Repurposing Foundation Model for Generalizable Medical Time Series Classification

Nan Huang
University of North Carolina at Charlotte
NC, US
nhuang5@uncc.edu

Haishuai Wang
Zhejiang University
Zhejiang, China
haishuai.wang@zju.edu.cn

Zihuai He
Stanford University
Stanford, CA, US
zihuai@stanford.edu

Marinka Zitnik
Harvard University
Boston, MA, US
marinka@hms.harvard.edu

Xiang Zhang
University of North Carolina at Charlotte
NC, US
xiang.zhang@charlotte.edu

Abstract

Medical time series (MedTS) classification suffers from poor generalizability in real-world deployment due to inter- and intra-dataset heterogeneity, such as varying numbers of channels, signal lengths, task definitions, and patient characteristics. To address this, we propose FORMED, a novel framework for repurposing a backbone foundation model, pre-trained on generic time series, to enable **highly generalizable** MedTS classification on unseen datasets. FORMED combines the backbone with a novel classifier comprising two components: (1) task-specific channel embeddings and label queries, dynamically sized to **match any number of channels and target classes**, and (2) a shared decoding attention layer, jointly trained across datasets to **capture medical domain knowledge** through task-agnostic feature-query interactions. After repurposing, FORMED achieves seamless adaptation to unseen MedTS datasets through lightweight label query training (0.1% of parameters), eliminating the need for full fine-tuning or architectural redesign. We evaluate FORMED on 5 diverse MedTS datasets, benchmarking against 11 Task-Specific Models (TSM) and 4 Task-Specific Adaptation (TSA) methods. Our results demonstrate FORMED's dominant performance, achieving up to **35% absolute** improvement in **F1**-score (on ADFTD dataset) over specialized baselines. Further analysis reveals consistent generalization across varying channel configurations, time series lengths, and clinical tasks, which are key challenges in real-world deployment. By decoupling domain-invariant representation learning from task-specific adaptation, FORMED establishes a scalable and resource-efficient paradigm for foundation model repurposing in healthcare. This approach prioritizes clinical adaptability over rigid task-centric design, offering a practical pathway for real-world implementation. *Code release upon acceptance.*

Preprint. Under review.

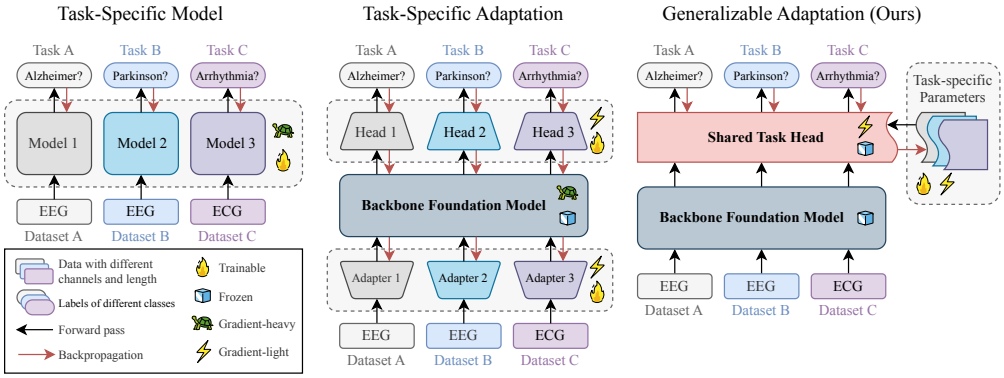

Figure 1: Paradigms of building models for different MedTS classification tasks. **Task-Specific Model (TSM):** Traditional classification models are designed for specific input shape and output classes, thus require retraining from scratch for each new dataset. **Task-Specific Adaptation (TSA):** By using a pre-trained and fixed backbone foundation models, the adaptation to new datasets requires training fewer parameters for each dataset, such as pre- and post-backbone adapters, which makes the combined model no longer applicable to other tasks, lacking generalization across tasks, and more prone to overfitting. **Generalizable Adaptation (GA):** Generalizable adaptation is a post-backbone adaptation module that is shared across tasks of different datasets, which carries domain knowledge and transferable to unseen datasets with training of lightweight task-specific parameters, offering both generalizability and robustness against overfitting.

1 Introduction

Medical time series (MedTS) classification, such as on electroencephalograms (EEG) and electrocardiograms (ECG), is critical for diagnosing a wide spectrum of medical conditions, including Alzheimer's Disease (AD; Jeong [1]), Parkinson's Disease (PD; Aljalal et al. [2, 3]), and heart Arrhythmia [4]. Despite significant advancements in developing deep learning models for these tasks, their effective generalization across diverse datasets, sometimes even among individual patients within the same dataset, remains a significant hurdle. This limitation critically obstructs the translation of advanced predictive algorithms into reliable real-world clinical applications.

Several unique challenges inherent to MedTS data compound the poor generalizability. Firstly, **inter-dataset heterogeneity** arises from variations in physiological data domains, data collection equipment, and experimental protocols, leading to differences in the *number of channels*, *sample durations*, *sampling rates*, and *diagnostic targets* [5]. Secondly, **intra-dataset heterogeneity** persists even within single datasets, with variations occurring across recording times, experimental sessions, and, most importantly, *among patients due to unique physiological characteristics* [6, 5]. This often leads to models overfitting to training data and failing to generalize to unseen patients. Thirdly, **data insufficiency** is a persistent issue: the high cost of data collection and privacy concerns often result in small MedTS datasets [7], making it difficult to train robust models capable of addressing the aforementioned heterogeneity [5].

Previous attempts to tackle these issues, such as employing Task-Specific Adaptation (TSA; see Figure 1) in models like Yang et al. [8], have shown limited success. These methods may unintentionally focus on extracting features relevant only to the initial training task, thereby failing to generalize effectively to new datasets or different medical conditions, as evidenced by marginal or even negative performance gains through pre-training. While on the other hand, recent advancements in foundation models for time series offer a promising avenue. Despite their *predominant focus on forecasting tasks* [9, 10], they demonstrate the ability to learn generic representations of time series data [11], thanks to pre-training on large-scale general time series data. This can be beneficial for MedTS classification tasks as well. Our pilot study indicates that directly adapting these models for MedTS classification is better than TSA models trained from scratch, but still fall short in capturing the intricate patterns necessary for specific diagnostic tasks when compared to state-of-the-art (SOTA) Task-Specific Models (TSMs; see Figure 1). This is primarily due to their lack of ability to capture the task-agnostic domain knowledge, which is crucial for generalization across datasets.

To address these limitations, this paper introduces FORMED (**Fo**undation model **R**epurposed for **Med**ical time series classification), a novel approach designed to repurpose foundation models for MedTS classification. FORMED aims to achieve **Generalizable Adaptation** (GA; see Figure 1).

This is achieved by utilizing a pre-trained foundation model as its backbone to capture **generic temporal features**, and integrating a novel classifier design to handle MedTS heterogeneity, by architecturally separating **medical domain knowledge** from **task-specific knowledge**. This allows the model to effectively learn and utilize both task-agnostic and task-specific features, enabling seamless handling of datasets with arbitrary channel configurations, dynamic time series lengths, and diverse diagnostic targets across multiple tasks. Therefore, FORMED enables the backbone model to effectively leverage the commonalities across datasets, while also being flexible enough to adapt to the unique characteristics of each dataset, thus achieving generalization across datasets and tasks.

To facilitate this research, we curate a comprehensive *repurposing cohort* comprising five MedTS datasets (two on ECG and three on EEG), totaling approximately 340,000 samples or 90 million time-points. These datasets exhibit diverse channel configurations (ranging from 12 to 33 channels), varied diagnostic tasks (from binary neurological to 5-class cardiovascular classification), and differing dataset sizes. This provides different levels of difficulties (both inter- and intra-dataset) for the model to learn from, and serves as a robust training and evaluation platform for the proposed method.

FORMED is strongly supported by empirical results on two aspects: First, for datasets partially included in the repurposing cohort, FORMED achieves state-of-the-art level performance on unseen patients, outperforming 15 TSA and TSM models across all datasets. Second, for completely new datasets not included in the cohort, FORMED can be efficiently adapted by updating only a small proportion of parameters while outperforming the baseline TSA model, and shows robust adapt-time scaling performance with the amount of trainable parameters. This demonstrates the model's ability to generalize across datasets and tasks, and its potential for real-world applications in healthcare.

2 Related Work

Foundation Models for General Time Series. To date, no foundation model has been specifically designed for time series classification tasks, let alone MedTS classification; instead, recent advances in time series foundation models mainly concentrate on forecasting tasks [11]. Given their success in forecasting, re-purposing these models for MedTS classification is a tempting prospect, yet it presents significant theoretical and practical challenges. These models often have major limitations, such as an inherent design for *univariate time series* in a channel-independent fashion [12], and requiring *TSAs* that prevent them from being directly applicable to MedTS classification tasks [13–15]. Given that medical time series are typically multi-variate, a critical aspect of our repurposing framework is to effectively integrate the information from multi-channel features extracted by these backbones.

For instance, models like Time-LLM [16], UniTime [17] and GPT4TS [18] are backboned on large language models. Consequently, they naturally handle time series data in a univariate manner, lacking the ability to integrate information across multiple channels crucial for MedTS classification. Moreover, consistent with the findings of Tan et al. [19], our empirical observations show that LLM-based time series foundation models do not always achieve optimal performance, even on general time series datasets. Similarly, while TimeGPT [20] and TimesFM [21] are pre-trained on large scale time series data, they typically operate under a channel-independence assumption, treating co-evolving multivariate time series data as a collection of independent univariate series. This shares the same limitation for direct application to multi-channel MedTS. Our proposed repurposing framework is specifically designed to address this by incorporating mechanisms to accommodate the multi-channel nature of MedTS data, allowing for effective integration of information across channels.

UniTS [22] stands out as capable of handling multivariate time series data and has been trained on multiple task domains including classification. However, its scale and design often necessitate fine-tuning the entire model or employing prompt learning for optimal performance. This approach is both computationally expensive (see Figure 1) and data-greedy due to the vast number of parameters to tune, rendering it less suitable for often small-scale MedTS datasets.

Despite their efforts and successes, current foundation models require significant adaptations. Thus, **a key challenge, which FORMED directly addresses, is the adaptation of these powerful but often channel-independent or forecasting-focused models to the multichannel classification demands of MedTS**. This is achieved by integrating dedicated architectural components to address the complexities and multi-channel nature of MedTS. While all the aforementioned models represent potential backbones for our repurposing framework, due to resource limitations and the primary goal

to validate the efficacy of our proposed framework, we selected the advanced TimesFM [21] as our backbone model for this study for its outstanding zero-shot forecasting performance.

Adaptation of Foundation Models for MedTS Classification.

General-purpose foundation models typically require specific techniques to be effectively adapted for downstream tasks. Common approaches include *prompting*, *fine-tuning*, *re-programming*, and our proposed **re-purposing**. Although a detailed comparison of all techniques is beyond the immediate scope (further discussion in Appendix A), we will focus here on the distinction between re-programming and re-purposing, as this differentiation is central to our proposed approach for MedTS classification.

Re-programming often involves reusing a pre-trained model's backbone (*e.g.*, its Transformer layers) without altering its internal weights, but wrapping it with new input adapters and task-specific output heads (as illustrated by the TSA approach in Figure 1). While this can adapt a model to a new data domain or task type, a significant drawback is that the resulting model often **loses its general-purpose nature**. Both the input adapters and task heads become highly specialized, and **cannot be reused** in future datasets with different configurations, hindering the model's ability to generalize across different tasks, datasets, or to handle the inherent heterogeneity within MedTS [19].

