TransEHR: Self-Supervised Transformer for Clinical Time Series Data

Yanbo Xu* YXU465@GATECH.EDU
Shangqing Xu* SXU452@GATECH.EDU
Manav Ramprassad MANAV.RAMPRASAD@GATECH.EDU
Alexey Tumanov ATUMANOV@GATECH.EDU
Chao Zhang CHAOZHANG@GATECH.EDU
Georgia Institute of Technology, GA

Abstract

Deep neural networks, including the Transformer architecture, have achieved remarkable performance in various time series tasks. However, their effectiveness in handling clinical time series data is hindered by specific challenges: 1) Sparse event sequences collected asynchronously with multivariate time series, and 2) Limited availability of labeled data. To address these challenges, we propose TransEHR[1], a self-supervised Transformer model designed to encode multi-sourced asynchronous sequential data, such as structured Electronic Health Records (EHRs), efficiently. We introduce three pretext tasks for pre-training the Transformer model, utilizing large amounts of unlabeled structured EHR data, followed by fine-tuning on downstream prediction tasks using the limited labeled data. Through extensive experiments on three real-world health datasets, we demonstrate that our model achieves state-of-the-art performance on benchmark clinical tasks, including in-hospital mortality classification, phenotyping, and length-of-stay prediction. Our findings highlight the efficacy of TransEHR in effectively addressing the challenges associated with clinical time series data, thus contributing to advancements in healthcare analytics.

1. Introduction

Clinical time series data, such as structured Electronic Health Records (EHRs), provide critical information for healthcare. They contain detailed measurements and clinical events that help diagnose, treat, and manage patients. Structured EHR data is generated continuously throughout a patient's hospital stay and can include a wide range of information, such as vital signs, lab tests, and medication administrations. Recently, deep sequential models, such as recurrent neural networks (RNNs) and Transformers, have been applied to clinical time series data and achieved state-of-the-art performance in predicting patient outcomes such as mortality, length of stay, and sepsis onset (Xu et al., 2018; Zerveas et al., 2021; Tipirneni and Reddy, 2022). However, the predictive performance of these models is often hindered by two unique challenges in complex clinical time-series data.

One of the main challenges in clinical time series data is the presence of diverse data types. It encompasses continuous multivariate time series, like evenly sampled vital signs, as well as discrete event sequences, including lab tests and medication administrations, which are generated asynchronously with random timestamps. Although deep learning advancements in healthcare primarily concentrate on continuous measurements (Xu et al., 2018; Che et al., 2018; Shukla and Marlin, 2019; Horn et al., 2020; Zerveas et al., 2021; Tipirneni and Reddy, 2022), the event data, particularly the timestamp of medical events, can contain more critical information. Alternatively, neural point process models (Zuo et al., 2020; Zhang et al., 2020) have achieved successful applications in analyzing short event sequences within the medical domain. However, these models overlook the presence of regularly recorded continuous measurements in conjunction with long irregularly sampled discrete events. Therefore, the joint modeling of multiple sources of complex structured EHR data presents a nontrivial task. The second challenge lies in the difficulty of obtaining labeled data, despite the abundance of input data collected in EHR systems. Acquiring clinical labels can be costly and may raise ethical concerns. Labeling the data often demands substantial time, effort, specialized infrastructure, and domain exper-

* These authors contributed equally
1. Our code is available at https://github.com/SigmaTsing/TransEHR.git.

tise. Consequently, there is a pressing need to develop healthcare models capable of delivering accurate predictions with limited labeled data or by leveraging the vast amount of available unlabeled data.

To tackle these challenges, we propose `TransEHR`, a self-supervised Transformer model for modeling clinical time series data from contemporary EHR systems. Our framework offers a unified triplet representation that encompasses the two types of structured EHR time series. This asynchronous triplet representation eliminates the necessity for traditional data preprocessing techniques that involve timeline discretization and binning values into two-dimensional feature matrices, which may lead to the loss of crucial temporal information. Instead, `TransEHR` embeds triplets using distinct encoding procedures and leverages the Transformer architecture to capture the intricate temporal dependencies present in the sequences. The non-recurrent structure of the Transformer facilitates excellent training efficiency and scalability, unlike recurrent networks that often encounter issues such as inefficient sequential processing and gradient vanishing problems.

To leverage unlabeled data, our proposed model, `TransEHR`, utilizes self-supervision during training. It pretrains the Transformer model on a pretext task with unlabeled data, then fine-tunes it for downstream supervised tasks using labeled data. The objective is to tailor the pretext tasks specifically for multi-sourced EHR time-series data, aiming to learn robust representations. We employ a denoising method inspired by masked language modeling (MLM), corrupting the input by randomly masking values within triplets and predicting masked values using Mean Square Error (MSE) loss. Additionally, we propose training a discriminative model to predict the substitution of triplets (Clark et al., 2020), another pretext task based on a sample-efficient approach. We also design a third pretext task focused on training a Transformer Hawkes process model (Zuo et al., 2020) to forecast event occurrence based on historical data. This task aims to capture changes in patients' status and aids in predicting clinical tasks like disease risk and mortality.

We pretrain `TransEHR` on three real-world health datasets and evaluate its performance on benchmark predictive tasks: in-hospital mortality, length-of-stay and phenotype classification. Our model surpasses supervised and self-supervised learning baselines, achieving a 3.0% AU-PRC improvement for mortality classification and 1.5 days MAE reduction for LOS prediction compared to the previous state-of-the-art method. Remarkably, `TransEHR` achieves comparable AU-ROC scores using only 50% of the labeled data for fine-tuning. Our results highlight the significance of modeling rich event information alongside conventional multivariate measurements in clinical time series data.

2. Related Work

Transformer for multivariate time series data. Most of the deep models developed for time series are still based on recurrent or temporal convolutional neural networks, which hinders parallel processing. Recent work (Horn et al., 2020) utilizes differentiable set function learning that is highly parallelizable and has a low memory footprint, making it scalable for large datasets with long time series. With the success of Transformer models in computer vision (CV) and natural language processing (NLP) domains, a few recent works have started employing the Transformer architecture for EHR time series data. BEHRT (Li et al., 2020) is designed for simple EHR data, which only contains sparse diagnosis sequences, and does not apply to modern EHR data containing real-valued multivariate time series. Zerveas et al. (2021) extended the masked Transformer language model to handle multivariate time series. In their approach, they treat time series as a set of triplets and propose Continuous Value Embedding to encode continuous time and variable values without discretization (Tipirneni and Reddy, 2022).

Transformer for event data. Temporal point processes, such as the Hawkes process (Hawkes, 1971), utilize an *intensity function* to model event sequences. Neural Hawkes process models parameterize the intensity function using recurrent neural networks (Du et al., 2016; Mei and Eisner, 2017) and Transformers (Zuo et al., 2020; Zhang et al., 2020). The Transformer model consists of multiple Transformer layers, each containing a self-attention mechanism and a feed-forward neural network. The attention mechanism assigns attention weights between any two events, where small weights indicate weak dependencies and large weights indicate strong dependencies. This makes the Transformer model more powerful than recurrent neural networks in modeling event sequences.

Self-supervised learning. The current state-of-the-art methods for language representation learning, such as BERT (Kenton and Toutanova, 2019) and XLNet (Yang et al., 2019), utilize denoising autoencoders (Vincent et al., 2008) that mask a small subset of the input sequence and train the network to reconstruct the original input. However, these MLM approaches incur high computational costs because the network only learns from a fraction of the tokens per example. In an alternative approach, Clark et al. (2020) propose a replaced token detection pretraining task. This method corrupts the input by replacing some of the tokens with samples from a proposal distribution. The model then learns to distinguish between genuine input tokens and synthetic replacements, making it more computationally efficient compared to MLM.

