task in time series analysis. Classical methods often rely on distance-based algorithms such as k-Nearest Neighbors (k-NN) (Lee et al., 2012) with metrics like Euclidean distance or Dynamic Time Warping. With deep learning, a wide range of models have been proposed, including RNNs (Lai et al., 2018), LSTMs (Karim et al., 2017), CNNs (Luo and Wang, 2024), Transformers (Liu et al., 2021b; Le et al., 2024), and hybrid architectures (Khan et al., 2021; Zhang et al., 2024). While effective, these models typically require dataset-specific training, limiting scalability. More recently, representation learning approaches pretrain encoders to map raw signals into embedding spaces for downstream tasks. Examples include TS-TCC (Eldele et al., 2021) with contrastive temporal context, T-Loss (Franceschi et al., 2020) with autoregressive prediction, TS2Vec (Yue et al., 2022) with hierarchical contrastive objectives, and Times-URL (Liu and Chen, 2023) with unified self-supervised objectives. These methods generalize well across benchmarks but still require training a dedicated encoder and then fine-tuning or attaching a task-specific model like classifier, making the two-stage pipeline costly and less flexible.

Large Language Models for Time Series Analysis. Recent advancements in LLMs lead to growing interest in their application to time series analysis. Early methods adapt pretrained LLMs to the textualized time series data by fine-tuning them with task-specific datasets (Xue and Salim, 2023). A more popular strategy is to convert raw time series into patches that serve as input tokens (Nie et al., 2023). GPT4TS (Zhou et al., 2023) reprograms the input time series into text style representations and augments the context with declarative prompts. LLM4TS (Chang et al., 2025) uses a two-stage fine-tuning strategy to align patched time series data with LLMs and the downstream tasks. InstructTime (Cheng et al., 2024) reframes time series classification as a multimodal instruction-following task by discretizing signals into tokens and fine-tuning a language model to generate textual labels. However, despite promising results, these approaches still rely on additional training, which incurs considerable computational cost. Moreover, while LLMs are inherently valued for their cross-task generality, most existing studies tend to yield time series forecasting models. Extending these models to other time series applications such as classification and anomaly detection requires extra adaptation or retraining, limiting their universality.

Multi-Agent Collaboration. Recent studies enable multiple LLM-powered agents to address complex problems through extensive collaboration and collective intelligence (Wu et al., 2023). These approaches expand LLMs' reasoning capabilities beyond NLP domains. Research has applied multi-agent collaboration in programming (Xie et al., 2024; Wang et al., 2025), structured data processing (Wang et al., 2024b; Sui et al., 2025), and computer vision (Jiang et al., 2024; Li et al., 2025). Recently, researchers have also begun exploring multi-agent collaboration for time series analysis. TS-Reasoner (Ye et al., 2025) introduces a multi-agent framework that leverages an LLM as a task decomposer and collaborates with specialized operators for multi-step time series inference, but it focuses on complex reasoning and relies heavily on predefined toolkits. MERIT (Zhou et al., 2025b) proposes a multi-agent framework to augment training data for time series representation learning, but still requires training a dedicated encoder, limiting plug-and-play applicability. ReasonTSC (Zhou et al., 2025a) integrates typical time series analysis models with LLM-based reasoning to improve classification, yet still depends on well-trained base

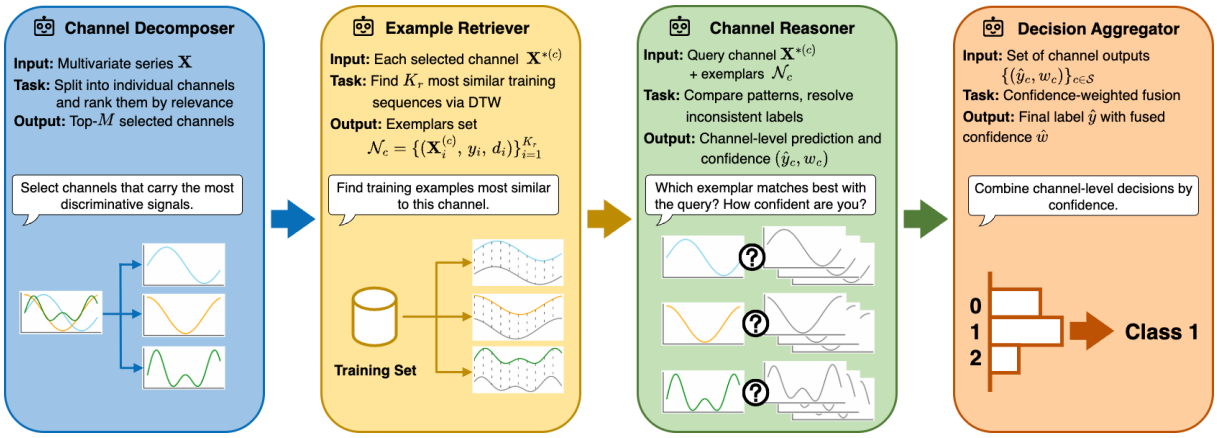

Figure 1: Overview of Our FETA Framework.

time series classifiers.