Re-purposing, as introduced in this work with our FORMED framework, takes a different philosophical approach. It aims to adapt a pre-trained foundation model (often one excelling in tasks like forecasting) to a new class of tasks—in our case, MedTS classification. This is achieved with thoughtful modifications, particularly in how task-specific knowledge is integrated, while striving to maintain the model's core learned representations. The goal is for the repurposed model to serve as a robust and generalizable tool within the target domain (MedTS classification), capable of being efficiently applied to new datasets and diagnostic challenges. Its emphasis on generalizability, data-efficiency, and leveraging domain insights makes re-purposing particularly suitable for adapting powerful, channel-independent time series foundation models to the complexities of multi-channel MedTS classification, and creating a new foundation model for the field.

Forecasting versus Classification in Time Series. The fundamental differences between time series forecasting and classification are key to understanding the challenges in adapting existing foundation models. **Forecasting** typically involves predicting future sequence values within the same domain as the input (a sequence $\rightarrow$ sequence mapping) [23, 24], often by extrapolating patterns **within individual channels**. In contrast, **classification** maps an input sequence to a distinct categorical label (a sequence $\rightarrow$ category mapping) [25]. This frequently requires synthesizing complex patterns across **multiple interacting channels** to derive a diagnostic outcome (*e.g.*, diagnosing disease from multi-channel EEG; [26])—a process inherently different from predicting future signal values. This intrinsic divergence in objectives and the nature of data interpretation means that adapting a forecasting foundation model for MedTS classification is **NOT a simple modification of the output layer**. It demands a more comprehensive re-purposing strategy, such as our FORMED framework, designed to bridge these task-specific requirements and effectively handle the complexities of multivariate medical data.

3 Problem Statement

Foundation models, pre-trained on diverse forecasting tasks, have demonstrated a strong capability to capture general time series patterns. However, transforming these into general-purpose classification models, especially for MedTS, is non-trivial. This section formally defines the problem and the core concepts underpinning our proposed two-stage adaptation process: repurposing and adapting.

Definition 3.1. Repurposing: The process of changing the objective of a pre-trained foundation model to a new class of tasks for which it was not originally trained. This involves introducing and training a relatively small, adaptable output network while keeping the pre-trained backbone fixed.

Let the original pre-trained model consists of a backbone $f : \mathbb{R}^T \rightarrow \mathbb{R}^{L \times D}$ for extracting features from a univariate time series of length T into L patched tokens of dimension D, and an original task head (*e.g.*, for forecasting, $g : \mathbb{R}^{L \times D} \rightarrow \mathbb{R}^N$). We leverage the frozen backbone f for representation learning. For multivariate MedTS input $\boldsymbol{X} \in \mathbb{R}^{C \times T}$ with C channels, the backbone is applied

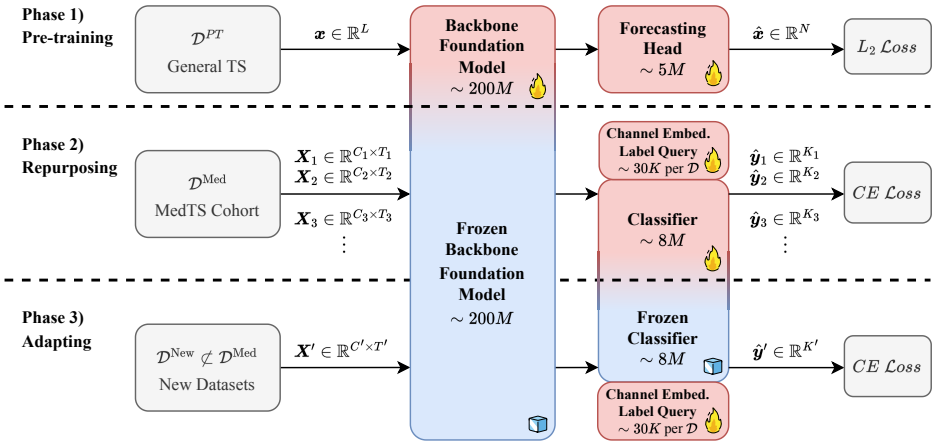

Figure 2: The three-stage process of adapting a time series foundation model for MedTS classification tasks. 1) **Pre-training** is already done on diverse general time series datasets with forecasting tasks. 2) **Repurposing** the foundation model involves changing the forecasting head to a classification head, while keeping the rest of the model fixed, and the new model is then trained on a cohort of MedTS datasets to capture domain knowledge in MedTS. 3) **Adapting** the repurposed model to the new MedTS datasets, where only the minimal task-specific parameters are trained, leveraging the previously learned domain knowledge from the repurposed model.

channel-wise to extract features:

$$\begin{aligned} &\mathbf{f} : \mathbb{R}^{C \times T} \to \mathbb{R}^{C \times L \times D} \\ \Leftrightarrow\ &\mathbf{f}(\boldsymbol{X}) := [f(\boldsymbol{X}_{c,:})]_{c=1}^{C} \end{aligned} \tag{1}$$

We then introduce a novel, trainable classification head h_θ. This head is designed to be adaptable to specific task characteristics, such as the number of input channels C and the number of output classes K, through learnable task-specific parameters: **Channel Embedding** $\boldsymbol{E} \in \mathbb{R}^{C \times D}$ and **Label Queries** $\boldsymbol{Q} \in \mathbb{R}^{K \times D}$. The mapping becomes:

$$\begin{aligned} h_\theta|_{\boldsymbol{Q},\boldsymbol{E}} :&\mathbb{R}^{C \times L \times D} \to \Delta^K \\ \Rightarrow\ (h_\theta \circ \mathbf{f})|_{\boldsymbol{Q},\boldsymbol{E}} :&\mathbb{R}^{C \times T} \to \Delta^K \end{aligned} \tag{2}$$

where $\Delta^K = \left\{ \boldsymbol{d} \in [0,1]^K : \sum_{i=1}^{K} d_i = 1 \right\}$ is the probability simplex for K classes.

During the **repurposing stage**, h_θ containing shared parameter θ along with collections of task-specific embeddings $\mathbf{E} = \{\boldsymbol{E}_i\}$ and $\mathbf{Q} = \{\boldsymbol{Q}_i\}$ are trained across a cohort of diverse MedTS datasets $\mathcal{D}^{\text{Med}} = \left\{\mathcal{D}_i^{\text{Med}}\right\}$. The objective is to learn domain knowledge for MedTS classification within θ. This translates to minimizing the cross-entropy loss $\mathcal{L}_{\text{CE}}$:

$$\theta^*, \mathbf{E}^*, \mathbf{Q}^* = \arg\min_{\theta,\mathbf{E},\mathbf{Q}} \mathbb{E}_{i,(\boldsymbol{X}_i,\boldsymbol{y}_i)\in\mathcal{D}_i^{\text{Med}}} \left[\mathcal{L}_{\text{CE}} \left(h_\theta|_{\boldsymbol{Q}_i,\boldsymbol{E}_i} \left(\mathbf{f}(\boldsymbol{X}_i) \right), \boldsymbol{y}_i \right) \right] \tag{3}$$

Definition 3.2. Adapting: The process of applying the repurposed model (with fixed θ^* and frozen $\mathbf{f}$) to new, unseen MedTS datasets or tasks. This involves learning only a minimal set of new task-specific parameters (new $\boldsymbol{E}'$ and $\boldsymbol{Q}'$) for the new dataset.

For a new dataset $\mathcal{D}^{\text{New}}$ (with potentially different C' channels, T' time steps, and K' classes), the pre-trained backbone $\mathbf{f}$ and the shared parameters θ^* of the classifier h_{θ^*} remain frozen. Only new Channel Embeddings $\boldsymbol{E}' \in \mathbb{R}^{C' \times D}$ and Label Queries $\boldsymbol{Q}' \in \mathbb{R}^{K' \times D}$ are initialized and trained:

$$\boldsymbol{E}'^*, \boldsymbol{Q}'^* = \arg\min_{\boldsymbol{E}',\boldsymbol{Q}'} \mathbb{E}_{(\boldsymbol{X}',\boldsymbol{y}')\in\mathcal{D}^{\text{New}}} \left[\mathcal{L} \left(h_{\theta^*}|_{\boldsymbol{Q}',\boldsymbol{E}'} \left(\mathbf{f}(\boldsymbol{X}') \right), \boldsymbol{y}' \right) \right] \tag{4}$$

This two-stage process (illustrated in Figure 2) allows the model to learn general MedTS domain knowledge and efficiently specialize to new tasks with minimal new data and computation.

4 Model Architecture

Our FORMED framework repurposes a pre-trained time series foundation model to serve as a versatile MedTS classification tool. It comprises two main parts: a frozen backbone feature extractor and our novel attention-based classifier designed for adaptability and generalizability.

4.1 Backbone Feature Extractor for Variable Length Time Series

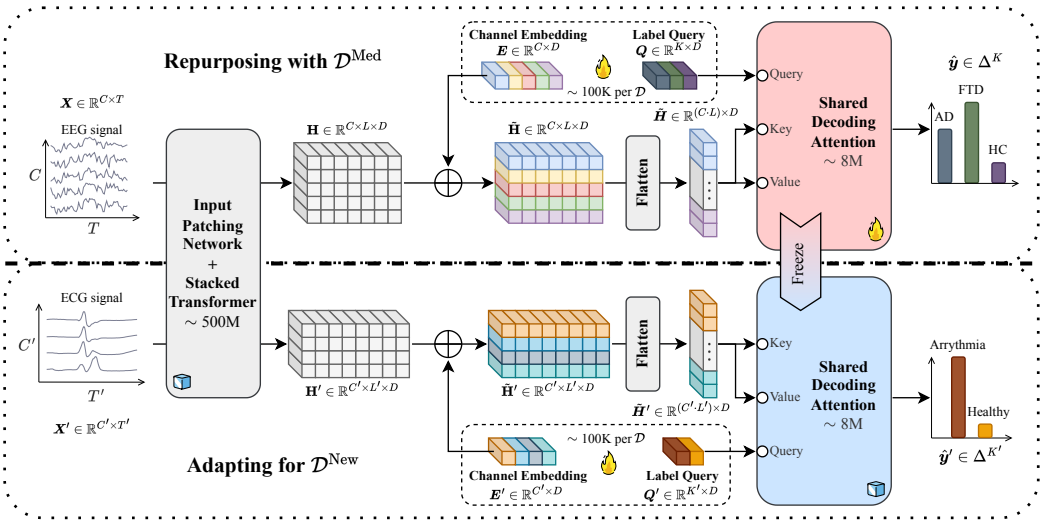

Figure 3: The architecture of the proposed model in repurposing and adapting. The backbone foundation model acts as a feature extractor and remains frozen all the time. The **Channel Embeddings** (CEs) and **Label Queries** (LQs) are task-specific parameters that are learned during both repurposing and adapting, and new ones will be created and learned if encountering new datasets. The **Shared Decoding Attention** (SDA) is a shared Transformer decoder layer that captures the interaction between all the features and classes, which once get trained on curated MedTS datasets $\mathcal{D}^{\text{Med}}$ during repurposing, will be fixed and reused when adapting to all future datasets and tasks $\mathcal{D}^{\text{New}}$. The $\oplus$ denotes broadcast addition.

We employ a powerful pre-trained time series foundation model as the backbone feature extractor. For this work, we selected TimesFM [21] due to its demonstrated ability to capture complex temporal patterns from variable-length time series, pre-trained on a vast and diverse corpus of general time series data. The backbone's primary role is to transform an input univariate time series $\boldsymbol{x} \in \mathbb{R}^T$ into a sequence of rich feature tokens $\boldsymbol{H} \in \mathbb{R}^{L \times D}$. As defined in Equation (1), for multi-channel MedTS data $\boldsymbol{X} \in \mathbb{R}^{C \times T}$, this backbone is applied independently to each channel, yielding stacked feature tokens $\mathbf{H} \in \mathbb{R}^{C \times L \times D}$. The internal architecture of TimesFM (*e.g.*, its patching mechanism and Transformer layers) is kept frozen throughout our framework and detailed in Appendix B. The crucial output for our purpose is $\mathbf{H}$, which serves as the input to our novel classifier.