While self-supervised learning has demonstrated significant performance improvements in image and text data, its application to time-series data has been limited. However, some efforts have been made in this area. For example, Jawed et al. (2020) used a 1D CNN for dense univariate time-series classification, achieving increased accuracy by incorporating forecasting as an additional task in a multi-task learning framework. Zerveas et al. (2021) also pretrained a Transformer model using a denoising objective and demonstrated improved performance on regression and classification tasks involving dense multivariate time series. Furthermore, Tipirneni and Reddy (2022) demonstrated that time-series forecasting can be a viable and effective self-supervision task for pretraining a Transformer model; Dong et al. (2023) propose an alternative self-reconstruction task considering neighborhood similarity; Zhou et al. (2023) apply frozen pre-trained language models as a series encoder.

3. Method

3.1. Problem Setup

The goal of `TransEHR` is to learn robust representations from the multi-sourced EHR data, consisting of both continuous multivariate time series and discrete event sequences in continuous timelines, with limited labeled data. For multivariate time series, we use a triplet $(\boldsymbol{z}_j, \boldsymbol{v}_j, t_j)$ to denote a K-dimensional multivariate data point observed at time $t_j \in \mathbb{R}^+$, where the indicator vector $\boldsymbol{z}_j \in \{0,1\}^K$ for $[\boldsymbol{z}_j]_k = 1$ if the k-th variate is observed at time t_j and 0 otherwise, the value vector $\boldsymbol{v}_j \in (\mathbb{R} \cup \emptyset)^K$ for $[\boldsymbol{v}_j]_k \in \mathbb{R}$ if a value was measured at time t_j and $\emptyset$ if the value is missing. Similarly, to denote a set of events that happened at time t_s, we can use a triplet $(\boldsymbol{e}_s, \emptyset, t_s)$, where event indicator $\boldsymbol{e}_s = \{0,1\}^D$ for $[\boldsymbol{e}_s]_d = 1$ if the d-th event occurred at time t_s and 0 otherwise. Together with a static feature vector $\boldsymbol{x}_0 \in \mathbb{R}^L$ denoting the characteristics of a patient that do not vary over time, we can represent the observed EHR time series input as $\mathcal{D}_N = \left\{(\boldsymbol{z}_j^{(i)}, \boldsymbol{v}_j^{(i)}, t_j^{(i)})|_{j=1}^{T_i}, (\boldsymbol{e}_{s_j}^{(i)}, \emptyset, t_{s_j}^{(i)})|_{s_j=1}^{S_i}, \boldsymbol{x}_0^{(i)}\right\}_{i=1}^N$, where the superscript (i) denotes patient i in the data set. Beyond the unlabeled data $\mathcal{D}_N$, we also have access to a limited data set $\mathcal{D}_{N'}$ with labels $\{y^{(i')}\}_{i'=1}^{N'}$ (usually $N' << N$). Our objective is to learn a Transformer model $\boldsymbol{f}(\cdot)$ by first pre-training it on $\mathcal{D}_N$ with self-supervision and then tuning it on $\mathcal{D}_{N'}$ for the final supervised downstream task. We summarize the notations used in this paper in Appendix A Table 4.

3.2. Triplet Embedding

Given the three different attributes specified in the triplets for representing an EHR input data point, we apply different encoding procedures to obtain their initial embeddings accordingly.

Categorical type encoding. We fit an embedding matrix $\boldsymbol{U} = [\boldsymbol{U}_K, \boldsymbol{U}_D] \in \mathbb{R}^{M\times(K+D)}$ for encoding the indicator vectors $\boldsymbol{z}_j$ and $\boldsymbol{e}_s$. That is, we compute their embeddings as $\boldsymbol{U}_K \boldsymbol{z}_j^\top$ and $\boldsymbol{U}_D \boldsymbol{e}_s^\top$ respectively.

Continuous value encoding. We fit a linear projection layer to encode the dense multivariate value vector $\boldsymbol{v}_j \in \mathbb{R}^{K\times 1}$ observed at time t_j. It is zero-padded if any value is missing in the original observation or masked in a pretext task (will introduce later). Each element $[\boldsymbol{v}_j]_k$ is normalized using the mean and variance computed from training samples, so the value embedding vector $\boldsymbol{c}_j \in \mathbb{R}^{M\times 1}$ is defined as

$$\boldsymbol{c}_j = \boldsymbol{W}_v \boldsymbol{v}_j + \boldsymbol{b}_v$$

with parameters $\boldsymbol{W}_v \in \mathbb{R}^{M\times K}$ and $\boldsymbol{b}_v \in \mathbb{R}^{M\times 1}$.

Temporal time encoding. We use trigonometric functions (Zuo et al., 2020) to generate deterministic temporal embedding vector $\boldsymbol{u}(t) \in \mathbb{R}^{M\times 1}$ for time t:

$$[\boldsymbol{u}(t)]_m = \begin{cases} \cos\left(t/10000^{\frac{m-1}{M}}\right), & \text{if } m \text{ is odd,} \\ \sin\left(t/10000^{\frac{m}{M}}\right), & \text{if } m \text{ is even,} \end{cases}$$

where $[\boldsymbol{u}(t)]_m$ is the m-th element of vector $\boldsymbol{u}(t)$.

Note that all three encoding vectors reside in the embedding space $\mathbb{R}^{M\times 1}$. Thus, given a multivariate triplet $(\boldsymbol{z}_j, \boldsymbol{v}_j, t_j)$ or an event triplet $(\boldsymbol{e}_s, \emptyset, t_s)$, we can compute the corresponding M-dimensional embedding as a sum of the single-attribute encoding vectors:

$$\text{Multivariate Embedding:} \quad \boldsymbol{X}_j = \boldsymbol{U}_K \boldsymbol{z}_j^\top + \boldsymbol{c}_j + \boldsymbol{u}(t_j),$$

$$\text{Event Embedding:} \quad \boldsymbol{E}_s = \boldsymbol{U}_D \boldsymbol{e}_s^\top + \boldsymbol{u}(t_s).$$

After the embedding layers, we obtain $\boldsymbol{X} = [\boldsymbol{X}_1, \cdots, \boldsymbol{X}_T] \in \mathbb{R}^{M\times T}$ for a multivariate time series input and $\boldsymbol{E} = [\boldsymbol{E}_1, \cdots, \boldsymbol{E}_S] \in \mathbb{R}^{M\times S}$ for an

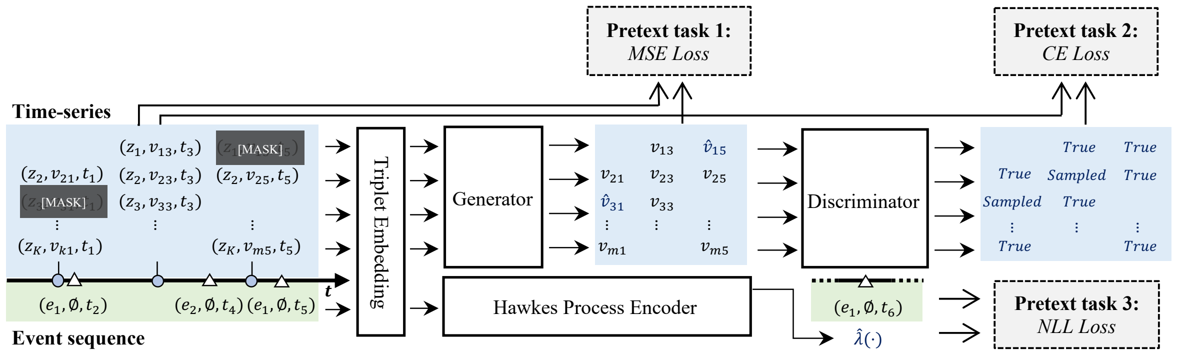


Figure 1: An overview of the three pretext tasks designed in `TransEHR` for self-supervised learning on unlabeled EHR time series data.

event sequence input. We pass them through two separate self-attention modules that share the same architecture. Details of the Transformer architecture are described in Appendix B.