4 Methodology

4.1 Overview

We propose a multi-agent framework that decomposes multivariate time series classification into modular sub-tasks, with each agent responsible for a specific stage of the pipeline. As shown in Figure 1, the framework consists of four specialized agents: **Channel Decomposer**, **Example Retriever**, **Channel Reasoner**, and **Decision Aggregator**. The Channel Decomposer isolates individual univariate channels from the multivariate input and filters out relatively irrelevant ones, ensuring that downstream reasoning focuses only on informative signals. This design enables the system to function in a fully training-free manner, leveraging nearest-neighbor evidence and lightweight in-context reasoning rather than model fine-tuning. Together, these agents form a coherent pipeline for training-free time series classification, which we describe in detail in the following sections.

4.2 Channel Decomposer: Selection and Preprocessing

The Channel Decomposer serves as the entry point of the entire agentic workflow. Time series classification spans a wide range of application scenarios, where sequence lengths can vary from a few dozen to several thousand time steps and the number of channels can range from a handful to thousands. Such heterogeneity poses significant challenges for direct LLM-based reasoning. The module therefore has two purposes: (1) enforcing channel independence, i.e., representing each input sequence using only one channel, which has been shown to preserve discriminative temporal patterns in Transformer-based architectures (Nie et al., 2023); and (2) pruning irrelevant or redundant channels to reduce reasoning overhead, particularly when the original dimensionality is large. In this way, the module optimizes the input for downstream LLM-based reasoning by reducing noise and focusing on the most informative channels.

Formally, given $\mathbf{X} \in \mathbb{R}^{T\times C}$, the Channel Decomposer isolates channel-wise sequences

$$\mathbf{X}^{(c)} = [x_1^{(c)}, \ldots, x_T^{(c)}], \quad c \in \{1, \ldots, C\},$$

and ranks channels by a fused, training-free relevance score computed on the training set. The goal is to identify the most informative channels, reduce redundancy, and ensure that downstream agents focus on dimensions most correlated with class distinctions. The following subsections detail this process.

Length normalization and per-sample z-normalization. Time series from different channels and samples often have varying lengths and scales. To make them comparable, we follow the preprocessing in (Dau et al., 2019) to normalize each channel independently:

$$\tilde{\mathbf{X}}^{(c)} = \frac{\mathbf{X}^{(c)} - \mu}{\sigma}, \tag{7}$$

where μ and σ denote the mean and standard deviation of $\mathbf{X}^{(c)}$. We then uniformly subsample each sequence to a fixed target length L via uniform index mapping. This step prevents overly long sequences from inflating the LLM input size and degrading reasoning efficiency, while preserving global shape and trends through uniform subsampling. The re-

sulting per-channel training set is denoted as

$$\mathcal{D} = \{(\tilde{\mathbf{X}}_i^{(c)}, y_i)\}_{i=1}^{N}.$$

This preprocessing ensures that subsequent channel evaluation operates on standardized and comparable inputs.

Prototype-margin score (B). An informative channel should maximize inter-class separation while minimizing intra-class variance. To quantify this, we compute a prototype-margin score that compares inter-class separation against intra-class compactness, akin to Fisher discriminant ratio in classical feature selection (Gu et al., 2012). Specifically, for each class $y \in \mathcal{Y}$ we compute the class centroid

$$\boldsymbol{\mu}_y = \frac{1}{|\mathcal{D}_y|} \sum_{i \in \mathcal{D}_y} \tilde{\mathbf{X}}_i^{(c)}, \tag{8}$$

the average within-class spread

$$\mathrm{W} = \frac{1}{\sum_y |\mathcal{D}_y|} \sum_y \sum_{i \in \mathcal{D}_y} \|\tilde{\mathbf{X}}_i^{(c)} - \boldsymbol{\mu}_y\|_2, \tag{9}$$

and the mean pairwise centroid distance

$$\mathrm{B} = \frac{2}{K(K-1)} \sum_{y<y'} \|\boldsymbol{\mu}_y - \boldsymbol{\mu}_{y'}\|_2. \tag{10}$$

The prototype-margin score is then

$$B_c = \frac{\mathrm{B}}{\mathrm{W} + \varepsilon}, \quad \varepsilon > 0. \tag{11}$$

A higher B_c indicates that channel c better separates classes with compact clusters, making it more discriminative.

Approximate 1NN leave-one-out accuracy (C). Beyond cluster structure, we also evaluate how well each channel supports classification via nearest-neighbor retrieval. To this end, we adopt leave-one-out 1NN accuracy as a channel-level discriminability measure, which is a standard evaluation criterion in time series classification (Lines and Bagnall, 2015). Specifically, we use a probe subset of size n_{probe} for efficiency. For each probed index i, we find

$$j^\star = \arg\min_{j \neq i} \|\tilde{\mathbf{X}}_i^{(c)} - \tilde{\mathbf{X}}_j^{(c)}\|_2^2, \tag{12}$$

and check if $y_{j^\star} = y_i$. The channel score is

$$C_c = \frac{1}{n_{\text{probe}}} \sum_{i \in \text{probe}} \mathbf{1}[y_{j^\star} = y_i]. \tag{13}$$

Channels with higher C_c demonstrate stronger standalone classification ability, making them valuable candidates.

Score fusion and ranking. Since B_c emphasizes structural separation and C_c emphasizes predictive accuracy, we combine them to obtain a balanced channel relevance score. After standardizing both via z-scores, we compute

$$S_c = \alpha\, \mathrm{z}(B_c) + (1-\alpha)\, \mathrm{z}(C_c), \quad \alpha \in [0,1]. \tag{14}$$

Channels are then ranked by S_c, and the top-M are selected for downstream reasoning. This fusion balances structural separability and predictive accuracy, ensuring that the selected channels provide meaningful evidence for downstream reasoning.