4.2 Attention-based Classifier for Varying Channel and Class

Traditional approaches often use a simple linear layer or convolution layer for classification [27, 8], which can be restrictive when dealing with the inherent variability of MedTS data (*e.g.*, varying numbers of channels, classes, and sequence lengths). To address these unique challenges, we propose a novel attention-based classifier built upon a Transformer decoder layer [28], inspired by advancements in vision tasks [29, 30] but with key modifications tailored for MedTS. This classifier comprises three main components: Channel Embeddings (CEs), Label Queries (LQs), and a Shared Decoding Attention (SDA) mechanism.

Channel Embeddings (CEs). MedTS are inherently multi-variate. To enable the model to distinguish information from different physiological channels and understand their task-specific relevance, we introduce learnable Channel Embeddings $\boldsymbol{E} \in \mathbb{R}^{C \times D}$. For a given dataset with C channels, these embeddings are broadcast-added to the corresponding channel-wise feature tokens $\mathbf{H}$ (from

Equation (1)) to produce "channel-aware" feature tokens $\tilde{\mathbf{H}} \in \mathbb{R}^{C \times L \times D}$:

$$\tilde{\mathbf{H}}c, l, : = \mathbf{H}c, l, : \oplus \boldsymbol{E}_{c,:} \quad \forall l \in \{1, 2, \cdots, L\}, c \in \{1, 2, \cdots, C\} \tag{5}$$

These CEs are task-specific and learned during both repurposing (part of $\mathbf{E}$ in Equation (3)) and adapting ($\boldsymbol{E}'$ in Equation (4)).

Label Queries (LQs). To handle varying numbers of diagnostic classes (K) per task and provide distinct learnable "anchors" for each class, we use Label Queries $\boldsymbol{Q} \in \mathbb{R}^{K \times D}$. Each row $\boldsymbol{Q}_{i,:}$ is a learnable embedding representing the i-th class. These queries actively seek evidence for their respective classes within the channel-aware feature tokens. Like CEs, LQs are task-specific and learned during repurposing and adapting. In practice, we employ k learnable queries for each class to capture potentially complex or multiple defining patterns, where k is a hyperparameter determining the number of distinct "perspectives" or "sub-pattern detectors" for each class. This results in a total of $K \cdot k$ queries in $\boldsymbol{Q} \in \mathbb{R}^{(K \cdot k) \times D}$. Each group of k queries corresponding to a specific class independently attends to the feature tokens to gather evidence.

Shared Decoding Attention (SDA). The core of our classifier is the SDA mechanism, a single Transformer decoder layer whose parameters (θ in Equation (3)) are shared across all datasets in the repurposing cohort and then frozen during adaptation to new tasks. The SDA takes all $K \cdot k$ Label Queries from $\boldsymbol{Q}$ as queries, and the flattened, channel-aware feature tokens $\texttt{Flatten}(\tilde{\mathbf{H}}) \in \mathbb{R}^{(C \cdot L) \times D}$ as keys and values. It performs multi-head attention followed by an `FeedForwardNetwork` to produce an initial set of logits $\hat{\boldsymbol{y}}_{\text{raw}} \in \mathbb{R}^{K \cdot k}$:

$$\hat{\boldsymbol{y}}_{\text{raw}} = \texttt{FeedForwardNetwork}\left(\texttt{MultiHeadAttention}(\boldsymbol{Q}, \texttt{Flatten}(\tilde{\mathbf{H}}), \texttt{Flatten}(\tilde{\mathbf{H}}))\right) \tag{6}$$

Since we have k queries (and thus k raw logits) for each of the K classes, these k logits are then averaged to produce a single, final logit for each class, resulting in the final class logits $\hat{\boldsymbol{y}} \in \mathbb{R}^K$:

$$\hat{\boldsymbol{y}}_j = \frac{1}{k} \sum_{i=1}^{k} (\hat{\boldsymbol{y}}_{\text{raw}})_{(j-1)k+i} \quad \forall j \in \{1, 2, \cdots, K\} \tag{7}$$

These final logits $\hat{\boldsymbol{y}}$ are then used with a `softmax` function for probability prediction and loss calculation. Critically, the parameters within the `MultiHeadAttention` and the `FeedForwardNetwork` (collectively θ) are independent of the specific number of input channels C, token length L, or the total number of queries $K \cdot k$. This architectural choice forces SDA to learn generalizable interaction patterns while allowing for richer, more nuanced class representations.

5 Experiments

Here we describe the experimental setup, including the datasets chosen to evaluate FORMED against key MedTS challenges, the baselines selected to highlight the advantages of our repurposing framework, and evaluation metrics. Additional training details are included in Appendix C.

Datasets. To thoroughly evaluate FORMED, we curated a **MedTS cohort** of 5 diverse datasets for the crucial repurposing stage (Figure 2). This cohort, comprising two ECG datasets and three EEG datasets (details in Table 3), provides a breadth of configurations and task types. This enables the **learning of robust, generalizable MedTS domain knowledge** during repurposing. Critically, these datasets span a variety of combinations of channels, sampling rates, sample durations, diagnostic labels, and overall sizes. This inherent diversity is intentional, enabling us to directly **assess FORMED's ability to handle inter-dataset heterogeneity** during repurposing.

Furthermore, all datasets within the cohort and for subsequent adaptation are split following strict patient-independent settings [6, 33]. This ensures that the test sets contain subjects entirely unseen during training, rigorously **evaluating the model's robustness against intra-dataset heterogeneity** and its capacity to generate to new patients rather than memorizing subject-specific features.

To specifically test the adapting stage and FORMED's generalization to entirely new, unseen tasks and potentially different data characteristics, we also include out-of-domain datasets (ECG200 [31] and StandWalkJump [32]). Performance on these datasets, especially with **limited data for adaptation**, will demonstrate FORMED's **utility under data insufficiency**.

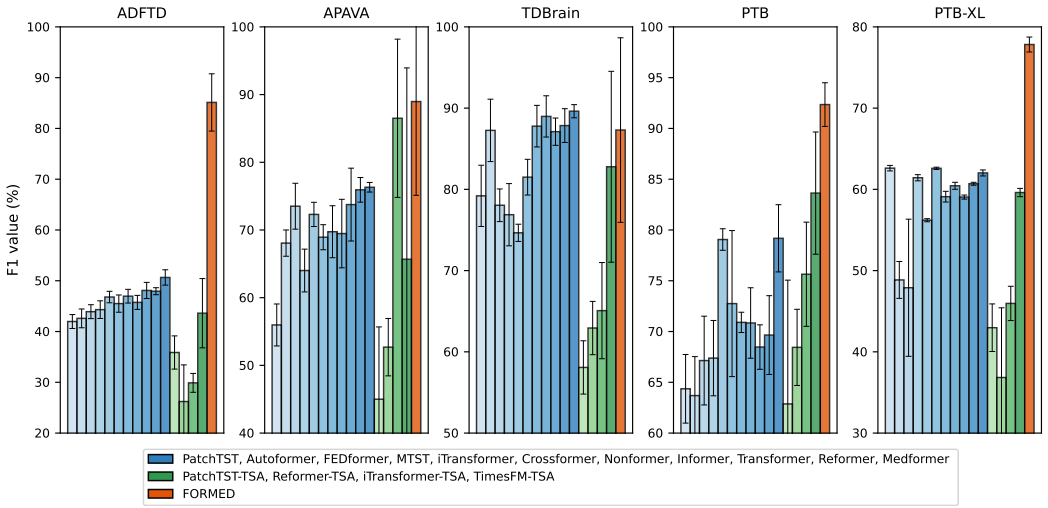

Figure 4: In-domain F1 performance on the MedTS cohort datasets. FORMED achieves SOTA level performance across all datasets in all metrics. Numerical results are shown in Table 4. Other metrics are included in Appendix E.

Baselines. We compare FORMED with 15 SOTA baselines including 11 TSM and 4 TSA models.

The TSM models—Autoformer [35], Crossformer [36], FEDformer [37], Informer [38], iTransformer [39], MTST [40], Nonformer [41], PatchTST [12], Reformer [42], Transformer [28] and Medformer [6]—are trained independently on each dataset. They represent the conventional approach, used to verify the benefits of FORMED's repurposing stage in learning transferable domain knowledge.

The TSA models aim to share knowledge across tasks. *TimesFM-TSA* is a key baseline, created by adapting the same TimesFM [21] backbone with a task-specific CNN head for classification. This allows for a direct evaluation of the additional benefits provided by FORMED's novel classifier design and the two-stage repurposing/adapting strategy over a more straightforward adaptation of the same foundation model. The additional TSA models, *PatchTST-TSA*, *Reformer-TSA*, and *iTransformer-TSA*, are variants of their TSM counterparts. Their backbones are shared across tasks with task-specific heads, and they are configured to have a similar parameter amount to FORMED for demonstrating the superiority of using pre-trained foudnation models over training from scratch.

Evaluations. The effectiveness of our method is primarily demonstrated through its classification performance on the test sets, using metrics such as accuracy, precision, recall, F1 score, AUROC, and AUPRC. Strong performance on these patient-independent test sets will underscore FORMED's ability to overcome intra-dataset heterogeneity. The generalization ability to unseen tasks, particularly under conditions of potential data insufficiency and inter-dataset heterogeneity, is evaluated through adapting experiments on the small, out-of-domain datasets. These experiments, crucial for assessing the efficacy of the adapting stage where only minimal parameters (CEs and LQs) are learned, are conducted on five random seeds for all models, with results averaged to ensure reliability.

5.1 Evaluation on Repurposing: Generalize to Unseen Subjects

Setup. For repurposing datasets in MedTS cohort, we trained 55 TSM models (11 models for each), and 4 TSA models with 5 task-specific heads each. Our FORMED model is trained on all 5 datasets, using a fixed $k = 16$ for all datasets.