3.3. Self-Supervised Learning

We design three pretext tasks for pre-training the Transformer model in `TransEHR` by leveraging the unlabelled EHR data $\mathcal{D}_N$, depicted in Figure 1.

Pretext task 1: Denoising values. We define the first pretext task by drawing inspiration from the widely used masked language modeling MLM technique in NLP. The core idea of MLM is to randomly mask certain positions within the input sequence and train a generator primarily consisting of a Transformer network encoder to predict the original values of the masked tokens.

In our context, considering a multivariate input time series denoted as $\boldsymbol{x} = (\boldsymbol{z_j}, \boldsymbol{vj}, \boldsymbol{t_j})|j=1^T$, we introduce a random set of masks denoted as $\boldsymbol{m} = \{(k,j) : 1 \le k \le K, 1 \le j \le T\}$, and obtain the masked input as

$$\boldsymbol{x}^{\text{masked}} = \text{REPLACE}(\boldsymbol{x}, \boldsymbol{m}, \texttt{[MASK]}).$$

Consequently, we obtain the embedding of the masked input as $\boldsymbol{X}^{\text{masked}}$. The objective of our first pretext task is to learn a generator $G : \mathbb{R}^{M\times T} \to \mathbb{R}^{K\times T}$ parameterized by $\boldsymbol{\theta}_G$, which minimizes the MSE between the generated values and the true values of the masked positions. Formally, the MSE Loss is defined as:

$$\mathcal{L}_1(\boldsymbol{\theta}_G; \mathcal{D}_N) = \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}_N} \sum_{(k,j)\in\boldsymbol{m}} (\boldsymbol{v}_{kj} - \hat{\boldsymbol{v}}_{kj})^2, \quad (1)$$

where $\hat{v}_{kj}$ represents the generated value at position (k,j) in $G(\boldsymbol{X}^{\text{masked}}; \boldsymbol{\theta}_G)$. The generator G primarily consists of the aforementioned Transformer network $\boldsymbol{f}_G(\cdot)$ producing the hidden representations $\boldsymbol{H}_G = \boldsymbol{f}_G(\boldsymbol{X}^{\text{masked}})$, and a linear layer $\boldsymbol{g}_G(\cdot)$ reconstructing the values $\hat{\boldsymbol{v}} = \boldsymbol{g}_G(\boldsymbol{H}_G) \in \mathbb{R}^{K\times T}$. $\boldsymbol{\theta}_G$ encompasses all the parameters of networks $\boldsymbol{f}_G$ and $\boldsymbol{g}_G$.

Pretext task 2: Classifying value replacements. Inspired by the sample-efficient training method ELECTRA (Clark et al., 2020), which has been shown to outperform MLM in NLP, we introduce our second pretext task as a classification task to determine whether each token in a corrupted input has been replaced by a sampled value from the generator or not. This approach enhances the efficiency of pre-training as it operates on all tokens rather than just a small set of masked tokens as in MLM.

Given the original input $\boldsymbol{x}$ and the randomly masked positions $\boldsymbol{m}$, the corrupted input is obtained by replacing the masked values in the input triplets with the predicted values generated by G:

$$\boldsymbol{x}^{\text{corrupted}} = \text{REPLACE}(\boldsymbol{x}, \boldsymbol{m}, \hat{\boldsymbol{v}}).$$

The corresponding embedding is denoted as $\boldsymbol{X}^{\text{corrupted}}$. The key idea behind ELECTRA is to train an additional discriminator $D : \mathbb{R}^{M\times T} \to \{\text{True}, \text{Sampled}\}^{K\times T}$ parameterized by $\boldsymbol{\theta}_G$ that classifies whether each token in $\boldsymbol{x}^{\text{corrupted}}$ has been replaced by the generator G or not. This is achieved by minimizing the cross-entropy (CE) loss defined as follows:

$$\mathcal{L}_{\text{Disc}}(\boldsymbol{\theta}_D; \mathcal{D}_N) = \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}_N} \sum_{(k,j)} \left[-\mathbb{1}(\hat{\boldsymbol{v}}_{kj} = \boldsymbol{v}_{kj}) \cdot \log \boldsymbol{p}_{kj} - \mathbb{1}(\hat{\boldsymbol{v}}_{kj} \neq \boldsymbol{v}_{kj}) \cdot \log(1 - \boldsymbol{p}_{kj}) \right],$$

where $\boldsymbol{p}_{kj}$ represents the predicted probability of a token being replaced at position (k,j) in $D(\boldsymbol{X}^{\text{corrupted}}; \boldsymbol{\theta}_D)$. The discriminator D primarily consists of another Transformer network $\boldsymbol{f}_D(\cdot)$

that generates the hidden representations $\boldsymbol{H}_D = \boldsymbol{f}_D(\boldsymbol{X}^{\text{corrupted}})$, and a linear layer $\boldsymbol{g}_D(\cdot)$ followed by a sigmoid function, which predicts the probabilities of a token being replaced as $\boldsymbol{p} = \text{Sigmoid}\big(\boldsymbol{g}_D(\boldsymbol{H}_D)\big) \in [0,1]^{K\times T}$. The parameters $\boldsymbol{\theta}_D$ encompass all the parameters of networks $\boldsymbol{f}_D$ and $\boldsymbol{g}_D$. Therefore, the objective of the second pretext task is to minimize the combined loss:

$$\mathcal{L}_2(\boldsymbol{\theta}; \mathcal{D}_N) = \mathcal{L}_1(\boldsymbol{\theta}_G; \mathcal{D}_N) + \lambda \mathcal{L}_{\text{Disc}}(\boldsymbol{\theta}_D; \mathcal{D}_N), \quad (2)$$

where parameters $\boldsymbol{\theta} = \{\boldsymbol{\theta}_G, \boldsymbol{\theta}_D\}$. In practice, generator G is typically a smaller network than discriminator D. After pre-training, we discard the generator and only fine-tune the encoder of the discriminator $\boldsymbol{f}_D$ on downstream tasks.

Pretext task 3: Forecasting time to event. We have designed two pretext tasks that work exclusively with multivariate time series data. To leverage event data, such as clinical events like lab tests or medication prescriptions, which often provide rich information regarding a patient's status change, we propose the third pretext task. A pre-trained network that accurately predicts these clinical events can potentially facilitate downstream benchmark tasks such as mortality and disease risks.