In summary, the Channel Decomposer provides a lightweight, training-free mechanism to evaluate and select channels. By enforcing channel independence and pruning redundancy, the module yields a compact, informative subset of channels, reducing input dimensionality and streamlining downstream LLM reasoning.

4.3 Example Retriever: In-Context Exemplar Selection

After selecting a compact set of informative channels, the next step is to provide each channel agent with concrete exemplars that can guide in-context reasoning. The motivation here is aligned with in-context learning (Liu et al., 2021a): instead of training parameters, we supply the LLM with a small set of relevant examples, allowing it to adapt to the classification task by analogy. In the time series domain, this means retrieving similar labeled sequences from the training set so that the LLM can compare the query against concrete precedents. This design is supported by recent evidence that LLMs are capable of understanding fundamental temporal patterns such as trends, seasonality, and fluctuations (Zhou et al., 2025a), making it feasible to directly compare retrieved exemplars with the query sequence for classification. Meanwhile, labeled time series are often scarce; Zhou et al. (2025b) resort to multi-agent data augmentation to compensate for limited training data in representation learning. In contrast, our retriever operates *without* any model training or augmentation: only a handful of labeled exemplars are retrieved on-the-fly per query to ground reasoning.

Formally, for each selected channel $c \in \mathcal{S}$ and a test sequence $\mathbf{X}^{*(c)}$, the Example Retriever searches for its K_r most similar training sequences using Dynamic Time Warping (DTW) (Berndt and

Clifford, 1994). The retrieved set is

$$\mathcal{N}_c = \{(\mathbf{X}_i^{(c)}, y_i, d_i)\}_{i=1}^{K_r}, \tag{15}$$

where $d_i = \mathrm{DTW}(\mathbf{X}^{*(c)}, \mathbf{X}_i^{(c)})$ denotes the alignment-based distance between the test and the i-th training sequence. DTW is particularly suitable for this role because it robustly accounts for local temporal distortions, variable speeds, and phase shifts, allowing two sequences with similar overall patterns but misaligned time indices to be matched effectively. This makes it a long-standing standard for time series similarity, and far more reliable than rigid measures such as Euclidean distance.

The retrieved exemplars $\mathcal{N}_c$ provide both labels $\{y_i\}$ and DTW distances $\{d_i\}$. Since the K_r neighbors may disagree, we do not rely on majority voting; instead, we present them as diverse candidates and summarize their evidence (e.g., label histograms, neighbor-distance profiles, and representative exemplars) to form the in-context "demonstration set" for the Channel Reasoner.

In short, the Example Retriever enables training-free, interpretable classification by grounding each channel agent in nearest-neighbor evidence rather than learned representations. Leveraging DTW ensures that similarity reflects structural alignment rather than rigid pointwise matching, yielding robust, informative exemplars, even under label-scarce regimes.

4.4 Channel Reasoner: Pattern-Based In-Context Reasoning

The K_r nearest neighbors for each channel provide concrete exemplars, but their labels may disagree; a naive majority vote can overlook temporal nuances that differentiate classes. The Channel Reasoner therefore leverages a reasoning LLM to perform in-context comparison of temporal patterns: given the query sequence and its retrieved exemplars, the agent contrasts trends, fluctuations, alignments, and other descriptive characteristics to adjudicate among conflicting candidates.

For each selected channel $c \in \mathcal{S}$, we instantiate an agent $A_c \in \mathcal{A}$. The agent is presented with the query $\mathbf{X}^{*(c)}$ and the retrieved exemplar set

$$\mathcal{N}_c = \{(\mathbf{X}_i^{(c)}, y_i, d_i)\}_{i=1}^{K_r},$$

where $d_i = \mathrm{DTW}(\mathbf{X}^{*(c)}, \mathbf{X}_i^{(c)})$ are alignment-based distances from the Example Retriever. Based on this input, the reasoning LLM outputs a channel-level decision

$$O_c = (\hat{y}_c, w_c) = A_c\big(\mathbf{X}^{*(c)}, \mathcal{N}_c\big), \tag{16}$$

where $\hat{y}_c \in \mathcal{Y}$ is the predicted label for channel c, and $w_c \in [0, 1]$ is a confidence score explicitly generated by the model. Each channel agent operates independently, and all reasoners run in parallel without inter-agent communication, enabling scalable and efficient inference across high-dimensional time series.

Because we employ a reasoning LLM, the agent can internally incorporate descriptive statistics of the query $\mathbf{X}^{*(c)}$ (e.g., mean, variance, extrema, turning points) without explicit feature engineering. The prompt instructs A_c to: (i) compare the query with each exemplar, using DTW distances $\{d_i\}$ as similarity evidence; (ii) identify the exemplars whose temporal patterns align best with the query; and (iii) justify the chosen label while providing a self-assessed confidence w_c. This converts raw nearest-neighbor evidence into a principled, interpretable decision rather than a heuristic vote.

In summary, the Channel Reasoner operationalizes exemplar-based in-context reasoning at the channel level. By combining retrieved neighbors with intrinsic descriptive cues of the query, the reasoning LLM produces both a label $\hat{y}_c$ and a confidence score w_c, which together form the structured outputs passed to the Decision Aggregator.