Results. The results compellingly demonstrate that the proposed repurposing, which allows the SDA to capture shared MedTS domain knowledge while CEs and LQs handle task-specifics, is more effective than both TSM and TSA approaches for complex classification tasks. As shown in Figure 4, FORMED achieves significant improvements over all baselines. This is particularly pronounced on medium-to-large datasets, such as ADFTD, PTB, and PTB-XL, where performance gains up to 30-40%. On the relatively small TDBrain dataset, which represents a simpler task where TSMs' performance also saturate, FORMED performs on par with the strongest TSMs while maintaining a clear advantage over TSA methods. This superiority stems from FORMED's ability to learn from the collective knowledge of the diverse MedTS cohort, demonstrating that SDA learns generalizable

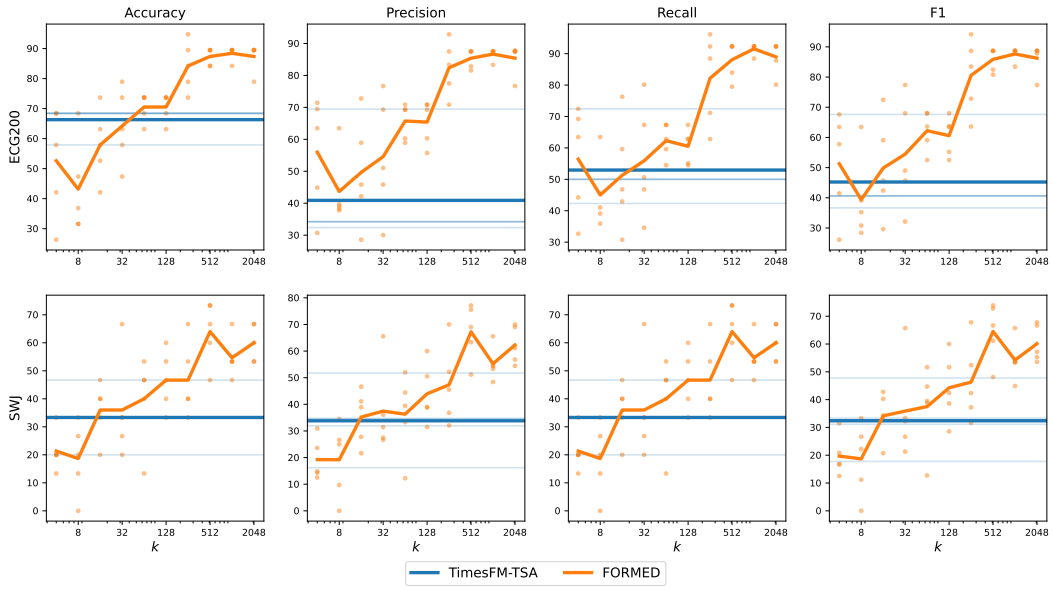

Figure 5: Adapt-time scaling on unseen, out-of-domain dataset. FORMED's performance scales well with k following power law, and outperforms TimesFM-TSA starting from $k = 64$ on ECG200, and from $k = 16$ on StandWalkJump. Numerical results see Table 5.

patterns relevant to MedTS classification. Such robust learning across varied datasets translates directly into **addressing intra-dataset heterogeneity**, *i.e.*, better generalization to unseen subjects.

Interestingly, TSA models generally underperform compared to TSMs, and significantly to FORMED. This might suggest that their simpler task-specific heads on a shared backbone are less effective at navigating the substantial inter-dataset heterogeneity within the MedTS cohort than dedicated SDA, CE and LQ architecture. TimesFM-TSA, with the same backbone as FORMED, provides the clearest comparison to this hypothesis.

5.2 Evaluation on Adapting: Generalize to Unseen Task

Setup. In this evaluation, we assess FORMED's ability to generalize to entirely new, unseen tasks, a critical test of its adapting stage and its robustness to **inter-dataset heterogeneity** and **data insufficiency**. We use the TimesFM-TSA model as a strong baseline. The FORMED model is obtained from the repurposing stage; its backbone and SDA parameters are kept frozen. Only newly initialized CEs and LQs are trained for these new tasks, demonstrating **data-efficient adaptation**. We also explore how performance scales by varying the number of queries per class (k) from 4 to 2048.

Results. The TimesFM-TSA baseline, lacking a sophisticated pre-adaptation to the MedTS domain for its classification components, can struggle to generalize from limited data on new tasks, often overfitting to the training data. In contrast, FORMED, by leveraging the rich MedTS domain knowledge captured in its frozen SDA during the repurposing stage, demonstrates superior adaptation. As seen in Figure 5, FORMED generally outperforms the TimesFM-TSA baseline on these unseen datasets, even with less adaptable parameters (FORMED at $k = 16$ outperforms TimesFM-TSA with only $\sim$ 1/6 of the parameter). The performance of FORMED steadily improves with an increase in k. This shows that FORMED can effectively utilize increased capacity within its minimal adaptable components (CEs and LQs) to better handle new tasks without requiring extensive retraining or large datasets. This scalability and efficiency in the adapting stage validates FORMED's **strength against data insufficiency and inter-dataset heterogeneity**, making it particularly well-suited for real-world clinical applications where new diagnostic tasks may emerge with **limited available data**.

6 Discussions and Conclusion

In this paper, we introduced **FORMED**, a novel framework that repurposes general time series foundation models for robust and adaptable MedTS classification. FORMED's core architectural

innovation lies in its attention-based classifier, featuring SDA, CEs, and LQs. This design uniquely equips models to **handle variable input lengths, diverse channel configurations, and dynamic numbers of output classes**—addressing limitations of conventional TSM and TSA approaches.

Our comprehensive experiments demonstrate FORMED's strong generalization capabilities, achieving **state-of-the-art performance for unseen patients within datasets (intra-dataset) and effectively adapting to unseen tasks (inter-dataset).** This highlights two significant findings: first, the feasibility and effectiveness of leveraging powerful pre-trained foundation models as backbones for complex MedTS classification. Second, it validates the superiority of our FORMED repurposing framework. The framework excels at capturing transferable MedTS domain knowledge within its shared components during an initial repurposing stage, which then **enables highly efficient and data-scarce adaptation to new tasks by learning only minimal, task-specific parameters.**

Despite these promising results, we acknowledge limitations. Our validation primarily utilized TimesFM as the backbone. While this serves as a strong proof-of-concept for the FORMED framework's efficacy, its performance and interaction with other diverse time series foundation models warrant future investigation. Additionally, the composition and scale of the MedTS cohort employed during the repurposing stage may also influence the breadth of the captured domain knowledge.

In conclusion, FORMED presents a significant step towards more **generalizable**, **adaptable**, and **data-efficient** deep learning solutions for the unique challenges posed by medical time series analysis, offering a robust framework for future advancements in the field.

References

[1] Jaeseung Jeong. EEG dynamics in patients with Alzheimer's disease. *Clinical Neurophysiology*, 115(7):1490–1505, July 2004. ISSN 1388-2457. doi: 10.1016/j.clinph.2004.01.001.

[2] Majid Aljalal, Saeed A. Aldosari, Marta Molinas, Khalil AlSharabi, and Fahd A. Alturki. Detection of Parkinson's disease from EEG signals using discrete wavelet transform, different entropy measures, and machine learning techniques. *Scientific Reports*, 12(1):22547, December 2022. ISSN 2045-2322. doi: 10.1038/s41598-022-26644-7.

[3] Majid Aljalal, Saeed A. Aldosari, Khalil AlSharabi, Akram M. Abdurraqeeb, and Fahd A. Alturki. Parkinson's Disease Detection from Resting-State EEG Signals Using Common Spatial Pattern, Entropy, and Machine Learning Techniques. *Diagnostics*, 12(5):1033, April 2022. ISSN 2075-4418. doi: 10.3390/diagnostics12051033.

[4] Yanrui Jin, Zhiyuan Li, Mengxiao Wang, Jinlei Liu, Yuanyuan Tian, Yunqing Liu, Xiaoyang Wei, Liqun Zhao, and Chengliang Liu. Cardiologist-level interpretable knowledge-fused deep neural network for automatic arrhythmia diagnosis. *Communications Medicine*, 4(1):1–8, February 2024. ISSN 2730-664X. doi: 10.1038/s43856-024-00464-4.

[5] Nagarajan Ganapathy, Ramakrishnan Swaminathan, and Thomas M. Deserno. Deep Learning on 1-D Biosignals: a Taxonomy-based Survey. *Yearbook of Medical Informatics*, 27(1):98–109, August 2018. ISSN 0943-4747. doi: 10.1055/s-0038-1667083.

[6] Yihe Wang, Nan Huang, Taida Li, Yujun Yan, and Xiang Zhang. Medformer: A Multi-Granularity Patching Transformer for Medical Time-Series Classification, May 2024.

[7] Shruti Kaushik, Abhinav Choudhury, Pankaj Kumar Sheron, Nataraj Dasgupta, Sayee Natarajan, Larry A. Pickett, and Varun Dutt. AI in Healthcare: Time-Series Forecasting Using Statistical, Neural, and Ensemble Architectures. *Frontiers in Big Data*, 3, March 2020. ISSN 2624-909X. doi: 10.3389/fdata.2020.00004.

[8] Chaoqi Yang, M. Westover, and Jimeng Sun. BIOT: Biosignal Transformer for Cross-data Learning in the Wild. *Advances in Neural Information Processing Systems*, 36:78240–78260, December 2023.

[9] Jiexia Ye, Weiqi Zhang, Ke Yi, Yongzi Yu, Ziyue Li, Jia Li, and Fugee Tsung. A Survey of Time Series Foundation Models: Generalizing Time Series Representation with Large Language Model, May 2024.

[10] Qingsong Wen, Tian Zhou, Chaoli Zhang, Weiqi Chen, Ziqing Ma, Junchi Yan, and Liang Sun. Transformers in Time Series: A Survey, February 2022.

[11] Yuxuan Liang, Haomin Wen, Yuqi Nie, Yushan Jiang, Ming Jin, Dongjin Song, Shirui Pan, and Qingsong Wen. Foundation Models for Time Series Analysis: A Tutorial and Survey. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, August 2024. doi: 10.1145/3637528.3671451.

[12] Yuqi Nie, Nam H. Nguyen, Phanwadee Sinthong, and Jayant Kalagnanam. A Time Series is Worth 64 Words: Long-term Forecasting with Transformers, November 2022.

[13] Defu Cao, Furong Jia, Sercan O. Arik, Tomas Pfister, Yixiang Zheng, Wen Ye, and Yan Liu. TEMPO: Prompt-based Generative Pre-trained Transformer for Time Series Forecasting, April 2024.

[14] Chenxi Sun, Hongyan Li, Yaliang Li, and Shenda Hong. TEST: Text Prototype Aligned Embedding to Activate LLM's Ability for Time Series, February 2024.

[15] Ching Chang, Wen-Chih Peng, and Tien-Fu Chen. LLM4TS: Two-Stage Fine-Tuning for Time-Series Forecasting with Pre-Trained LLMs, September 2023.

[16] Ming Jin, Shiyu Wang, Lintao Ma, Zhixuan Chu, James Y. Zhang, Xiaoming Shi, Pin-Yu Chen, Yuxuan Liang, Yuan-Fang Li, Shirui Pan, and Qingsong Wen. Time-LLM: Time Series Forecasting by Reprogramming Large Language Models, January 2024.

[17] Xu Liu, Junfeng Hu, Yuan Li, Shizhe Diao, Yuxuan Liang, Bryan Hooi, and Roger Zimmermann. UniTime: A Language-Empowered Unified Model for Cross-Domain Time Series Forecasting, February 2024.

[18] Tian Zhou, PeiSong Niu, Xue Wang, Liang Sun, and Rong Jin. One Fits All:Power General Time Series Analysis by Pretrained LM, May 2023.

[19] Mingtian Tan, Mike A. Merrill, Vinayak Gupta, Tim Althoff, and Thomas Hartvigsen. Are Language Models Actually Useful for Time Series Forecasting?, June 2024.

[20] Azul Garza, Cristian Challu, and Max Mergenthaler-Canseco. TimeGPT-1, May 2024.

[21] Abhimanyu Das, Weihao Kong, Rajat Sen, and Yichen Zhou. A decoder-only foundation model for time-series forecasting, April 2024.

[22] Shanghua Gao, Teddy Koker, Owen Queen, Thomas Hartvigsen, Theodoros Tsiligkaridis, and Marinka Zitnik. UNITS: A Unified Multi-Task Time Series Model, May 2024.

[23] Bryan Lim and Stefan Zohren. Time-series forecasting with deep learning: a survey. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 379(2194):20200209, February 2021. doi: 10.1098/rsta.2020.0209.

[24] Bingxin Wang, Xiaowen Fu, Yuan Lan, Luchan Zhang, Wei Zheng, and Yang Xiang. Large Transformers are Better EEG Learners, April 2024.

[25] Mohammed Ali, Ali Alqahtani, Mark W. Jones, and Xianghua Xie. Clustering and Classification for Time Series Data in Visual Analytics: A Survey. *IEEE Access*, 7:181314–181338, 2019. ISSN 2169-3536. doi: 10.1109/ACCESS.2019.2958551.