To model the event data, we utilize the Transformer Hawkes Process (THP) (Zuo et al., 2020), which assumes that an event sequence is sampled from a Hawkes Process where the occurrence of a current event depends on all the past events. Given an event sequence observed up to time t_s: $\boldsymbol{e}_{1:t_s} = (\boldsymbol{e}_{s'}, \emptyset, t_{s'})|_{s'=1}^{s}$, we obtain the event embedding $\boldsymbol{E}_{1:t_s}$. The objective of our third task is to learn a Hawkes Process encoder $E : \mathbb{R}^{M\times T} \times \mathbb{R}^+ \to \mathbb{R}^{+K\times 1}$ parameterized by $\boldsymbol{\theta}_E$ that outputs intensity functions $\boldsymbol{\lambda}_k(t|\boldsymbol{e}_{1:t_s})$ for each event type k at $t > t_s$. The conditional density of the time to the next event given the past events $\boldsymbol{e}_{1:t_s}$ is defined as $p(t|\boldsymbol{e}_{1:t_s}) = \boldsymbol{\lambda}(t|\boldsymbol{e}_{1:t_s}) \exp\big(-\int_{t_s}^{t} \boldsymbol{\lambda}(\tau|\boldsymbol{e}_{1:t_s}) d\tau\big)$, where the overall intensity function

$$\boldsymbol{\lambda} = \sum_k \boldsymbol{\lambda}_k(t|\boldsymbol{e}_{1:t_s}) \text{ for } t > t_s.$$

We minimize the negative log-likelihood (NLL) loss:

$$\mathcal{L}_{\text{NLL}}(\boldsymbol{\theta}_E; \mathcal{D}_N) = \mathbb{E}_{\boldsymbol{e}\sim\mathcal{D}_n}\Big[-\sum_{s=1}^{S} \log \boldsymbol{\lambda}(t_s|\boldsymbol{e}_{1:t_s}) + \int_{t_1}^{t_S} \boldsymbol{\lambda}(t|\boldsymbol{e}_{1:t})dt\Big], \quad (3)$$

where the second integration term is approximated by Monte Carlo integration, for which the details can be found in the paper (Zuo et al., 2020). In order to output $\boldsymbol{\lambda}$, the Hawkes process encoder E primarily consists of a Transformer network $\boldsymbol{f}_E(\cdot)$ that outputs the hidden representation $\boldsymbol{H}_E = \boldsymbol{f}_E(\boldsymbol{E})$ and a K-head linear layer $\boldsymbol{g}E(\cdot)$ with a softplus function. This layer estimates the conditional intensity functions $[\boldsymbol{\lambda}k(t|\boldsymbol{e}_{1:t_s})]_{k=1}^{K} = \text{Softplus}\big(\boldsymbol{g}_E([\boldsymbol{H}_E[t_s,:], t])\big)$. The parameters $\boldsymbol{\theta}_D$ encompass all the parameters of networks $\boldsymbol{f}_D$ and $\boldsymbol{g}_D$.

The complete objective of our third pretext task is to minimize the combined loss:

$$\mathcal{L}_3(\boldsymbol{\theta}; \mathcal{D}_N) = \mathcal{L}_2(\boldsymbol{\theta}_G, \boldsymbol{\theta}_D; \mathcal{D}_N) + \psi \mathcal{L}_{\text{NLL}}(\boldsymbol{\theta}_E; \mathcal{D}_N), \quad (4)$$

where the parameters $\boldsymbol{\theta}$ consist of $\boldsymbol{\theta}_G$, $\boldsymbol{\theta}_D$, and $\boldsymbol{\theta}_E$. After pre-training, we take the encoders $\boldsymbol{f}_D$ and $\boldsymbol{f}_E$ and fine-tune them on downstream tasks. In these tasks, the row vectors of the two hidden representations, $\boldsymbol{H}_D$ and $\boldsymbol{H}_E$, are concatenated to form the final representation vector.

3.4. Supervised Fine-Tuning

Given a Transformer encoder $\boldsymbol{f}(\cdot)$ that was pre-trained on unlabeled data $\mathcal{D}_N$ for a pretext task, we fine-tune it through supervised learning on the downstream dataset $\mathcal{D}_{N'}$ with labels $\boldsymbol{y}_{N'}$. We then train an additional multilayer perceptron (MLP) with parameters $\boldsymbol{\theta}_{\text{MLP}}$ to produce the final predictions on the labels. To fine-tune our model, we minimize the CE loss as:

$$\min_{\boldsymbol{\theta}} \mathbb{E}_{(\boldsymbol{x},\boldsymbol{e},\boldsymbol{x}_0)\in\mathcal{D}_{N'}} \text{CELoss}(\hat{y}, y), \quad (5)$$

where y is the true label for the input data triplet $(\boldsymbol{x}, \boldsymbol{e}, \boldsymbol{x}_0)$, and $\hat{y} = \text{Sigmoid}\big(\text{MLP}([\boldsymbol{h}, \boldsymbol{x}_0]; \boldsymbol{\theta}_{\text{MLP}})\big)$. Here, $\boldsymbol{h} \in \mathbb{R}^M$ is the concatenation of the row vectors of the hidden representation $\boldsymbol{H}$ output by $\boldsymbol{f}$. The parameters $\boldsymbol{\theta}$ include the parameters of $\boldsymbol{f}$ and $\boldsymbol{\theta}_{\text{MLP}}$.

4. Experiments

We demonstrate the effectiveness of our model using three real-world healthcare datasets collected from different Intensive Care Units (ICUs). We compare `TransEHR` to baseline methods on three benchmark tasks: predicting in-hospital mortality, length-of-stay, and acute care phenotypes.

4.1. Datasets

Physionet-12 (Goldberger et al., 2000): This dataset is derived from the Physionet Challenge 2012. It comprises 12,000 adult ICU stays with time series data on 36 covariates and events (excluding '*MechVet*' here). The objective of the challenge is to predict in-hospital mortality based on the first 48 hours of data for each stay. We follow the setup outlined in Horn et al. (2020)'s by merging all three sets (set-a, set-b, and set-c) and performing a $64:16:20$ train-validation-test split.

AmsterdamUMCdb (Thoral et al., 2021): This dataset consists of 23,106 ICU stays collected from 20,109 adult patients at the Amsterdam University Medical Center. We extract the same set of variables as in Physionet-12, except we zero-pad the '*GCS*' variable because it is missing in this data set. We exclude ICU stays with a length-of-stay of less than 48 hours or no data available within the first 48 hours. This results in a total of 7,484 time series, and we perform a $64:16:20$ train-validation-test split.

MIMIC-III (Johnson et al., 2016): This publicly available database contains 61,532 adult ICU stays from 46,476 patients, collected at the Beth Israel Deaconess Medical Center between 2001 and 2012. We preprocess the data using the code repository provided by Johnson et al. (2018) and extract the same set of variables as in Physionet-12. To create the benchmark tasks, we follow the procedure described in Harutyunyan et al. (2019)'s, obtaining 41,902 stays after excluding pediatric patients and the records with unknown length-of-stay. We perform an $85:15$ train-test split on the entire dataset, followed by an $80:20$ train-validation split on the training set. Furthermore, we also prepare the MIMIC-III data by including only the first 48 hours of each stay. We exclude ICU stays with unknown or less than 48 hours of length-of-stay, as well as those with no available data within the first 48 hours. This results in a training set of 17,903 ICU stays and a test set of 3,236 ICU stays.

All datasets consist of 36 variables, including 8 vital signs (e.g., diastolic blood pressure, heart rate, temperature, etc.) and 28 lab test results. We use all the variables to construct $K = 36$ multivariate time series and utilize the lab events to form $D = 28$ event sequences. Additionally, we include two time-invariant static features: age and gender. Details of data preprocessing are in Appendix C.