4.5 Decision Aggregator: Confidence-Weighted Fusion

The final stage aggregates channel-level predictions into a single decision. Because different channels may provide complementary or conflicting evidence, the aggregator combines labels $\{\hat{y}_c\}_{c\in\mathcal{S}}$ and confidences $\{w_c\}_{c\in\mathcal{S}}$ into a robust consensus via confidence-weighted fusion.

Given the candidate class set $\mathcal{Y}$ and all agent outputs, we first discard invalid predictions (labels not in $\mathcal{Y}$). If all valid agents predict the same label, we enter *consensus mode* and compute the fused confidence as

$$\hat{w} \;=\; 1 - \prod_{c\in\mathcal{S}} (1 - w_c),$$

which increases with both the number of agreeing agents and their confidences. The final decision is this common label $\hat{y}$ with confidence $\min(0.99, \hat{w})$.

If predictions are not unanimous, we switch to *confidence-weighted mode*. Let the clipped confidence be $\tilde{w}_c = \min(\text{clip}_{\text{hi}}, \max(\text{clip}_{\text{lo}}, w_c))$ with fixed bounds $0 < \text{clip}_{\text{lo}} < \text{clip}_{\text{hi}} < 1$. Each agent contributes $\tilde{w}_c$ to its predicted class and a small smoothing weight $\varepsilon > 0$ to other classes:

$$s_y = \sum_{c \in \mathcal{S}} \begin{cases} \tilde{w}_c, & \text{if } \hat{y}_c = y, \\ \varepsilon, & \text{otherwise.} \end{cases}$$

We then normalize scores and select

$$\hat{y} = \arg\max_{y \in \mathcal{Y}} s_y, \qquad \hat{w} = \frac{s_{\hat{y}}}{\sum_{y \in \mathcal{Y}} s_y}.$$

If no valid agent remains after filtering, the aggregator returns a null decision with zero confidence.

In short, the aggregator balances agreement and diversity: it boosts confidence under strong consensus and re-weights votes by calibrated confidence under disagreement. This yields a final decision that is robust and interpretable, reflecting the strength of cross-channel evidence.

5 Experiments

In this section, we empirically evaluate the effectiveness of FETA.

5.1 Experimental Setup

Datasets. We evaluate on nine representative *and challenging* datasets from the UEA classification archive (Bagnall et al., 2018). This subset spans diverse application scenarios and exhibits substantial heterogeneity in sequence length, channel count, and number of classes. We adopt the official train/test splits and report classification accuracy. Notably, this selection is intentionally difficult: as shown in Table 1, most baselines obtain only modest average accuracy on these nine datasets, underscoring the challenge and leaving ample room for improvement.

Baselines. We compare our method against a various types of state-of-the-art baselines, including the distance-based method DTW (Berndt and Clifford, 1994), pattern-based method WEASEL+MUSE (Schäfer and Leser, 2018), deep learning model MLSTM-FCNs (Karim et al., 2019), transformer-based model TST (Zerveas et al., 2020), representation learning methods TS-TCC (Eldele et al., 2021), T-Loss (Franceschi et al., 2020), TS2Vec (Yue et al., 2022), Times URL (Liu and Chen, 2023), and the multi-agent method MERIT (Zhou et al., 2025b).

Implementation Details. We maintain the original training-test splits from the UEA archive. we adopt LLaMA 3.1, DeepSeek R1, and Qwen 3 as the backbone LLMs. Model configurations are detailed in Appendix C. Prompts are dataset-agnostic, as shown in Appendix D.

5.2 Main Results

Here, we compare FETA against diverse baselines on nine datasets. As shown in Table 1, all three FETA variants perform strongly despite requiring **no training**. FETA (Qwen3) achieves the best average accuracy at 47.3%, surpassing strong classical (WEASEL+MUSE 43.7%), representation-learning (MERIT 44.4%), and other deep/Transformer baselines. FETA (DeepSeek R1) ranks second overall with 46.6%, closely tracking Qwen3 and outperforming all non-FETA methods. Notably, even though FETA (LLaMA3.1) uses a smaller, non-reasoning model, it still reaches 38.9%, matching TST (38.9%) and clearly exceeding the direct DeepSeek R1 baseline (32.5%) without our pipeline.

Beyond averages, FETA delivers strong per-dataset results. On *AtrialFibrillation*, FETA (Qwen3) attains 46.7%, far above the best baseline (33.3%). On *ERing*, FETA (LLaMA3.1) reaches 24.4% vs. $\sim$13.3% for others. On *StandWalkJump*, FETA (DeepSeek R1) achieves 60.0%, substantially outperforming TS2Vec (40.0%) and TST (36.7%). These results indicate that the FETA pipeline extracts discriminative evidence effectively, independent of model scale or specialized reasoning capabilities.