[26] Zhijiang Wan, Manyu Li, Shichang Liu, Jiajin Huang, Hai Tan, and Wenfeng Duan. EEGformer: A transformer–based brain activity classification method using EEG signal. *Frontiers in Neuroscience*, 17:1148855, March 2023. ISSN 1662-4548. doi: 10.3389/fnins.2023.1148855.

[27] George Zerveas, Srideepika Jayaraman, Dhaval Patel, Anuradha Bhamidipaty, and Carsten Eickhoff. A Transformer-based Framework for Multivariate Time Series Representation Learning. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, KDD '21, pages 2114–2124, New York, NY, USA, August 2021. Association for Computing Machinery. ISBN 978-1-4503-8332-5. doi: 10.1145/3447548.3467401.

[28] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is All you Need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[29] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-End Object Detection with Transformers, May 2020.

[30] Xiaoxia Meng, Xiaowei Wang, Shoulin Yin, and Hang Li. Few-shot image classification algorithm based on attention mechanism and weight fusion. *Journal of Engineering and Applied Science*, 70(1):14, March 2023. ISSN 2536-9512. doi: 10.1186/s44147-023-00186-9.

[31] Robert Thomas Olszewski. Generalized feature extraction for structural pattern recognition in time-series data.

[32] Vahid Behravan, Neil E. Glover, Rutger Farry, and Patrick Y. Chiang. Motion Artifact Contaminated ECG Database.

[33] Yihe Wang, Taida Li, Yujun Yan, Wenzhan Song, and Xiang Zhang. How to evaluate your medical time series classification?

[34] Yihe Wang, Yu Han, Haishuai Wang, and Xiang Zhang. Contrast Everything: A Hierarchical Contrastive Framework for Medical Time-Series. *Advances in Neural Information Processing Systems*, 36:55694–55717, December 2023.

[35] Haixu Wu, Jiehui Xu, Jianmin Wang, and Mingsheng Long. Autoformer: Decomposition Transformers with Auto-Correlation for Long-Term Series Forecasting. In *Advances in Neural Information Processing Systems*, volume 34, pages 22419–22430. Curran Associates, Inc., 2021.

[36] Yunhao Zhang and Junchi Yan. Crossformer: Transformer Utilizing Cross-Dimension Dependency for Multivariate Time Series Forecasting. September 2022.

[37] Tian Zhou, Ziqing Ma, Qingsong Wen, Xue Wang, Liang Sun, and Rong Jin. FEDformer: Frequency Enhanced Decomposed Transformer for Long-term Series Forecasting. In *Proceedings of the 39th International Conference on Machine Learning*, pages 27268–27286. PMLR, June 2022.

[38] Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang. Informer: Beyond Efficient Transformer for Long Sequence Time-Series Forecasting. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(12):11106–11115, May 2021. ISSN 2374-3468. doi: 10.1609/aaai.v35i12.17325.

[39] Yong Liu, Tengge Hu, Haoran Zhang, Haixu Wu, Shiyu Wang, Lintao Ma, and Mingsheng Long. iTransformer: Inverted Transformers Are Effective for Time Series Forecasting, October 2023.

[40] Yitian Zhang, Liheng Ma, Soumyasundar Pal, Yingxue Zhang, and Mark Coates. Multi-resolution Time-Series Transformer for Long-term Forecasting. In *Proceedings of The 27th International Conference on Artificial Intelligence and Statistics*, pages 4222–4230. PMLR, April 2024.

[41] Yong Liu, Haixu Wu, Jianmin Wang, and Mingsheng Long. Non-stationary Transformers: Exploring the Stationarity in Time Series Forecasting. *Advances in Neural Information Processing Systems*, 35:9881–9893, December 2022.

[42] Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. Reformer: The Efficient Transformer, January 2020.

[43] Margaret Mitchell, Simone Wu, Andrew Zaldivar, Parker Barnes, Lucy Vasserman, Ben Hutchinson, Elena Spitzer, Inioluwa Deborah Raji, and Timnit Gebru. Model Cards for Model Reporting. In *Proceedings of the Conference on Fairness, Accountability, and Transparency*, FAT* '19, pages 220–229, New York, NY, USA, January 2019. Association for Computing Machinery. ISBN 978-1-4503-6125-5. doi: 10.1145/3287560.3287596.

[44] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-Rank Adaptation of Large Language Models, October 2021.

[45] Song Wang, Yaochen Zhu, Haochen Liu, Zaiyi Zheng, Chen Chen, and Jundong Li. Knowledge Editing for Large Language Models: A Survey, September 2024.

[46] Hila Chefer, Shir Gur, and Lior Wolf. Transformer Interpretability Beyond Attention Visualization. pages 782–791, 2021.

[47] Yaru Hao, Li Dong, Furu Wei, and Ke Xu. Self-Attention Attribution: Interpreting Information Interactions Inside Transformer. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(14):12963–12971, May 2021. ISSN 2374-3468. doi: 10.1609/aaai.v35i14.17533.

[48] Yongchao Zhou, Andrei Ioan Muresanu, Ziwen Han, Keiran Paster, Silviu Pitis, Harris Chan, and Jimmy Ba. Large Language Models Are Human-Level Prompt Engineers, March 2023.

[49] Laria Reynolds and Kyle McDonell. Prompt Programming for Large Language Models: Beyond the Few-Shot Paradigm. In *Extended Abstracts of the 2021 CHI Conference on Human Factors in Computing Systems*, CHI EA '21, pages 1–7, New York, NY, USA, May 2021. Association for Computing Machinery. ISBN 978-1-4503-8095-9. doi: 10.1145/3411763.3451760.

[50] Kaiyang Zhou, Jingkang Yang, Chen Change Loy, and Ziwei Liu. Learning to Prompt for Vision-Language Models. *International Journal of Computer Vision*, 130(9):2337–2348, September 2022. ISSN 1573-1405. doi: 10.1007/s11263-022-01653-1.

[51] Jeremy Howard and Sebastian Ruder. Universal Language Model Fine-tuning for Text Classification. In Iryna Gurevych and Yusuke Miyao, editors, *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 328–339, Melbourne, Australia, July 2018. Association for Computational Linguistics. doi: 10.18653/v1/P18-1031.

[52] Ning Ding, Yujia Qin, Guang Yang, Fuchao Wei, Zonghan Yang, Yusheng Su, Shengding Hu, Yulin Chen, Chi-Min Chan, Weize Chen, Jing Yi, Weilin Zhao, Xiaozhi Wang, Zhiyuan Liu, Hai-Tao Zheng, Jianfei Chen, Yang Liu, Jie Tang, Juanzi Li, and Maosong Sun. Parameter-efficient fine-tuning of large-scale pre-trained language models. *Nature Machine Intelligence*, 5 (3):220–235, March 2023. ISSN 2522-5839. doi: 10.1038/s42256-023-00626-4.

[53] Andreas Miltiadous, Katerina D. Tzimourta, Theodora Afrantou, Panagiotis Ioannidis, Nikolaos Grigoriadis, Dimitrios G. Tsalikakis, Pantelis Angelidis, Markos G. Tsipouras, Euripidis Glavas, Nikolaos Giannakeas, and Alexandros T. Tzallas. A Dataset of Scalp EEG Recordings of Alzheimer's Disease, Frontotemporal Dementia and Healthy Subjects from Routine EEG. *Data*, 8(6):95, June 2023. ISSN 2306-5729. doi: 10.3390/data8060095.

[54] Andreas Miltiadous, Emmanouil Gionanidis, Katerina D. Tzimourta, Nikolaos Giannakeas, and Alexandros T. Tzallas. DICE-Net: A Novel Convolution-Transformer Architecture for Alzheimer Detection in EEG Signals. 11:71840–71858. ISSN 2169-3536. doi: 10.1109/ACCESS.2023.3294618.

[55] J. Escudero, D. Abásolo, R. Hornero, P. Espino, and M. López. Analysis of electroencephalograms in Alzheimer's disease patients with multiscale entropy. *Physiological Measurement*, 27 (11):1091–1106, November 2006. ISSN 0967-3334. doi: 10.1088/0967-3334/27/11/004.

[56] Hanneke van Dijk, Guido van Wingen, Damiaan Denys, Sebastian Olbrich, Rosalinde van Ruth, and Martijn Arns. The two decades brainclinics research archive for insights in neurophysiology (TDBRAIN) database. *Scientific Data*, 9(1):333, June 2022. ISSN 2052-4463. doi: 10.1038/s41597-022-01409-z.

[57] A. L. Goldberger, L. A. Amaral, L. Glass, J. M. Hausdorff, P. C. Ivanov, R. G. Mark, J. E. Mietus, G. B. Moody, C. K. Peng, and H. E. Stanley. PhysioBank, PhysioToolkit, and PhysioNet: components of a new research resource for complex physiologic signals. *Circulation*, 101(23): E215–220, June 2000. ISSN 1524-4539. doi: 10.1161/01.cir.101.23.e215.

[58] Patrick Wagner, Nils Strodthoff, Ralf-Dieter Bousseljot, Dieter Kreiseler, Fatima I. Lunze, Wojciech Samek, and Tobias Schaeffter. PTB-XL, a large publicly available electrocardiography dataset. *Scientific Data*, 7(1):154, May 2020. ISSN 2052-4463. doi: 10.1038/s41597-020-0495-6.

A Comparison of Adaptation Techniques in Foundation Models

As discussed in Section 2, adaptation techniques for foundation models mainly includes *Prompting*, *Fine-tuning*, *Re-programming*, and *Re-purposing*. We have introduced re-programming and re-purposing, and here we provide a brief overview of prompting and fine-tuning, and compare these techniques based on three aspects: *Data Efficiency*, *New Task Type*, and *Generalizability*.

Prompting & Fine-tuning: Both are common adaptation techniques for foundation models, where prompting involves conditioning the model with specific instructions or cues, either handcrafted [48, 49] or learned through data [50], and fine-tuning involves updating the model's internal parameters on dedicated dataset [51, 52]. While they focus on different aspects of adaptation, they share the commonality of not altering the model's core architecture, therefore the functionality of the model remains unchanged, *e.g.*, model for forecasting remains a forecasting model. Moreover, fine-tuning is often more data-intensive, as it necessitates updating the entire model's parameters, whereas prompting typically only requires learning a limited set of task-specific embeddings or prompts.

In general, these techniques can be categorized based on three aspects: *Data efficiency*, as the scale of dataset used for adaptation, typically measured by the number of parameters updated; *New Task Type*, as the ability to adapt to new tasks that are different from the original task, such as from forecasting to classification; and *Generalizability*, as the ability for the adapted model to be used on unseen datasets and share knowledge across tasks. Table 1 provides a comparison of these techniques based on these aspects.

Table 1: Comparison of adaptation techniques of time series foundation models.

Adaptation	Data Efficiency	New Task Type	Generalizability
Prompting	✓		✓[1]
Fine-tuning			✓
Re-programming		✓	
Re-purposing	✓	✓	✓

B Implementation Details

We take TimesFM [21] as the backbone for repurposing based on our preliminary comparative analysis of existing time series foundation models. TimesFM is pre-trained on a largest-scale dataset of diverse time series data for forecasting tasks and is able to capture general time series patterns within dynamic length of historical input. To repurpose it for MedTS classification, we can break down the model's anatomy into three parts, the input patching network, the stacked Transformer, and the output prediction network.