4.2. Benchmark Tasks

We evaluate three prediction tasks using the aforementioned datasets. We use the first 48 hours of each ICU stay for in-hospital mortality and length-of-stay, whereas the complete duration for phenotyping.
In-hospital mortality: This task involves binary classification, with a prevalence of approximately 13% to 14% across all three datasets. We evaluate the performance of this task using the metrics of macro-average AU-ROC and AU-PRC.
Length-of-stay (LOS): For this task, we approach it as both a regression task and a multi-class classification task by categorizing the real-valued LOS into ten discrete buckets (for detailed information, refer to Harutyunyan et al. (2019)'s). We assess the performance using the mean absolute error (MAE) and Cohen's linear weighted kappa (Cohen, 1960).
Phenotyping: The objective of this task is to retrospectively classify patients into 25 acute care phenotypes (Harutyunyan et al., 2019). Since 99% of patients in the data have multiple phenotype labels, this task is considered a multi-label classification problem. We evaluate this task exclusively on the MIMIC-III dataset, as the other two datasets lack the necessary labels. The evaluation metrics used for this task are Micro/Macro-avg AU-ROC.

4.3. Baselines

We compare our proposed `TransEHR` with the current state-of-the-art methods developed for modeling EHR time series data. The majority of these methods are supervised models, with the exception of STraTS (Tipirneni and Reddy, 2022), which is a self-supervised Transformer model. It is worth noting that all of these methods exclusively work with multivariate time series data and disregard the information contained in event timestamps.

- **(C-)LSTM** (Harutyunyan et al., 2019): This method utilizes a standard (channel-wise) Long Short-Term Memory (LSTM) network, which is fitted to a binned multivariate input matrix.
- **GRU-D** (Che et al., 2018): This approach extends the Gated Recurrent Unit (GRU) by incorporating a decay term, which is employed for imputing missing values in multivariate time series.
- **SeFT** (Horn et al., 2020): SeFT employs triplets to represent the irregular time series data, applies set functions to encode these triplets, and utilizes a GRU network to model the temporal dynamics.
- **Transformer** (Vaswani et al., 2017): In this

Table 1: Experimental results for mortality prediction on the three datasets: 'Self' pre-training procedure refers to the model being pre-trained using the same dataset, while 'Transferred' pre-training procedure refers to the model being pre-trained using the other two distinct datasets.

Model	Pre-train Procedure	Physionet-12		MIMIC-III		AmsterdamUMCdb	
		AU-ROC	AU-PRC	AU-ROC	AU-PRC	AU-ROC	AU-PRC
SeFT	-	85.1 ± 0.4	52.4 ± 1.1	83.9 ± 0.4	46.3 ± 0.5	-	-
GRU-D	-	86.3 ± 0.3	53.7 ± 0.9	85.7 ± 0.2	52.0 ± 0.8	-	-
Transformer	-	86.3 ± 0.8	52.8 ± 2.2	86.5 ± 0.8	51.7 ± 0.8	80.5 ± 0.2	50.7 ± 1.3
STraTS	Self	86.5 ± 0.5	$\mathbf{54.0 \pm 0.4}$	86.7 ± 0.6	50.5 ± 0.6	81.0 ± 0.2	50.6 ± 0.3
`TransEHR`	Self	86.7 ± 0.2	52.9 ± 1.2	87.6 ± 0.3	$\mathbf{53.5 \pm 0.6}$	80.9 ± 0.3	51.1 ± 0.5
	Transferred	$\mathbf{87.1 \pm 0.1}$	53.8 ± 0.6	$\mathbf{87.8 \pm 0.5}$	52.8 ± 1.3	$\mathbf{81.5 \pm 0.1}$	$\mathbf{53.4 \pm 0.8}$

Table 2: Experimental results of length-of-stay prediction on the three datasets.

Model	Pre-train Procedure	Physionet-12		MIMIC-III		AmsterdamUMCdb	
		Kappa↑	MAE↓	Kappa↑	MAE↓	Kappa↑	MAE↓
Transformer	-	32.8 ± 0.3	6.56 ± 0.6	39.9 ± 1.3	3.15 ± 0.09	31.4 ± 0.4	7.16 ± 0.04
STraTS	Self	32.6 ± 1.0	7.36 ± 1.4	41.6 ± 0.2	4.09 ± 0.14	31.2 ± 0.9	9.01 ± 0.17
`TransEHR`	Self	$\mathbf{34.9 \pm 0.1}$	6.43 ± 0.5	41.9 ± 0.4	2.98 ± 0.08	31.7 ± 1.0	7.04 ± 0.02
	Transferred	34.0 ± 0.3	$\mathbf{6.42 \pm 0.2}$	$\mathbf{42.3 \pm 0.2}$	$\mathbf{2.91 \pm 0.06}$	$\mathbf{31.9 \pm 0.5}$	$\mathbf{6.97 \pm 0.02}$

method, the GRU network from SeFT is replaced with a Transformer network to capture the temporal dynamics.

• **STraTS** (Tipirneni and Reddy, 2022): STraTS is a self-supervised Transformer model that undergoes pre-training through a time-series forecasting task.

4.4. Pre-training and fine-tuning Setup

We conducted two sets of experiments for each task, involving pre-training and fine-tuning stages. In the first group of experiments, the model was pre-trained on two datasets without any revealed labels. Subsequently, it was fine-tuned using the training and validation sets of the third dataset, and evaluated on its corresponding test set. This procedure aimed to train the model to acquire robust representations capable of transfer learning across different source domains. In our results, we refer to this procedure as 'transferred' pre-training.

In the second group of experiments, the model was pre-trained on each dataset's own training set, again without any revealed labels. It was then fine-tuned using the labeled training and validation sets, and evaluated on the respective test set. This procedure aimed to train the model to learn generalized representations, thereby mitigating the risk of overfitting to unbalanced classes in the supervised setting. In our results, we denote this procedure as 'self' pre-training. Implementation details including the training hyper-parameters are described in Appendix D.

Table 3: Experimental results of phenotype multi-label classification on MIMIC-III dataset.

Model	Pre-train Procedure	MIMIC-III AU-ROC	
		Macro-avg	Micro-avg
LSTM	-	76.8 ± 0.4	81.8 ± 0.4
C-LSTM	-	77.4 ± 0.4	82.3 ± 0.4
Transformer	-	80.9 ± 0.1	85.3 ± 0.2
STraTS	Self	80.7 ± 0.2	84.8 ± 0.3
`TransEHR`	Self	81.2 ± 0.2	85.6 ± 0.2
	Transferred	$\mathbf{81.6 \pm 0.1}$	$\mathbf{85.8 \pm 0.1}$

4.5. Results

Mortality prediction. Table 1 presents the experimental results for mortality prediction on the three datasets. We include results of SeFT and GRU-D from their respective papers for comparison. Our best-performing model, referred to as `TransEHR` throughout the paper, is pre-trained using Pretext task 3 (details refer to Section 4.5.1). From the table, we observe that `TransEHR`, pre-trained on the two distinct datasets (referred to as 'Transferred' pre-train), outperforms all other methods consistently. Notably, on MIMIC-III and AmsterdamUMCdb, `TransEHR` demonstrates significant improvements compared to the strongest baseline, STraTS, achieving an approximate 3% increase in AU-PRC. This is particularly valuable for an imbalanced binary classification task such as mortality prediction, with a prevalence of approximately 14%.

Length-of-stay prediction. Table 2 shows the results for LOS prediction. For the LOS multi-class classification task, TransEHR outperforms STraTS across all datasets, achieving an average increase of approximately 1% in Kappa score. In the LOS regression task, TransEHR surpasses STraTS with a significant reduction of ∼1.5 days in MAE on average.