Why does FETA work? We attribute the gains to four complementary factors. **(i) Channel decomposition and selection.** By isolating and ranking channels, FETA preserves the most informative signals while shortening overly long sequences, preventing inflated input length that would otherwise hinder LLM reasoning efficiency. **(ii) Exemplar-driven in-context reasoning.** DTW-based retrieval supplies pattern-aligned neighbors, enabling each channel agent to compare the query against concrete precedents and recognize trends, shifts, and fluctuations without a learned encoder. **(iii) Confidence-aware decision fusion.** Requiring the LLM to output a calibrated confidence per channel and aggregating via confidence-weighted fusion enhances interpretability (per-channel rationale + certainty) and more effectively consolidates het-

Dataset	DTW	WEASEL +MUSE	MLSTM -FCNs	TST	TS-TCC	T-Loss	TS2Vec	Times -URL	DeepSeek R1	MERIT	FETA (LLaMA3.1)	FETA (DeepSeek R1)	FETA (Qwen3)
AtrialFibrillation	22.0	33.3	26.7	26.7	26.7	26.7	26.7	26.7	20.0	33.3	33.3	40.0	**46.7**
ERing	13.3	13.3	13.3	13.3	13.3	13.3	13.3	13.3	13.3	13.3	**24.4**	23.7	23.0
EigenWorms	55.7	**89.0**	50.4	61.1	61.8	61.1	62.6	64.1	48.4	65.6	57.3	60.3	61.1
EthanolConcentration	29.3	**43.0**	37.3	32.3	33.1	32.7	33.8	34.2	21.7	35.0	28.5	36.2	38.0
FingerMovements	55.0	49.0	58.0	56.7	58.3	58.3	60.0	61.7	38.0	**63.3**	51.0	57.0	58.0
HandMovementDirection	27.8	36.5	36.5	30.6	33.3	30.6	33.3	36.1	25.7	**38.9**	25.7	35.1	37.8
MotorImagery	39.0	50.0	51.0	42.0	44.0	43.0	45.0	46.0	40.0	48.0	43.0	50.0	**52.0**
SelfRegulationSCP2	48.3	46.0	47.2	50.6	52.2	51.7	53.3	54.4	45.0	55.6	46.7	**56.7**	56.1
StandWalkJump	33.3	33.3	6.7	36.7	40.0	36.7	40.0	43.3	40.0	46.7	40.0	**60.0**	53.3
Average	35.9	43.7	36.3	38.9	40.3	39.3	40.9	42.2	32.5	44.4	38.9	46.6	**47.3**

Table 1: Comparison of classification accuracy (%) across datasets. Baselines cover diverse categories. FETA variants are compared under the same evaluation. Best results per dataset are in **bold**, and second-best results are underlined. FETA achieves the highest accuracy, driven by multivariate series decomposition, DTW-aligned exemplars retrieval, channel-level in-context reasoning, and confidence-weighted aggregation.

Dataset	FETA	w/o Decomposer	w/o Retriever	w/o Reasoner	w/o Aggregator
AtrialFibrillation	**40.0**	20.0	33.3	33.3	20.0
Δ	–	-20.0	-6.7	-6.7	-20.0
ERing	**23.7**	19.7	20.7	21.1	21.5
Δ	–	-4.0	-3.0	-2.6	-2.2
StandWalkJump	**60.0**	40.0	46.7	40.0	53.3
Δ	–	-20.0	-13.3	-20.0	-6.7
Average	**41.2**	26.6	33.6	31.5	31.6
Δ	–	-14.6	-7.6	-9.7	-9.6

Table 2: Ablation study of FETA (DeepSeek R1). Each Δ row shows the accuracy drop relative to the full model. FETA achieves optimal performance through its modular design, with each component contributing significantly to accurate classification.

erogeneous channel decisions than naive voting. **(iv) Modular multi-agent design.** By delegating distinct roles to different agents and consolidating their outputs through confidence-weighted aggregation, FETA effectively balances local channel-level insights with global consensus, yielding robust and interpretable classification.

5.3 Ablation Study: Deep Dive into FETA's Key Components and Mechanisms

To assess the contribution of each component, we conduct an ablation study on three representative datasets with DeepSeek R1.

We evaluate four variants: (i) **FETA w/o Decomposer**, which removes channel decomposition/selection and feeds all channels jointly to the LLM; (ii) **FETA w/o Retriever**, which replaces DTW-based nearest-neighbor retrieval with label-wise random sampling from the training set; (iii) **FETA w/o Reasoner**, which directly adopts the top-1 exemplar's label without LLM reasoning; and (iv) **FETA w/o Aggregator**, which replaces confidence-weighted fusion with majority voting.

As shown in Table 2, removing the Channel Decomposer yields the largest degradation (–14.6% on average), with severe drops on *AtrialFibrillation* and *StandWalkJump* (–20.0% each), underscoring the importance of selecting informative channels and controlling input length for efficient LLM reasoning. Eliminating DTW-based exemplar retrieval causes a notable decline (–7.6% on average), confirming that alignment-aware neighbors are crucial for grounding in-context reasoning. Disabling the Channel Reasoner reduces accuracy by –9.7% on average, indicating that the reasoning step is necessary to resolve label disagreements and compare temporal patterns beyond majority heuristics. Finally, removing the Decision Aggregator leads to a –9.6% average drop, particularly pronounced on *AtrialFibrillation* (–20.0%), highlighting the benefit of confidence-aware fusion over naive voting.

Overall, these findings show that each module plays a distinct and complementary role, and their synergy under the multi-agent design enables robust, training-free classification.

6 Conclusion

We present FETA, a multi-agent framework for training-free time series classification through exemplar-based in-context reasoning. By decomposing multivariate series, retrieving DTW-aligned exemplars, reasoning at the channel level, and fusing results via confidence-weighted aggregation, FETA achieves efficient, interpretable, and robust classification without any training. Experiments show that FETA delivers competitive or superior accuracy compared to trained models, highlighting the untapped potential of LLMs as plug-and-play time series learners. This work opens a new direction for integrating reasoning LLMs with structured temporal data under zero-training constraints.