Input Patching Network. Given a univariate time series input $\boldsymbol{x} \in \mathbb{R}^T$ and binary mask $\boldsymbol{m} \in \{0,1\}^T$ with length T, they are first broken up into patches $\boldsymbol{X} \in \mathbb{R}^{L\times P}$ and $\boldsymbol{M} \in \{0,1\}^{L\times P}$ in a non-overlapping fashion, where P is the patch size and $L = \lceil \frac{T}{P} \rceil$ is the number of tokens. Each patch $\boldsymbol{X}_{i,:}$ is the concatenation of P consecutive elements of the input sequence $\boldsymbol{x}$ in a non-overlapping fashion and so is the $\boldsymbol{M}_{i,:}$. The $\boldsymbol{X}_{i,:}$ and $\boldsymbol{M}_{i,:}$ denote the i-th row of $\boldsymbol{X}$ and $\boldsymbol{M}$, respectively. The sequence of patches $\boldsymbol{X}$ and $\boldsymbol{M}$ are then projected to a sequence of tokens $\boldsymbol{Z} \in \mathbb{R}^{L\times D}$ in the model dimension D using an input residual block:

$$\boldsymbol{Z}_{i,:} = \texttt{InputResidualBlock}(\boldsymbol{X}_{i,:}; \boldsymbol{M}_{i,:}) \tag{8}$$

Stacked Transformer. Before passing into the stacked Transformer, the positional encoding will be added to the tokens to form the input sequence $\tilde{\boldsymbol{Z}} \in \mathbb{R}^{L\times D}$. The stacked Transformer is then applied to the input sequence $\tilde{\boldsymbol{Z}}$ to capture the temporal dependencies and extract features using casual self-attention, outputting feature rich tokens $\boldsymbol{H} \in \mathbb{R}^{L\times D}$:

$$\begin{aligned} \tilde{\boldsymbol{Z}}_{i,:} &= \boldsymbol{Z}_{i,:} \oplus \texttt{PositionalEncoding}(i) \\ \boldsymbol{H}_{i,:} &= \texttt{StackedTransformer}(\tilde{\boldsymbol{Z}}_{1,:}, \tilde{\boldsymbol{Z}}_{2,:}, ..., \tilde{\boldsymbol{Z}}_{i,:}; \dot{m}_1, \dot{m}_2, ..., \dot{m}_i) \end{aligned} \tag{9}$$

[1] Although the model structure is fixed and still applicable to other datasets and tasks, the engineered or learned prompts can be task-specific.

where $\dot{m}_i = \min\{\boldsymbol{M}_{i,:}\}$ is the mask for the i-th patch for masking out completely empty ones.

Output Prediction Network. The output prediction network is a residual block layer that maps the last output $\boldsymbol{H}_{L,:}$ from the Transformer back to the original input spaces $\hat{\boldsymbol{x}} \in \mathbb{R}^N$, forming the prediction of the next N time steps:

$$\hat{\boldsymbol{x}} = \texttt{OutputResidualBlock}(\boldsymbol{H}_{L,:}) \tag{10}$$

In summary, the duty of prediction lies solely on the last output prediction network, while the input patching network plus the stacked Transformer can be viewed as a feature extractor that maps the input time series $\boldsymbol{x}$ to a sequence of feature tokens $\boldsymbol{H}$ (Figure 3). This can be easily extended to process multivariate MedTS by processing each channel of input individually and stack the extracted features as $\mathbf{H} \in \mathbb{R}^{C \times L \times D}$ for data of C channels. This will serve as the backbone feature extractor for the downstream classification model.

C Experiment Setup

C.1 Experimental Setup

The experiments are ran on several hosts. The following list summarizes the hardware and software configurations used in our experiments:

Table 2: **Environment setup**.

Host No.	CPU	Memory (GB)	GPU
1	Intel Core i9-10900X	32	NVIDIA RTX A5000 × 1
2	AMD Ryzen Threadripper PRO 3995WX	512	NVIDIA RTX A5000 × 4
3	AMD EPYC 7713	1024	NVIDIA RTX A5000 × 8
4	AMD EPYC 7513	256	NVIDIA RTX A6000 × 8

Software/Package	Version
Python3	3.13.3
PyTorch	2.7.0
CUDA	12.4 – 12.6

C.2 Code Availability

The code will be available upon acceptance.

C.3 Dataset Availability & Preprocessing

The preprocessed datasets in the MedTS cohort are obtained from the authors of Wang et al. [6]. The datasets used for adapting are loaded through `sktime` API and no additional preprocessing is performed.

C.4 Data Splitting

For datasets in the MedTS cohort, the datasets are splitted into training, validation, and test sets in the ratio of 6:2:2 at patient level according to Wang et al. [6, 33]. The adapting datasets are not clearly defined in terms of patient, so we perform best effort to split the datasets into training, validation, and test sets in the ratio of 8:1:1 at recording level. The splitting is seeded and stratified to ensure that the training, validation, and test sets have similar distributions of the target classes. The training set is used for training the model, the validation set is used for early stopping, and the test set is used for evaluating the model performance.

C.5 Model Training

The frozen pre-trained backbone TimesFM model is the v2 version from the official GitHub repository, with 50 layers and model dimension of 1280, and patch size of 32. The classifier in FORMED is

initialized with default initialization method, and the embeddings are initialized to normal distribution with $\mu = 0$ and $\sigma^2 = 0.1$. The model is trained with `AdamW` optimizer with weight decay of 1×10^{-3} and a custom log-normal learning rate schedule. For each epoch t, the learning rate is calculated as:

$$\begin{aligned} \mathcal{LN}(t; \mu, \sigma^2, T) &= \begin{cases} 0 & t = 0 \\ \frac{T}{t} \cdot \exp\left(-\frac{(\ln t - \ln T - \mu)^2}{2\sigma^2}\right) & \text{otherwise} \end{cases} \\ \texttt{lr}(t; \mu, \sigma^2, T) &= \texttt{lr}_{\min} + \frac{\mathcal{LN}(t; \mu, \sigma^2, T)}{\max_t \mathcal{LN}(t; \mu, \sigma^2, T)} \cdot (\texttt{lr}_{\max} - \texttt{lr}_{\min}) \end{aligned} \tag{11}$$

where we use $\mu = 1, \sigma^2 = 1, T = 10$, $\texttt{lr}_{\min} = 1 \times 10^{-5}$ and $\texttt{lr}_{\max} = 1 \times 10^{-3}$ in our experiments. This creates a quick warm-up phase followed by a gradual decay of the learning rate for annealing. The Figure 6 shows the typical training trajectories of FORMED during repurposing.

During each epoch of training, the model is trained on 100 batches of data from each dataset sampled at random order. The batch size is tailored for each dataset according to the available samples in each dataset, making the 100 batches of data approximately equal to one effective iteration of the whole training set. Additionally, gradient clipping of 1.0 is applied to prevent exploding gradients and overshooting in earlier epochs. The model is trained for 100 epochs with early stopping on the validation set based on the average F1-score with patience of 10 epochs. The same training procedure is applied to all TSA and our method.

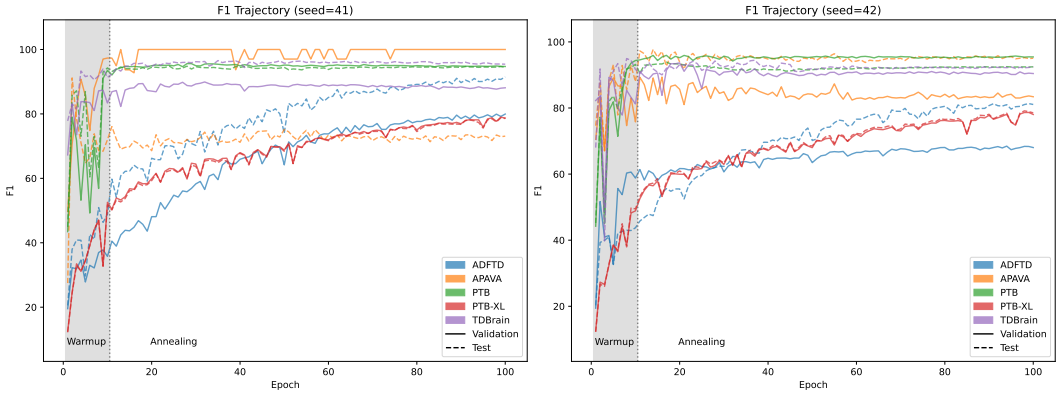

Figure 6: Typical training trajectories of FORMED during repurposing from two different seeds.

C.6 Hyperparameter Tuning

The initial hyperparameter k for the classifier in FORMED was not tuned due to the high cost associated with repurposing extra-large models like TimesFM, but rather set to an arbitrary value of $k = 16$ for all datasets. However, we do perform a grid search for the hyperparameter k during the adapting process, as shown in Figure 5. We initiate the search with powers of 4, until the performance shows no significant improvement. Then we perform a finer search with powers of 2. We find that $k = 256 \sim 1024$ is a good range for adapting to small datasets. This search is very efficient as the computation cost of increasing k is often negligible compared to the computation cost of large model backbones, despite being linear in k theoretically.

D Datasets

Here we provide the details of the datasets Table 3 used as the MedTS cohort for repurposing in Section 5. The datasets are publicly available, and we follow the pre-processing and splitting procedures as in Wang et al. [6].

E Experimental Results

Table 3: **MedTS Cohort Datasets**.

Dataset	Type	# Subject	# Sample	Sampling Rate	Sampling Length	# Channel	# Classes
ADFTD [53, 54]	EEG	88	69 762	256 Hz	256	19	3
APAVA [55]	EEG	23	5967	256 Hz	256	16	2
TDBrain [56]	EEG	72	6240	256 Hz	256	33	2
PTB [57]	ECG	198	64 356	250 Hz	300	15	2
PTB-XL [58]	ECG	17 596	191 400	250 Hz	250	12	5

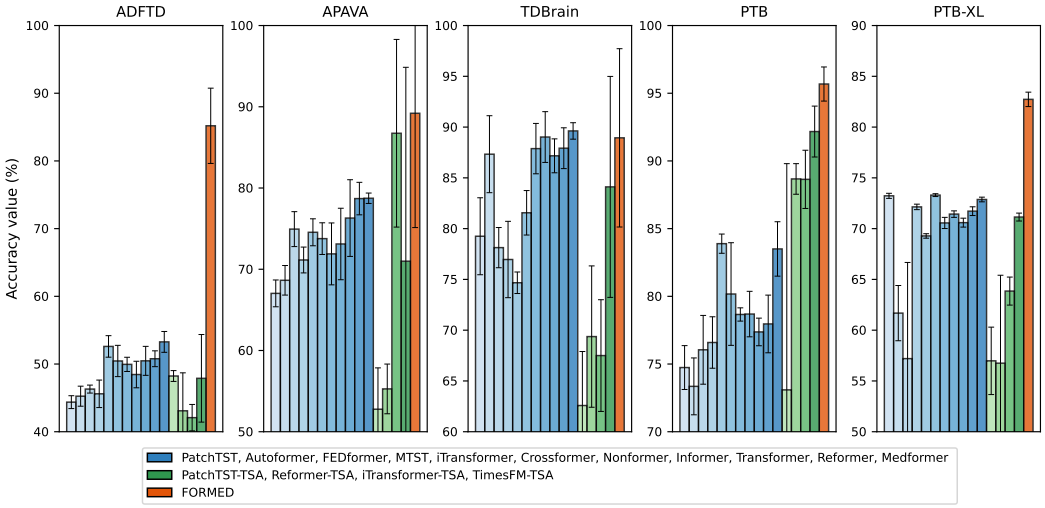

Figure 7: In-domain accuracy performance on the MedTS cohort datasets.