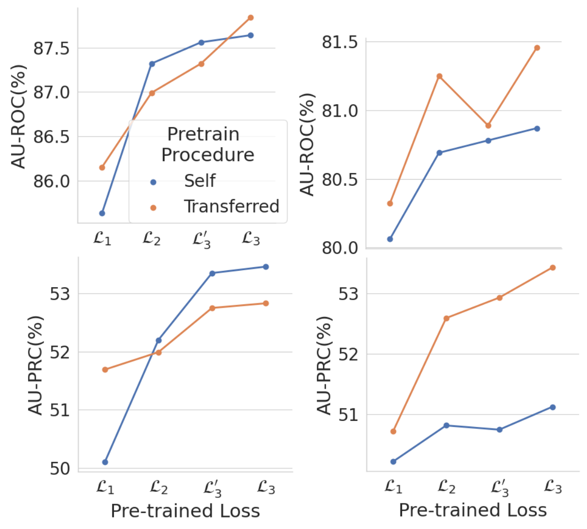
Phenotype prediction. Table 3 presents the results of the MIMIC-III phenotype prediction task. Our model, TransEHR, achieves superior performance compared to the LSTM baselines reported in Harutyunyan et al. (2019) as well as STraTS. Specifically, TransEHR achieves a 4.2% higher Macro-avg AU-ROC (3.5% higher micro-avg AU-ROC) compared to the LSTM baselines. TransEHR with the 'Transferred' pre-training procedure achieves ∼1% increase in AU-ROC compared to the strongest baseline, STraTS.

Based on these results, we observe that the self-pretrained TransEHR outperforms the supervised Transformer, indicating that unsupervised pre-training helps regulate overfitting in imbalanced or multi-label/multi-class supervised learning. Furthermore, the evidence that self-pretrained TransEHR outperforms STraTS highlights the effectiveness of the proposed pretext task 3 for forecasting event arrival times compared to forecasting multivariate values. Lastly, the transferred pre-trained TransEHR performs the best across all tasks and datasets, demonstrating the advancements achieved by leveraging unlabeled data from other sources.

4.5.1. Ablation study

Pretext tasks for pre-training. We investigate the three pre-training tasks defined in Section 3.3 for the downstream task of mortality prediction. We pre-train TransEHR by minimizing the losses $\mathcal{L}_1$, $\mathcal{L}_2$, $\mathcal{L}_3$ as defined in Equations (1), (2), and (4) respectively. We also consider a variant $\mathcal{L}_3'$ by substituting $\mathcal{L}_2$ with $\mathcal{L}_1$ in Equation (4). The results are depicted in Figure 4.5.1. It is noteworthy that incorporating a discriminator D to formulate a value replacement classification pretext task leads to an improvement in downstream task performance. Furthermore, the inclusion of the Hawkes process encoder E for time-to-event forecasting further enhances the overall performance. Thus, we identify pretext task 3 as the most effective for downstream classifications on EHR data.

Sensitivity to data availability. To assess the fine-tuning efficiency of the pre-trained TransEHR compared to the baseline models STraTS and Transformer, we vary the availability of labeled data from 10% to 50% for the downstream mortality prediction task



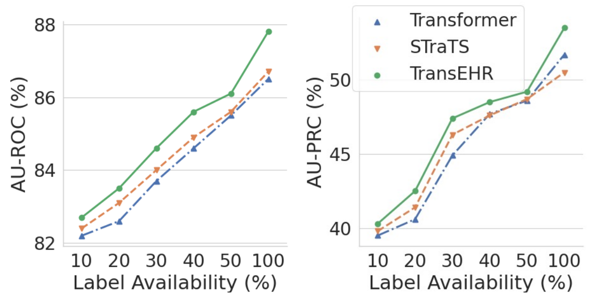
(a) MIMIC-III (b) AmsterdamUMCdb

Figure 2: Mortality prediction performance of TransEHR on the two datasets when pre-trained to minimize the different pretext losses ($\mathcal{L}_3'$ is a variant of $\mathcal{L}_3$ by substituting the $\mathcal{L}_2$ with $\mathcal{L}_1$ in Equation (4)).

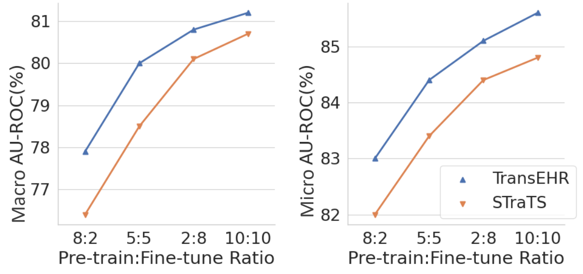
on MIMIC-III. The corresponding performance in terms of AU-ROC and AU-PRC is plotted in Figure 3(a). We observe that the performance gaps between TransEHR, STraTS, and Transformer are maximized between 20% and 40% data availability. Furthermore, when using only 50% of the labeled data, TransEHR achieves a comparable AU-ROC of ∼86.1% and an AU-PRC of ∼49.2%, whereas STraTS and Transformer require the use of the entire labeled dataset.

To investigate the optimal ratio for splitting the training data into a pre-training set (with no revealed labels) and a fine-tuning set, we conduct a further study. Currently, we use the full "unlabeled" training set for pre-training and then add the labels back for fine-tuning, corresponding to a 10:10 ratio as shown in Figure 3(b). We vary the ratio between 2:8, 5:5, and 8:2, and plot the corresponding AU-ROC scores in Figure 3(b). Remarkably, we find that TransEHR achieves comparable results to the supervised Transformer on both AU-ROC scores when using a split ratio of 2:8 and 5:5. This result confirms our earlier findings in Figure 3(a) that TransEHR requires a smaller set of labeled data for fine-tuning to achieve strong prediction performance on downstream tasks.

Fine-tuning on the full model vs. only the MLP layer. Regarding fine-tuning, we investigate the choice between fine-tuning the full pre-trained model and fine-tuning only the last MLP layer for downstream classification. We present the results of fine-tuning 'transferred' pre-trained TransEHR for mortality prediction on the three datasets in Figure 4. We observed a reduction of approximately 6% in AU-ROC and 13% in AU-PRC when fine-tuning only the last MLP layer compared to fine-tuning the full model. It is worth noting that this performance gap could potentially be mitigated with a larger and more sufficient pre-training dataset.



(a) Vary the label availability during MIMIC-III mortality fine-tuning



(b) Vary the pre-training vs. fine-tuning ratio in MIMIC-III phenotype training set

Figure 3: Performance changes when varying the pre-training or fine-tuning data availability.

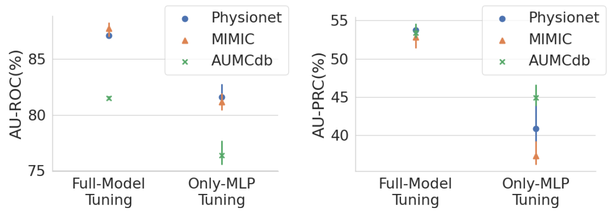


Figure 4: Comparison of fine-tuning the 'transferred' pre-trained TransEHR on the full model vs. only the last MLP layer for mortality prediction. Error bars present the standard deviation across three fine-tuning runs.

5. Discussion and Conclusions

We propose a self-supervised Transformer model and an efficient pre-training pretext task to address two key challenges in classification tasks using modern EHR time series data: 1) the asynchronous collection of event sequences alongside multivariate time series, and 2) the limited availability of labeled data. Our experiments on three real-world health datasets demonstrate the proposed model TransEHR achieves state-of-the-art performance on three benchmark clinical prediction tasks: in-hospital mortality, length-of-stay, and phenotyping.

TransEHR's impact on healthcare is marked by improved accuracy, efficient data utilization, and the advancement of healthcare analytics. Enhanced accuracy in benchmark clinical tasks is crucial for providing timely and accurate medical care. Healthcare professionals can rely on more accurate predictions and insights, ultimately leading to better patient care and resource allocation. The efficient use of data helps in building robust models even when labeled data is scarce, which is often the case in healthcare.