Limitations

FETA is currently mainly designed and evaluated for multivariate time series with structured numerical signals. However, real-world applications often involve hybrid or heterogeneous data, where time series coexist with modalities such as text descriptions, images, or categorical metadata. Extending FETA's LLM calling and modular framework to handle such multimodal or cross-domain inputs while maintaining its training-free and interpretable characteristics remains an important direction for future research.

Ethics Statement

This work uses nine publicly available datasets from the UEA multivariate time series classification archive, which are widely adopted in prior studies and contain no personally identifiable information. No new data collection, labeling, or human annotation was conducted. Our experiments employ pretrained large language models (LLaMA 3.1, DeepSeek R1, and Qwen3) in inference-only settings without any fine-tuning or additional training. While we are not aware of dataset- or task-specific ethical concerns, we acknowledge general risks associated with pretrained LLMs, including potential propagation of social or cultural biases and the environmental cost of large-scale model deployment.

References

Anthony Bagnall, Hoang Anh Dau, Jason Lines, Michael Flynn, James Large, Aaron Bostrom, Paul Southam, and Eamonn Keogh. 2018. The uea multivariate time series classification archive, 2018. *Preprint*, arXiv:1811.00075.

Donald J Berndt and James Clifford. 1994. Using dynamic time warping to find patterns in time series. In *Proceedings of the 3rd international conference on knowledge discovery and data mining*, pages 359–370.

Ching Chang, Wei-Yao Wang, Wen-Chih Peng, and Tien-Fu Chen. 2025. Llm4ts: Aligning pre-trained llms as data-efficient time-series forecasters. *ACM Transactions on Intelligent Systems and Technology*, 16(3):1–20.

Baiting Chen, Zhimei Ren, and Lu Cheng. 2024. Conformalized time series with semantic features. *Advances in Neural Information Processing Systems*, 37:121449–121474.

Mingyue Cheng, Yiheng Chen, Qi Liu, Zhiding Liu, and Yucong Luo. 2024. Advancing time series classification with multimodal language modeling. *Preprint*, arXiv:2403.12371.

Hoang Anh Dau, Anthony Bagnall, Kaveh Kamgar, Chin-Chia Michael Yeh, Yan Zhu, Shaghayegh Gharghabi, Chotirat Ann Ratanamahatana, and Eamonn Keogh. 2019. The ucr time series archive. *Preprint*, arXiv:1810.07758.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, and 181 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *Preprint*, arXiv:2501.12948.

Emadeldeen Eldele, Mohamed Ragab, Zhenghua Chen, Min Wu, Chee Keong Kwoh, Xiaoli Li, and Cuntai Guan. 2021. Time-series representation learning via temporal and contextual contrasting. *Preprint*, arXiv:2106.14112.

Mojtaba A Farahani, MR McCormick, Ramy Harik, and Thorsten Wuest. 2025. Time-series classification in smart manufacturing systems: An experimental evaluation of state-of-the-art machine learning algorithms. *Robotics and Computer-Integrated Manufacturing*, 91:102839.

Jean-Yves Franceschi, Aymeric Dieuleveut, and Martin Jaggi. 2020. Unsupervised scalable representation learning for multivariate time series. *Preprint*, arXiv:1901.10738.

Zahra Ghorrati, Ahmad Esmaeili, and T Eric Matson. 2024. A multi-section hierarchical deep neural network model for time series classification: Applied to wearable sensor-based human activity recognition. *IEEE Access*, 12:137851–137869.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, and 542 others. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Quanquan Gu, Zhenhui Li, and Jiawei Han. 2012. Generalized fisher score for feature selection. *Preprint*, arXiv:1202.3725.

Bowen Jiang, Zhijun Zhuang, Shreyas S. Shivakumar, Dan Roth, and Camillo J. Taylor. 2024. Multi-agent vqa: Exploring multi-agent foundation models in zero-shot visual question answering. *Preprint*, arXiv:2403.14783.

Fazle Karim, Somshubra Majumdar, Houshang Darabi, and Shun Chen. 2017. Lstm fully convolutional networks for time series classification. *IEEE access*, 6:1662–1669.

Fazle Karim, Somshubra Majumdar, Houshang Darabi, and Samuel Harford. 2019. Multivariate lstm-fcns for time series classification. *Neural Networks*, 116:237–245.

Mehak Khan, Hongzhi Wang, Adnan Riaz, Aya Elfatyany, and Sajida Karim. 2021. Bidirectional lstm-rnn-based hybrid deep learning frameworks for univariate time series classification. *The Journal of Supercomputing*, 77(7):7021–7045.

Guokun Lai, Wei-Cheng Chang, Yiming Yang, and Hanxiao Liu. 2018. Modeling long- and short-term temporal patterns with deep neural networks. *Preprint*, arXiv:1703.07015.

Xuan-May Le, Ling Luo, Uwe Aickelin, and Minh-Tuan Tran. 2024. Shapeformer: Shapelet transformer for multivariate time series classification. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 1484–1494.

Yen-Hsien Lee, Chih-Ping Wei, Tsang-Hsiang Cheng, and Ching-Ting Yang. 2012. Nearest-neighbor-based approach to time-series classification. *Decision Support Systems*, 53(1):207–217.