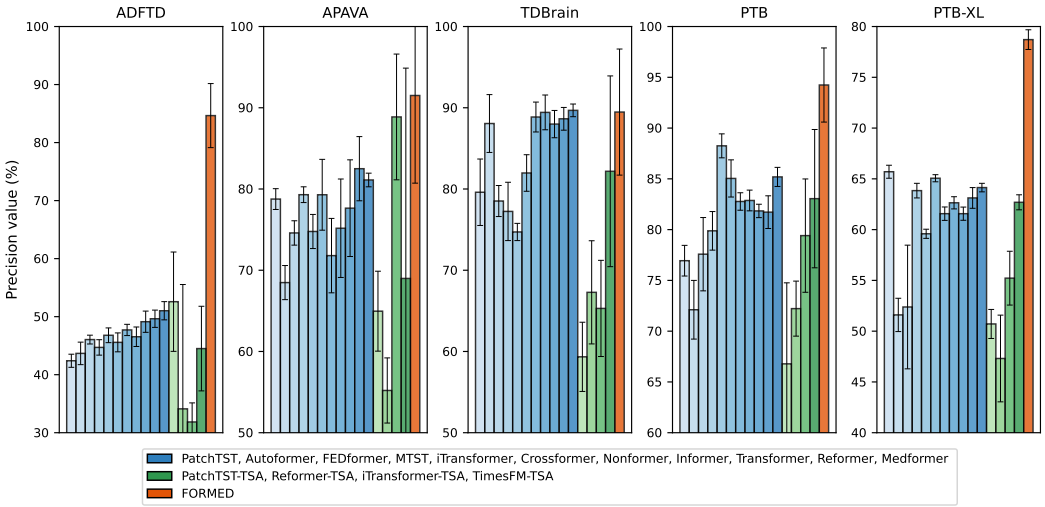

Figure 8: In-domain precision performance on the MedTS cohort datasets.

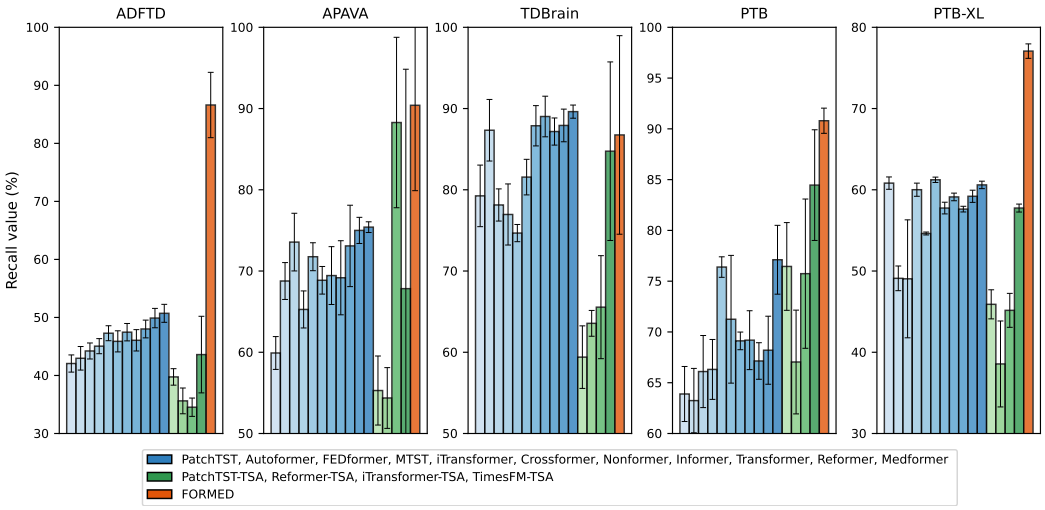

Figure 9: In-domain recall performance on the MedTS cohort datasets.

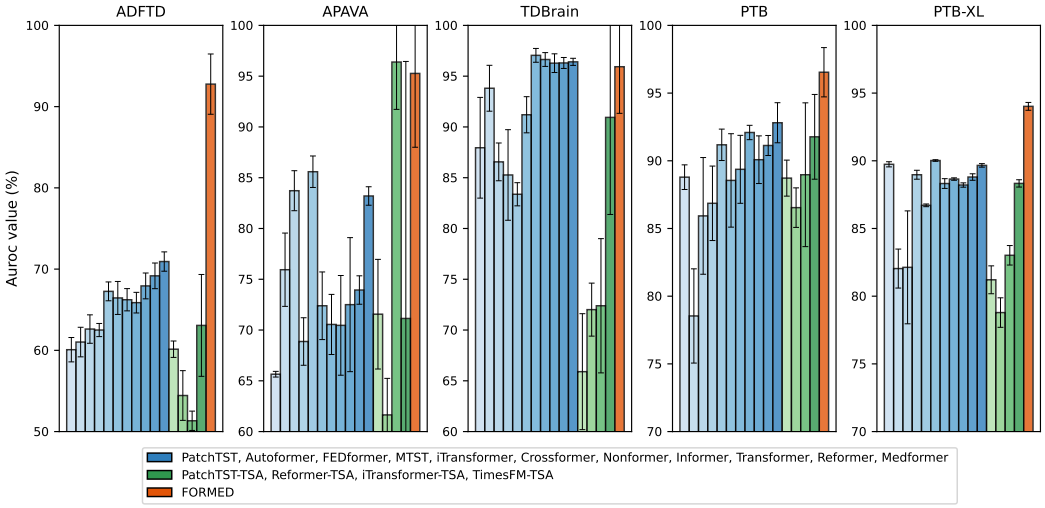

Figure 10: In-domain AUROC performance on the MedTS cohort datasets.

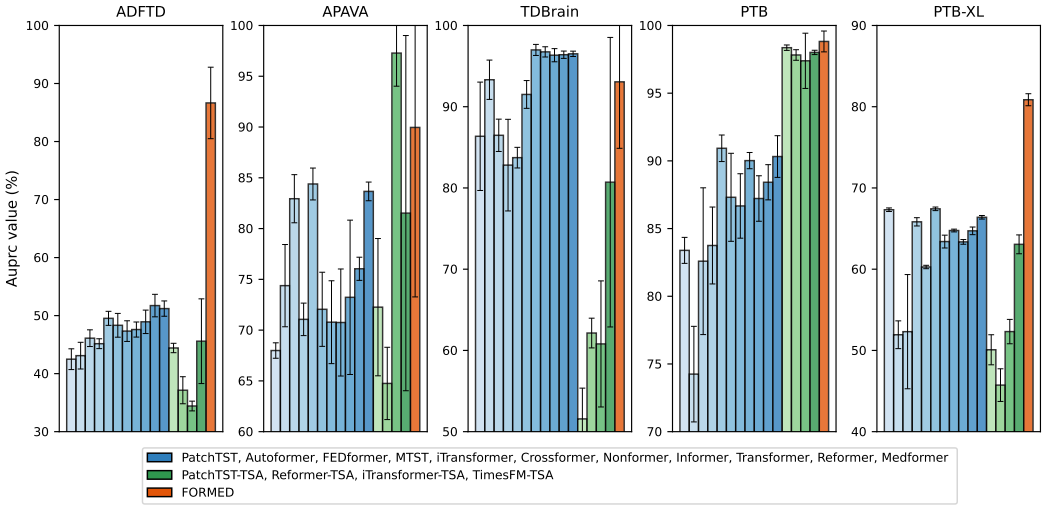

Figure 11: In-domain AUPRC performance on the MedTS cohort datasets.

Table 4: **Results on MedTS Cohort for disease classification.** Best results from non-TSM models are **bolded** and best results of all models are underlined. TimesFM-TSA shows great improvement over other TSA models and achieves competitive performance with the SOTA TSM models. While our model, FORMED, consistently outperforms the all other model on all datasets.