The proposed method has limitations arising from the insufficient availability of public health data, which can impede pre-training performance. Overcoming these limitations necessitates collaborative efforts to enhance data availability, thereby improving the performance and applicability of TransEHR. Additionally, there is potential for extending TransEHR by incorporating recent Transformer-based time series models, such as Liu et al. (2021); Zhou et al. (2021, 2022); Nie et al. (2022); Wu et al. (2022), etc. This integration could enable TransEHR to capitalize on the architectural advancements presented by these models, thereby further optimizing its capabilities. Further model considerations should also be made to accommodate transfer learning when leveraging and combining diverse data sources or encompass a broader range of data modalities, such as electrocardiogram (ECG) signals (Raghu et al., 2023).

Acknowledgements

This material is based upon work supported by the National Science Foundation under Grant Numbers NSF IIS-2106961, CAREER IIS-2144338, and CCF-2029004. **Disclaimer:** Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

References

Zhengping Che, Sanjay Purushotham, Kyunghyun Cho, David Sontag, and Yan Liu. Recurrent neural networks for multivariate time series with missing values. *Scientific reports*, 8(1):6085, 2018.

Kevin Clark, Minh-Thang Luong, Quoc V Le, and Christopher D Manning. Electra: Pre-training text encoders as discriminators rather than generators. *arXiv preprint arXiv:2003.10555*, 2020.

Jacob Cohen. A coefficient of agreement for nominal scales. *Educational and psychological measurement*, 20(1):37–46, 1960.

Jiaxiang Dong, Haixu Wu, Haoran Zhang, Li Zhang, Jianmin Wang, and Mingsheng Long. Simmtm: A simple pre-training framework for masked time-series modeling. *arXiv preprint arXiv:2302.00861*, 2023.

Nan Du, Hanjun Dai, Rakshit Trivedi, Utkarsh Upadhyay, Manuel Gomez-Rodriguez, and Le Song. Recurrent marked temporal point processes: Embedding event history to vector. In *KDD*, pages 1555–1564. ACM, 2016.

Ary L Goldberger, Luis AN Amaral, Leon Glass, Jeffrey M Hausdorff, Plamen Ch Ivanov, Roger G Mark, Joseph E Mietus, George B Moody, Chung-Kang Peng, and H Eugene Stanley. Physiobank, physiotoolkit, and physionet: components of a new research resource for complex physiologic signals. *circulation*, 101(23):e215–e220, 2000.

Hrayr Harutyunyan, Hrant Khachatrian, David C Kale, Greg Ver Steeg, and Aram Galstyan. Multitask learning and benchmarking with clinical time series data. *Scientific data*, 6(1):96, 2019.

Alan G Hawkes. Spectra of some self-exciting and mutually exciting point processes. *Biometrika*, 58 (1):83–90, 1971.

Max Horn, Michael Moor, Christian Bock, Bastian Rieck, and Karsten Borgwardt. Set functions for time series. In *International Conference on Machine Learning*, pages 4353–4363. PMLR, 2020.

Shayan Jawed, Josif Grabocka, and Lars Schmidt-Thieme. Self-supervised learning for semi-supervised time series classification. In *Advances in Knowledge Discovery and Data Mining: 24th Pacific-Asia Conference, PAKDD 2020, Singapore, May 11–14, 2020, Proceedings, Part I 24*, pages 499–511. Springer, 2020.

Alistair E W Johnson, David J Stone, Leo A Celi, and Tom J Pollard. The mimic code repository: enabling reproducibility in critical care research. *Journal of the American Medical Informatics Association*, 25 (1):32–39, 2018.

Alistair EW Johnson, Tom J Pollard, Lu Shen, Li-wei H Lehman, Mengling Feng, Mohammad Ghassemi, Benjamin Moody, Peter Szolovits, Leo Anthony Celi, and Roger G Mark. Mimic-iii, a freely accessible critical care database. *Scientific data*, 3 (1):1–9, 2016.

Jacob Devlin Ming-Wei Chang Kenton and Lee Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of naacL-HLT*, pages 4171–4186, 2019.

Yikuan Li, Shishir Rao, José Roberto Ayala Solares, Abdelaali Hassaine, Rema Ramakrishnan, Dexter Canoy, Yajie Zhu, Kazem Rahimi, and Gholamreza Salimi-Khorshidi. Behrt: transformer for electronic health records. *Scientific reports*, 10(1):1–12, 2020.

Shizhan Liu, Hang Yu, Cong Liao, Jianguo Li, Weiyao Lin, Alex X Liu, and Schahram Dustdar. Pyraformer: Low-complexity pyramidal attention for long-range time series modeling and forecasting. In *International conference on learning representations*, 2021.

Hongyuan Mei and Jason Eisner. The neural hawkes process: A neurally self-modulating multivariate point process. In *NIPS*, pages 6754–6764, 2017.

Yuqi Nie, Nam H Nguyen, Phanwadee Sinthong, and Jayant Kalagnanam. A time series is worth 64 words: Long-term forecasting with transformers. *arXiv preprint arXiv:2211.14730*, 2022.

Aniruddh Raghu, Payal Chandak, Ridwan Alam, John Guttag, and Collin M. Stultz. Sequential multi-dimensional self-supervised learning for clinical time series. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org, 2023.

Satya Narayan Shukla and Benjamin M Marlin. Interpolation-prediction networks for irregularly sampled time series. *arXiv preprint arXiv:1909.07782*, 2019.

Patrick J Thoral, Jan M Peppink, Ronald H Driessen, Eric JG Sijbrands, Erwin JO Kompanje, Lewis Kaplan, Heatherlee Bailey, Jozef Kesecioglu, Maurizio Cecconi, Matthew Churpek, et al. Sharing icu patient data responsibly under the society of critical care medicine/european society of intensive care medicine joint data science collaboration: the amsterdam university medical centers database (amsterdamumcdb) example. *Critical care medicine*, 49 (6):e563, 2021.

Sindhu Tipirneni and Chandan K Reddy. Self-supervised transformer for sparse and irregularly sampled multivariate clinical time-series. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 16(6):1–17, 2022.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, pages 1096–1103, 2008.

Haixu Wu, Tengge Hu, Yong Liu, Hang Zhou, Jianmin Wang, and Mingsheng Long. Timesnet: Temporal 2d-variation modeling for general time series analysis. *arXiv preprint arXiv:2210.02186*, 2022.

Yanbo Xu, Siddharth Biswal, Shriprasad R Deshpande, Kevin O Maher, and Jimeng Sun. Raim: Recurrent attentive and intensive model of multimodal patient monitoring data. In *Proceedings of the 24th ACM SIGKDD international conference on Knowledge Discovery & Data Mining*, pages 2565–2573, 2018.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. Xlnet: Generalized autoregressive pretraining for language understanding. *Advances in neural information processing systems*, 32, 2019.

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*, pages 2114–2124, 2021.

Qiang Zhang, Aldo Lipani, Ömer Kirnap, and Emine Yilmaz. Self-attentive hawkes process. In *ICML*, volume 119 of *Proceedings of Machine Learning Research*, pages 11183–11193. PMLR, 2020.

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. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 11106–11115, 2021.

Tian Zhou, Ziqing Ma, Qingsong Wen, Xue Wang, Liang Sun, and Rong Jin. Fedformer: Frequency enhanced decomposed transformer for long-term series forecasting. In *International Conference on Machine Learning*, pages 27268–27286. PMLR, 2022.

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

Simiao Zuo, Haoming Jiang, Zichong Li, Tuo Zhao, and Hongyuan Zha. Transformer hawkes process. In *ICML*, volume 119 of *Proceedings of Machine Learning Research*, pages 11692–11702. PMLR, 2020.