Mingcheng Li, Xiaolu Hou, Ziyang Liu, Dingkang Yang, Ziyun Qian, Jiawei Chen, Jinjie Wei, Yue Jiang, Qingyao Xu, and Lihua Zhang. 2025. Mccd: Multi-agent collaboration-based compositional diffusion for complex text-to-image generation. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 13263–13272.

Mengxia Liang, Xiaolong Wang, and Shaocong Wu. 2023. Improving stock trend prediction through financial time series classification and temporal correlation analysis based on aligning change point. *Soft Computing*, 27(7):3655–3672.

Jason Lines and Anthony Bagnall. 2015. Time series classification with ensembles of elastic distance measures. *Data Mining and Knowledge Discovery*, 29(3):565–592.

Haoxin Liu, Zhiyuan Zhao, Shiduo Li, and B. Aditya Prakash. 2025. Evaluating system 1 vs. 2 reasoning approaches for zero-shot time series forecasting: A benchmark and insights. *Preprint*, arXiv:2503.01895.

Jiachang Liu, Dinghan Shen, Yizhe Zhang, Bill Dolan, Lawrence Carin, and Weizhu Chen. 2021a. What makes good in-context examples for gpt-3? *Preprint*, arXiv:2101.06804.

Jiexi Liu and Songcan Chen. 2023. Timesurl: Self-supervised contrastive learning for universal time series representation learning. *Preprint*, arXiv:2312.15709.

Minghao Liu, Shengqi Ren, Siyuan Ma, Jiahui Jiao, Yizhou Chen, Zhiguang Wang, and Wei Song. 2021b. Gated transformer networks for multivariate time series classification. *arXiv preprint arXiv:2103.14438*.

Donghao Luo and Xue Wang. 2024. Moderntcn: A modern pure convolution structure for general time series analysis. In *The twelfth international conference on learning representations*, pages 1–43.

Mike A. Merrill, Mingtian Tan, Vinayak Gupta, Tom Hartvigsen, and Tim Althoff. 2024. Language models still struggle to zero-shot reason about time series. *Preprint*, arXiv:2404.11757.

Azza Mohamed, Mohamed El Rashid, and Khaled Shaalan. 2025. In-context learning in large language models (llms): Mechanisms, capabilities, and implications for advanced knowledge representation and reasoning. *IEEE Access*, 13:95574–95593.

Yuqi Nie, Nam H. Nguyen, Phanwadee Sinthong, and Jayant Kalagnanam. 2023. A time series is worth 64 words: Long-term forecasting with transformers. *Preprint*, arXiv:2211.14730.

OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mohammad Bavarian, Jeff Belgum, and 262 others. 2024. Gpt-4 technical report. *Preprint*, arXiv:2303.08774.

Patrick Schäfer and Ulf Leser. 2018. Multivariate time series classification with weasel+muse. *Preprint*, arXiv:1711.11343.

Songyuan Sui, Hongyi Liu, Serena Liu, Li Li, Soo-Hyun Choi, Rui Chen, and Xia Hu. 2025. Chain-of-query: Unleashing the power of llms in sql-aided table understanding via multi-agent collaboration. *Preprint*, arXiv:2508.15809.

Bing Wang, Changyu Ren, Jian Yang, Xinnian Liang, Jiaqi Bai, LinZheng Chai, Zhao Yan, Qian-Wen Zhang, Di Yin, Xing Sun, and Zhoujun Li. 2025. Mac-sql: A multi-agent collaborative framework for text-to-sql. *Preprint*, arXiv:2312.11242.

Yihe Wang, Nan Huang, Taida Li, Yujun Yan, and Xiang Zhang. 2024a. Medformer: A multi-granularity patching transformer for medical time-series classification. *Preprint*, arXiv:2405.19363.

Zilong Wang, Hao Zhang, Chun-Liang Li, Julian Martin Eisenschlos, Vincent Perot, Zifeng Wang, Lesly Miculicich, Yasuhisa Fujii, Jingbo Shang, Chen-Yu Lee, and Tomas Pfister. 2024b. Chain-of-table: Evolving tables in the reasoning chain for table understanding. *Preprint*, arXiv:2401.04398.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, and 1 others. 2023. Autogen: Enabling next-gen llm applications via multi-agent conversation. *arXiv preprint arXiv:2308.08155*.

Wenxuan Xie, Gaochen Wu, and Bowen Zhou. 2024. Mag-sql: Multi-agent generative approach with soft schema linking and iterative sub-sql refinement for text-to-sql. *Preprint*, arXiv:2408.07930.

Hao Xue and Flora D. Salim. 2023. Promptcast: A new prompt-based learning paradigm for time series forecasting. *Preprint*, arXiv:2210.08964.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, and 41 others. 2025. Qwen3 technical report. *Preprint*, arXiv:2505.09388.

Jingfeng Yang, Hongye Jin, Ruixiang Tang, Xiaotian Han, Qizhang Feng, Haoming Jiang, Bing Yin, and Xia Hu. 2023. Harnessing the power of llms in practice: A survey on chatgpt and beyond. *Preprint*, arXiv:2304.13712.

Wen Ye, Wei Yang, Defu Cao, Yizhou Zhang, Lumingyuan Tang, Jie Cai, and Yan Liu. 2025. Domain-oriented time series inference agents for reasoning and automated analysis. *Preprint*, arXiv:2410.04047.

Zhihan Yue, Yujing Wang, Juanyong Duan, Tianmeng Yang, Congrui Huang, Yunhai Tong, and Bixiong Xu. 2022. Ts2vec: Towards universal representation of time series. *Preprint*, arXiv:2106.10466.