Datasets	Adaptation	Models	Accuracy	Precision	Recall	F1 score	AUROC	AUPRC
ADFTD (3-Classes)	TSM	**Autoformer**	45.25±1.48	43.67±1.94	42.96±2.03	42.59±1.85	61.02±1.82	43.10±2.30
		Crossformer	50.45±2.31	45.57±1.63	45.88±1.82	45.50±1.70	66.45±2.03	48.33±2.05
		FEDformer	46.30±0.59	46.05±0.76	44.22±1.38	43.91±1.37	62.62±1.75	46.11±1.44
		Informer	48.45±1.96	46.54±1.68	46.06±1.84	45.74±1.38	65.87±1.27	47.60±1.30
		iTransformer	52.60±1.59	46.79±1.27	47.28±1.29	46.79±1.13	67.26±1.16	49.53±1.21
		MTST	45.60±2.03	44.70±1.33	45.05±1.30	44.31±1.74	62.50±0.81	45.16±0.85
		Nonformer	49.95±1.05	47.71±0.97	47.46±1.50	46.96±1.35	66.23±1.37	47.33±1.78
		PatchTST	44.37±0.95	42.40±1.13	42.06±1.48	41.97±1.37	60.08±1.50	42.49±1.79
		Reformer	50.78±1.17	49.64±1.49	49.89±1.67	47.94±0.69	69.17±1.58	51.73±1.94
		Transformer	50.47±2.14	49.13±1.83	48.01±1.53	48.09±1.59	67.93±1.59	48.93±2.02
		Medformer	53.27±1.54	51.02±1.57	50.71±1.55	50.65±1.51	70.93±1.19	51.21±1.32
	TSA	**iTransformer-TSA**	42.08±1.95	31.85±3.29	34.53±1.58	29.89±1.86	51.33±1.19	34.42±0.84
		PatchTST-TSA	48.23±0.80	52.58±8.56	39.75±1.42	35.87±3.27	60.14±1.01	44.42±0.81
		Reformer-TSA	43.09±5.61	34.11±21.43	35.62±2.22	26.21±7.22	54.44±3.06	37.14±2.33
		TimesFM-TSA	47.89±6.47	44.50±7.29	43.60±6.60	43.61±6.82	63.07±6.27	45.59±7.30
	GA	**FORMED (Ours)**	**85.19±5.56**	**84.65±5.52**	**86.61±5.63**	**85.12±5.64**	**92.77±3.71**	**86.65±6.16**
APAVA (2-Classes)	TSM	**Autoformer**	68.64±1.82	68.48±2.10	68.77±2.27	68.06±1.94	75.94±3.61	74.38±4.05
		Crossformer	73.77±1.95	79.29±4.36	68.86±1.70	68.93±1.85	72.39±3.33	72.05±3.65
		FEDformer	74.94±2.15	74.59±1.50	73.56±3.55	73.51±3.39	83.72±1.97	82.94±2.37
		Informer	73.11±4.40	75.17±6.06	69.17±4.56	69.47±5.06	70.46±4.91	70.75±5.27
		iTransformer	74.55±1.66	74.77±2.10	71.76±1.72	72.30±1.79	85.59±1.55	84.39±1.57
		MTST	71.14±1.59	79.30±0.97	65.27±2.28	64.01±3.16	68.87±2.34	71.06±1.60
		Nonformer	71.89±3.81	71.80±4.58	69.44±3.56	69.74±3.84	70.55±2.96	70.78±4.08
		PatchTST	67.03±1.65	78.76±1.28	59.91±2.02	55.97±3.10	65.65±0.28	67.99±0.76
		Reformer	78.70±2.00	82.50±3.95	75.00±1.61	75.93±1.82	73.94±1.40	76.04±1.14
		Transformer	76.30±4.72	77.64±5.95	73.09±5.01	73.75±5.38	72.50±6.60	73.23±7.60
		Medformer	78.74±0.64	81.11±0.84	75.40±0.66	76.31±0.71	83.20±0.91	83.66±0.92
	TSA	**iTransformer-TSA**	86.74±11.55	88.87±7.74	88.28±10.49	86.50±11.69	**96.39±4.66**	**97.28±3.26**
		PatchTST-TSA	52.77±5.09	64.96±4.92	55.29±4.25	45.00±10.68	71.56±5.40	72.26±6.76
		Reformer-TSA	55.27±3.06	55.20±4.01	54.37±3.74	52.69±4.24	61.64±3.60	64.75±3.56
		TimesFM-TSA	70.99±23.87	68.99±25.89	67.83±27.01	65.68±28.26	71.14±25.31	81.52±17.49
	GA	**FORMED (Ours)**	**89.20±14.07**	**91.51±10.79**	**90.40±10.51**	**88.97±13.85**	95.27±7.27	89.96±16.69
TDBrain (2-Classes)	TSM	**Autoformer**	87.33±3.79	88.06±3.56	87.33±3.79	87.26±3.84	93.81±2.26	93.32±2.42
		Crossformer	81.56±2.19	81.97±2.25	81.56±2.19	81.50±2.20	91.20±1.78	91.51±1.71
		FEDformer	78.13±1.98	78.52±1.91	78.13±1.98	78.04±2.01	86.56±1.86	86.48±1.99
		Informer	89.02±2.50	89.43±2.14	89.02±2.50	88.98±2.54	96.64±0.68	96.75±0.63
		iTransformer	74.67±1.06	74.71±1.06	74.67±1.06	74.65±1.06	83.37±1.14	83.73±1.27
		MTST	76.96±3.76	77.24±3.59	76.96±3.76	76.88±3.83	85.27±4.46	82.81±5.64
		Nonformer	87.88±2.48	88.86±1.84	87.88±2.48	87.78±2.56	97.05±0.68	96.99±0.68
		PatchTST	79.25±3.79	79.60±4.09	79.25±3.79	79.20±3.77	87.95±4.96	86.36±6.67
		Reformer	87.92±2.01	88.64±1.40	87.92±2.01	87.85±2.08	96.30±0.54	96.40±0.45
		Transformer	87.17±1.67	87.99±1.68	87.17±1.67	87.10±1.68	96.28±0.92	96.34±0.81
		Medformer	89.62±0.81	89.68±0.78	89.62±0.81	89.62±0.81	96.41±0.35	96.51±0.33
	TSA	**iTransformer-TSA**	67.50±5.50	65.30±5.92	65.55±6.34	65.07±5.92	72.39±6.61	60.80±7.76
		PatchTST-TSA	62.58±5.32	59.34±4.26	59.40±3.85	58.08±3.28	65.90±5.71	51.57±3.79
		Reformer-TSA	69.37±6.96	67.28±6.35	63.56±1.59	62.93±3.28	72.01±2.61	62.14±1.83
		TimesFM-TSA	84.11±10.88	82.18±11.74	84.75±10.99	82.78±11.75	90.94±9.56	80.71±17.82
	GA	**FORMED (Ours)**	**88.94±8.78**	**89.47±7.76**	**86.75±12.22**	**87.30±11.36**	**95.93±4.59**	**93.06±8.18**
PTB (2-Classes)	TSM	**Autoformer**	73.35±2.10	72.11±2.89	63.24±3.17	63.69±3.84	78.54±3.48	74.25±3.53
		Crossformer	80.17±3.79	85.04±1.83	71.25±6.29	72.75±7.19	88.55±3.45	87.31±3.25
		FEDformer	76.05±2.54	77.58±3.61	66.10±3.55	67.14±4.37	85.93±4.31	82.59±5.42
		Informer	78.69±1.68	82.87±1.02	69.19±2.90	70.84±3.47	92.09±0.53	90.02±0.60
		iTransformer	83.89±0.71	88.25±1.18	76.39±1.01	79.06±1.06	91.18±1.16	90.93±0.98
		MTST	76.59±1.90	79.88±1.90	66.31±2.95	67.38±3.71	86.86±2.75	83.75±2.84
		Nonformer	78.66±0.49	82.77±0.86	69.12±0.87	70.90±1.00	89.37±2.51	86.67±2.38
		PatchTST	74.74±1.62	76.94±1.51	63.89±2.71	64.36±3.38	88.79±0.91	83.39±0.96
		Reformer	77.96±2.13	81.72±1.61	68.20±3.35	69.65±3.88	91.13±0.74	88.42±1.30
		Transformer	77.37±1.02	81.84±0.66	67.14±1.80	68.47±2.19	90.08±1.76	87.22±1.68
		Medformer	83.50±2.01	85.19±0.94	77.11±3.39	79.18±3.31	92.81±1.48	90.32±1.54
	TSA	**iTransformer-TSA**	88.64±2.15	79.41±5.58	75.74±7.35	75.64±5.13	88.97±5.31	97.39±2.04
		PatchTST-TSA	73.09±16.71	66.78±7.98	76.45±4.32	62.87±12.19	88.72±1.33	98.36±0.20
		Reformer-TSA	88.67±1.13	72.22±2.71	67.04±5.11	68.44±3.75	86.54±1.46	97.82±0.39
		TimesFM-TSA	92.17±1.88	83.05±6.81	84.46±5.45	83.63±6.02	91.77±3.13	98.01±0.16
	GA	**FORMED (Ours)**	**95.68±1.26**	**94.24±3.65**	**90.80±1.24**	**92.35±2.15**	**96.54±1.82**	**98.82±0.77**
PTB-XL (5-Classes)	TSM	**Autoformer**	61.68±2.72	51.60±1.64	49.10±1.52	48.85±2.27	82.04±1.44	51.93±1.71
		Crossformer	73.30±0.14	65.06±0.35	61.23±0.33	62.59±0.14	90.02±0.06	67.43±0.22
		FEDformer	57.20±9.47	52.38±6.09	49.04±7.26	47.89±8.44	82.13±4.17	52.31±7.03
		Informer	71.43±0.32	62.64±0.60	59.12±0.47	60.44±0.43	88.65±0.09	64.76±0.17
		iTransformer	69.28±0.22	59.59±0.45	54.62±0.18	56.20±0.19	86.71±0.10	60.27±0.21
		MTST	72.14±0.27	63.84±0.72	60.01±0.81	61.43±0.38	88.97±0.33	65.83±0.51
		Nonformer	70.56±0.55	61.57±0.66	57.75±0.72	59.10±0.66	88.32±0.36	63.40±0.79
		PatchTST	73.23±0.25	65.70±0.64	60.82±0.76	62.61±0.34	89.74±0.19	67.32±0.22
		Reformer	71.72±0.43	63.12±1.02	59.20±0.75	60.69±0.18	88.80±0.24	64.72±0.47
		Transformer	70.59±0.44	61.57±0.65	57.62±0.35	59.05±0.25	88.21±0.16	63.36±0.29
		Medformer	72.87±0.23	64.14±0.42	60.60±0.46	62.02±0.37	89.66±0.13	66.39±0.22
	TSA	**iTransformer-TSA**	63.85±1.38	55.22±2.65	45.16±2.09	45.96±2.11	83.02±0.72	52.32±1.50
		PatchTST-TSA	56.98±3.32	50.71±1.43	45.90±1.79	42.97±2.93	81.21±1.03	50.08±1.85
		Reformer-TSA	56.75±8.66	47.31±4.27	38.56±5.29	36.84±8.58	78.79±1.09	45.73±2.01
		TimesFM-TSA	71.14±0.39	62.69±0.74	57.75±0.49	59.61±0.51	88.33±0.27	63.06±1.16
	GA	**FORMED (Ours)**	**82.73±0.71**	**78.71±0.97**	**77.07±0.89**	**77.83±0.92**	**94.02±0.29**	**80.86±0.74**

Table 5: **Results on adapting to unseen datasets.** Best results are highlighted in **bold**.

Datasets	Model	k factor	Accuracy	Precision	Recall	F1 score	AUROC	AUPRC
ECG200	TimesFM-TSA	N/A	66.32±4.71	40.89±15.98	52.95±11.39	45.23±12.63	80.26±9.79	91.24±5.22
	FORMED	4	52.63±18.23	55.99±17.59	56.41±17.19	51.28±17.23	65.90±18.05	81.64±10.65
		8	43.16±15.52	43.68±11.08	45.00±10.89	39.40±14.06	45.13±15.84	71.92±8.59
		16	57.89±11.77	49.65±16.85	51.28±17.36	49.86±16.41	68.46±20.1	85.77±8.72
		32	64.21±12.57	54.57±18.70	55.90±17.89	54.46±18.12	63.33±14.72	80.85±8.63
		64	70.53±4.71	65.73±5.64	62.31±5.45	62.24±6.58	73.85±7.23	88.05±4.42
		128	70.53±4.71	65.39±6.96	60.51±5.53	60.59±6.47	75.90±4.19	89.96±1.81
		256	84.21±8.32	82.40±8.57	82.18±14.44	80.56±12.31	89.23±11.13	95.35±5.06
		512	87.37±2.88	85.38±2.94	88.08±6.01	85.87±3.91	95.90±4.66	98.27±1.97
		1024	**88.42±2.35**	**86.67±1.86**	**91.54±1.72**	**87.65±2.33**	**98.21±1.15**	**99.23±0.50**
		2048	87.37±4.71	85.41±4.87	88.97±5.31	86.25±4.97	95.90±3.56	98.22±1.56
StandWalkJump	TimesFM-TSA	N/A	33.33±9.43	33.83±12.62	33.33±9.43	32.41±10.64	52.00±5.79	44.89±4.97
	FORMED	4	21.33±7.30	19.23±7.84	21.33±7.30	19.69±7.22	44.00±10.66	42.07±7.44
		8	18.67±12.82	19.18±13.99	18.67±12.82	18.68±13.19	46.13±12.73	40.63±7.32
		16	36.00±10.11	35.22±10.23	36.00±10.11	34.15±8.57	53.60±10.18	46.11±8.79
		32	36.00±18.01	37.45±16.15	36.00±18.01	35.87±17.37	57.33±13.64	53.11±15.30
		64	40.00±15.63	36.30±15.11	40.00±15.63	37.50±14.77	56.80±14.88	50.30±13.12
		128	46.67±10.54	43.96±11.27	46.67±10.54	44.26±12.08	63.60±11.72	55.75±13.10
		256	46.67±13.33	47.33±14.87	46.67±13.33	46.29±14.25	63.60±1.80	53.81±2.64
		512	**64.00±11.16**	**67.25±10.53**	**64.00±11.16**	**64.52±10.48**	**68.13±5.06**	**57.27±4.25**
		1024	54.67±7.30	55.24±6.28	54.67±7.30	54.23±7.44	62.13±1.66	53.30±1.42
		2048	60.00±6.67	62.29±7.03	60.00±6.67	60.12±6.63	65.33±2.98	56.34±2.27

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction accurately reflect the paper's contributions and scope.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The limitations are discussed in Section 6.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [NA]

Justification: There are no theoretical results in the paper.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: All details are described in Section 5 and Appendix C.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: The data can be obtained from the authors of Wang et al. [6] and openly available UCR UEA archive, as described in Appendix C. We will provide public access to code upon acceptance.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: All details are described in Section 5 and Appendix C.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: The error bars or raw data points are plotted in the figures (Figures 4, 5 and 7 to 11) and the standard deviation is reported in the tables (Tables 4 and 5).

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)

- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We provide the environment used to run the experiments in Appendix C. The exact training time is missing due to the use of multiple different machines and there is no consistent way to measure the time of execution.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We use publicly available and deidentified datasets (as described in Appendix D) in our experiments, and respect the NeurIPS Code of Ethics.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [No]

    Justification: The paper is focused on technical contributions and propose a solution to tackle the unique challenges in medical time series classification. Although the application field is healthcare, the paper does not directly relate to any specific application. The authors do not foresee any negative societal impacts of the work performed.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.

- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: The paper does not fall into the category of high-risk models.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: The paper properly credits the creators of the datasets and models used. The license and terms of use are also not present in the paper, but the authors will provide them along with the code release.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.

- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [NA]

    Justification: The paper does not release new assets.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.

- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

    Answer: [NA]

    Justification: The LLMs are not used for the core methods in this research.

    Guidelines:

    - The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy (https://neurips.cc/Conferences/2025/LLM) for what should or should not be described.