Appendix

Appendix A. Notation Table

Table 4 summarizes the notations in this paper .

Appendix B. Model architecture

After the embedding layers, we obtain $\boldsymbol{X} = [\boldsymbol{X}_1, \cdots, \boldsymbol{X}_T] \in \mathbb{R}^{M \times T}$ for a multivariate time series input and $\boldsymbol{E} = [\boldsymbol{E}_1, \cdots, \boldsymbol{E}_S] \in \mathbb{R}^{M \times S}$ for an event sequence input. We pass them through two separate self-attention modules that share the same architecture. We compute an H-head attention output $\boldsymbol{A}$ for the input $\boldsymbol{X}$ as follows:

$$\mathbf{A}_h = \text{Softmax}\left(\frac{\boldsymbol{X}\boldsymbol{W}_h^Q(\boldsymbol{X}\boldsymbol{W}_h^K)^\top}{\sqrt{M_K}}\right)\boldsymbol{X}\boldsymbol{W}_h^V,$$
$$\mathbf{A} = \text{Concat}(\mathbf{A}_1, \cdots, \mathbf{A}_H)\mathbf{W}^O,$$

where matrices $\mathbf{W}_h^Q, \mathbf{W}_h^K \in \mathbb{R}^{M \times M_K}$ and $\mathbf{W}_h^V \in \mathbb{R}^{M \times M_V}$ are the query, key, and value projections, respectively. The matrix $\mathbf{W}^O \in \mathbb{R}^{(M_V \times H) \times M}$ aggregates the final attention outputs. Finally, $\boldsymbol{A}$ is fed through a position-wise feed-forward neural network (FFN) to obtain the final hidden representations $\boldsymbol{H} \in \mathbb{R}^{T \times M}$ as follows:

$$\boldsymbol{H} = \text{FFN}(\mathbf{A}) = \max(0, \mathbf{A}\mathbf{W}_1^{FC} + \mathbf{b_1})\mathbf{W}_2^{FC} + \mathbf{b_2},$$

where $\mathbf{W_1^{FC}}, \mathbf{W_2^{FC}} \in \mathbb{R}^{M \times M_K}$, $\mathbf{b_1} \in \mathbb{R}^{M_K}$, and $\mathbf{b_2} \in \mathbb{R}^{M}$ are the parameters of the neural network. The j-th row in the matrix $\boldsymbol{H}$ encodes the time series up to time t_j.

Similarly, we obtain a hidden representation for the event sequences $\boldsymbol{E}$ with a dimension of $S \times M$, where the s-th row of the representation matrix encodes all the events up to time t_s. When computing the s-th row of the attention output $\boldsymbol{A}_h$, we need to mask future positions in the matrix $\boldsymbol{W}_h^Q$ to prevent assigning dependencies to future events. This is necessary because the Hawkes process assumes that current events depend only on the past history but not future events.

Appendix C. Data description

We present the detailed statistics of our dataset in Table 5. The list of covariates contains demographic features of Age and Gender, vital signs of Diastolic blood pressure, Glucose, Heart Rate, Mean blood pressure, Systolic blood pressure, Temperature, Respiratory rate, and Oxygen saturation, events of Albumin, Alkaline phosphate, Alanine aminotransferase, Asparate aminotransferase, Bilirubin, Blood urea nitrogen, Cholesterol, Creatinine, Fraction inspired oxygen, Glascow coma scale total, Bicarbonate, Hematocrit, Potassium, Lactate, Magnesium, Sodium, Diastolic blood pressure (noninvasive), Mean blood pressure (noninvasive), Systolic blood pressure (noninvasive), Partial pressure of carbon dioxide, Partial pressure of oxygen, pH, Platelets, O2 saturation in hemoglobin, Troponin-I, Troponin-T, Urine output, and White blood cell count. Data preprocessing scripts are available in our code repo.

Appendix D. Implementation details

All models are implemented using PyTorch. We referred to the implementation from Zuo et al. (2020) and Zerveas et al. (2021) when building our time series encoder and event series encoder. We set the input embedding dimension, M, to 256. The Transformer encoder consists of one encoding layer with 256 feed-forward variables and 2 attention heads, and the size of the linear output layer is 256. Specifically for pretext task 2, the generator D is configured to be one-fourth of the hidden size. For downstream tasks, we added a one-layer MLP with a hidden size of 64 on top of the encoders.

For pre-training, we chose the Adam optimizer with a learning rate of $2e-3$, a decay of 0.5 every 50 epochs, and a maximum of 2000 training epochs. For downstream tasks, we used Adam with a learning rate of $2e-4$ and an exponential decay of 0.8 every 20 epochs. The training process would be stopped either when it reaches the maximum epoch number of 500 or when the validation loss stops decreasing for 30 epochs. All experiments were conducted using a single NVIDIA RTX A5000 GPU.

Table 4: List of notations

Multivariate Notations	
t_j	Sample time of the j-th multivariate triplet
$\boldsymbol{v}_j^{(i)}$	Value vector of the j-th triplet of patient i. Each dimension $[\boldsymbol{v}_j]_k$ correspond to the k-th variable's sampled value at t_j (or ϕ if the value is missing)
$\boldsymbol{z}_j^{(i)}$	Multivariate indicator vector of the j-th triplet of patient i. Each dimension $[\boldsymbol{z}_j]_k$ is 1 if the k-th variable is observed (that is, $[\boldsymbol{v}_j^{(i)}]_k \neq \phi$ at time t_j, and 0 otherwise.
Event Notations	
$t_s^{(i)}$	Sample time of the s-th event triplet from patient i
$\boldsymbol{e}_s^{(i)}$	Event indicator vector of the s-th event triplet from patient i. Each dimension $[\boldsymbol{e}_s^{(i)}]_d$ is 1 if the $d-th$ event happens at t_s, or 0 otherwise.
General Data Notations	
$\mathcal{D}_N$	Set of observed EHR time series, composed of both multivariate triplets and event triplets
$\boldsymbol{X}$	Series of multivariate time series input
$\boldsymbol{E}$	Series of event sequence input
$\boldsymbol{y}^{(i)}$	Label of patient i
$\boldsymbol{x}_0^(i)$	Statistical demographics of patient i
Embedding Notations	
$\boldsymbol{u}(t_j)$	Temporal encoding of time t_j
$\boldsymbol{U}_K$	Embedding matrix for multivariate indicator vector $\boldsymbol{z}$
$\boldsymbol{c}$	Embedding bias vector for multivariate indicator vector $\boldsymbol{z}$
$\boldsymbol{U}_D$	Embedding matrix for event indicator vector $\boldsymbol{e}$
$\boldsymbol{W}_v$	Embedding matrix for multivariate value vector $\boldsymbol{v}$
$\boldsymbol{b}_v$	Embedding bias vector for multivariate value vector $\boldsymbol{v}$
Training Variables Notations	
$\boldsymbol{m}$	Randomly generated mask matrix
$\boldsymbol{\theta}_G$	Parameters of a Transformer-based sequence-to-sequence generator
$\boldsymbol{\theta}_D$	Parameters of a Transformer-based discriminator
$\boldsymbol{theta}_E$	Parameters of a Transformer-Hawkes event encoder
$\boldsymbol{\theta}_{\text{MLP}}$	Parameters of a MLP classifier.

Table 5: Details of datasets

	MIMIC-III	Physionet-2012	AmsterdamUMCdb
ICU stays	61,532	12,000	23,106
48h LOS volume	21,139	11,988	7484
Raw signs	154	37	1676
Max Length-of-Stay (days)	99	295	237