George Zerveas, Srideepika Jayaraman, Dhaval Patel, Anuradha Bhamidipaty, and Carsten Eickhoff. 2020. A transformer-based framework for multivariate time series representation learning. *Preprint*, arXiv:2010.02803.

Fuyao Zhang, Jielin Yin, Nan Wu, Xinyu Hu, Shikun Sun, and Yubao Wang. 2024. A dual-path model merging cnn and rnn with attention mechanism for crop classification. *European Journal of Agronomy*, 159:127273.

Jiahui Zhou, Dan Li, Lin Li, Zhuomin Chen, Shunyu Wu, Haozheng Ye, Jian Lou, and Costas J. Spanos. 2025a. Enhancing llm reasoning for time series classification by tailored thinking and fused decision. *Preprint*, arXiv:2506.00807.

Shu Zhou, Yunyang Xuan, Yuxuan Ao, Xin Wang, Tao Fan, and Hao Wang. 2025b. MERIT: Multi-agent collaboration for unsupervised time series representation learning. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 24011–24028, Vienna, Austria. Association for Computational Linguistics.

Tian Zhou, PeiSong Niu, Xue Wang, Liang Sun, and Rong Jin. 2023. One fits all:power general time series analysis by pretrained lm. *Preprint*, arXiv:2302.11939.

Dataset	#Classes	#Channels	Train	Test	Length
AtrialFibrillation	3	2	15	15	640
ERing	6	4	30	270	65
EigenWorms	5	6	131	128	17984
EthanolConcentration	4	3	261	263	1751
FingerMovements	2	28	316	100	50
HandMovementDirection	4	10	160	74	400
MotorImagery	2	64	278	100	3000
SelfRegulationSCP2	2	7	200	180	1152
StandWalkJump	3	4	12	15	2500

Table 3: Statistics of the nine UEA multivariate time series datasets used in our experiments.

A Dataset Details

We evaluate FETA on nine publicly available datasets from the UEA Multivariate Time Series Classification Archive (Bagnall et al., 2018). These datasets cover diverse application domains such as biomedical signals, motion capture, and sensor-based activity recognition, with wide variation in sequence length, channel dimensionality, and number of classes, making them challenging benchmarks for training-free classification. The statistics of the datasets are shown in Table 3.

All datasets are sourced from the official UEA repository[1] and are freely available for research purposes under a non-commercial academic license. Each dataset includes citation and usage instructions in accordance with the original publication. No additional data collection or annotation was performed in this work.

All datasets contain numerical sensor or signal data only, with no textual, image, or personally identifiable content. The datasets are purely scientific and contain no offensive, sensitive, or privacy-related material.

B Implementation Details

B.1 FETA Framework

All experiments are conducted using the following large language models as the backbone reasoning engines: LLaMA 3.1-8B (Grattafiori et al., 2024), DeepSeek R1 (DeepSeek-AI et al., 2025), and Qwen 3 (Yang et al., 2025). These models are used in inference-only mode without any additional training or fine-tuning. Detailed model configurations are provided in Appendix C, and the complete prompts used for each LLM agent are listed in Appendix D.

[1] https://www.timeseriesclassification.com

B.2 Tooling and Package Settings

All experiments are implemented in Python 3.10 using standard open-source libraries. Dynamic Time Warping (DTW) is implemented using the `fastdtw` library. No modifications were made to third-party packages beyond configuration and parameter settings. All experiments were executed on a Linux environment with the consistent random seed 42 to ensure reproducibility.

C LLMs' Inference Configurations

To ensure reproducibility and stability of outputs across all agents in our framework, we adopt consistent and deterministic decoding configurations for all backbone LLMs.

For `DeepSeek R1` model and `Qwen3` model, we use strictly deterministic inference: the `temperature` is set to 0.0 and `top_p` to 1.0. This disables stochastic sampling and enforces greedy decoding, ensuring identical outputs across repeated runs. All responses are generated through the official APIs provided by DeepSeek and Alibaba Cloud, respectively. Reported results are obtained from a single deterministic run without sampling variability.

For `LLaMA 3.1-8B`, we use the instruct variant deployed via Hugging Face Inference Endpoints hosted on 8×A100 GPUs. Due to platform constraints, exact deterministic settings (`temperature`=0.0, `top_p`=1.0) are not supported. As a practical approximation, we set `temperature`=0.01 and `top_p`=0.99 for all agents, which yields nearly deterministic outputs while maintaining compatibility with the inference backend. All reported LLaMA results are likewise based on a single inference pass.

D Prompts of FETA

Our prompts are uniform across datasets and not domain-specific.

[Instruction]
1. Compare the unlabeled sample ONLY with the retrieved examples shown above.
2. Focus on similarity in shape, spikes, oscillations, and recovery patterns. Ignore absolute value scale unless it clearly distinguishes classes.
3. If the majority of the retrieved examples have the same label, prioritize that label unless the sample strongly matches another.
4. Assign confidence based on neighbor consistency: All neighbors same label 0.9; 2/3 neighbors same label 0.7; Neighbors mixed evenly 0.5
5. Return ONLY one JSON object with EXACTLY these keys and no extra text.

[Response format]
{ "decision": <one of classes>,
"confidence": <0.0 to 1.0>,
"reasoning": "<one short sentence>"
}