Point-process based representation learning for Electronic Health Records

Hojjat Karami, *Fellow, IEEE*, Anisoara Ionescu, and David Atienza, *Member, IEEE*

***Abstract*— Irregular sampling of time series in electronic health records (EHRs) presents a challenge for model development. In addition, the pattern of missingness for certain clinical variables is not random, but rather determined by clinicians' decisions and the patient's state. Point process is a mathematical framework for handling event sequence data that is consistent with irregular sampling. We propose *NPP4EHR*, a deep neural network that can learn patient representations from irregularly sampled time series using the neural point process framework. We conducted various experiments to demonstrate the effectiveness of event and state encoding for characterizing conditional intensity functions, as well as for downstream prediction tasks, using two healthcare databases. Our proposed transformer event encoder outperforms state-of-the-art models in commonly used event sequence databases, as evidenced by its superior performance in negative log-likelihood and next-event prediction. Furthermore, we demonstrated the usefulness of state encoding for improved conditional intensity function characterization, as well as event encoding for better representation learning in downstream tasks, using two real-world healthcare datasets. We propose that our model could be leveraged for better representation learning in EHRs.**

***Index Terms*— electronic health records (EHRs), Point Process, irregular sampling, informative missingness,**

I. Introduction

[ML in healthcare] Machine learning has the potential to revolutionize healthcare by leveraging vast amounts of data available in electronic health records (EHRs) to develop more accurate clinical decision support systems. EHRs store patient health information, such as medical history, medications, lab results, and diagnostic images, which can be used as input for machine learning algorithms to identify patterns and associations that could inform more precise diagnoses, better treatment plans, and earlier interventions. Clinical decision support systems that use machine learning can provide real-time, evidence-based recommendations to healthcare providers, reducing errors and improving patient outcomes.

[irregular sampling challenge] One of the data challenges for machine learning (ML) when using electronic health records (EHRs) is irregular sampling. EHR data is often collected at different times and frequencies, depending on a patient's healthcare needs and visit schedules, which can result in uneven and irregularly sampled time series data. Asynchronous and incomplete observation of certain clinical variables can also be regarded as missingness in the data.

missingness in tabular and time series.

[sources of missingness] The sources of missing data in EHRs must be carefully understood. For instance, lab measurements are usually ordered as part of a diagnostic work-up, and the presence or absence of a data point conveys information about the patient's state [4].

[imputation methods] Imputation techniques are widely used in electronic health records (EHRs) to handle missing data. Imputation refers to the process of filling in the missing data with plausible values based on the available information. There are several imputation techniques available, each with its strengths and weaknesses. For example, mean imputation replaces missing values with the mean of the available values in the same column, while regression imputation uses a regression model to estimate the missing values based on the relationship between the variables. Other popular imputation methods include multiple imputations, hot deck imputation, and k-nearest neighbors imputation. The choice of imputation technique depends on the type and amount of missing data, as well as the goals of the analysis. Regardless of the technique used, it is important to carefully consider the impact of imputed values on the analysis and to report the imputation method used in the results.

There exist several ML frameworks that are inherently compatible with the missing or irregularly-sampled data such as Gaussian process and recurrent neural networks. Deep learning models that can handle irregularly sampled time series are crucial in many real-world applications, including stock market predictions, speech recognition, and medical diagnosis. These models must be able to process data that is not evenly spaced, which is common in time-sensitive applications. One popular deep learning architecture that can handle irregularly sampled time series is the Recurrent Neural Network (RNN). RNNs are well-suited for this task because they can process sequential data, taking into account not only the current input but also the previous inputs. Another architecture that can handle irregularly sampled time series is the Convolutional Neural Network (CNN), which can extract features from the data and then pass the processed data to an RNN for

This paragraph of the first footnote will contain the date on which you submitted your paper for review. It will also contain support information, including sponsor and financial support acknowledgment. For example, "This work was supported by DigiPredict Grant BS123456."

H. Karami is with the EPFL, Lausanne, Switzerland (e-mail: hojjat.karami@epfl.ch).

S. B. Author, Jr., was with Rice University, Houston, TX 77005 USA. He is now with the Department of Physics, Colorado State University, Fort Collins, CO 80523 USA (e-mail: author@lamar.colostate.edu).

T. C. Author is with the Electrical Engineering Department, University of Colorado, Boulder, CO 80309 USA, on leave from the National Research Institute for Metals, Tsukuba, Japan (e-mail: author@nrim.go.jp).

further analysis. Additionally, Attention Mechanisms can be integrated into RNNs and CNNs to help the model focus on important features in the data. These deep learning models offer a powerful toolset for handling irregularly sampled time series and providing useful insights into this type of data. However, it is important to note that these models do not explicitly account for the fact that the absence of some data points can convey unique information, and it is important to carefully consider the sources of missingness in the data.

Point process is the mathematical framework for describing the distribution of events in time or space. These events can be occurrences of earthquakes, nerve impulses, customer purchases, or anything else that can be counted or measured. In healthcare, lab measurements can be regarded as sequence of events that are ordered by clinicians. At the core of point process is the defintion of conditional intensity functions and the corresponding log-likelihood which simultanously models the occurence of events using the history of past events.

Point process is a mathematical framework for describing the distribution of events in time or space. These events can be occurrences of earthquakes, nerve impulses, customer purchases, or anything else that can be counted or measured. In healthcare, lab measurements can be regarded as a sequence of events that are ordered by clinicians. At the core of the point process is the definition of conditional intensity functions and the corresponding log-likelihood, which simultaneously models the occurrence of events using the history of past events. The conditional intensity function represents the instantaneous rate of event occurrence, given the history of past events. This function can be used to predict the probability of future events, which is particularly useful in healthcare applications such as predicting disease onset or detecting adverse events.

More recently, Neural Point Processes (NPP) have been developed to better characterize Conditional Intensity Functions (CIFs) by leveraging the power of deep neural networks. These models can be used for tasks such as predicting future events, estimating the rate of event occurrence, or identifying correlations between events. They offer a flexible and powerful way to analyze point process data, as they can handle complex dependencies between events and incorporate prior knowledge about the process. Traditionally, a point process encodes the timestamp and type of events. However, in healthcare, additional sources of information can be available that could be useful for CIF characterization. For example, the absolute value of a patient's vital signs might be predictive for next-event prediction. NPP models can incorporate such information by using event and state encoders that can learn to extract relevant features from the input data.

Our proposed model, NPP4EHR, is a deep learning model for electronic health records (EHRs) that is based on point process theory. It consists of two modules: a transformer event encoder, which is designed to handle the missingness patterns of certain laboratory values, and a deep attention module for encoding all other available data. Both modules can be jointly learned by using appropriate loss functions for conditional intensity function (CIF) characterization and any downstream task.

We have conducted extensive experiments on existing event sequence data to demonstrate the effectiveness of our improved transformer event encoder in comparison to existing methods. Furthermore, we have shown that encoding states can reduce the negative log-likelihood (NLL) in two EHR datasets and, in some cases, lead to better performance for next event prediction. Overall, our proposed model offers a powerful tool for handling the irregularly sampled time series data present in EHRs and has the potential to improve representation learning in this domain. The contributions of this paper are:

- An improved transformer event encoder that can effectively handle missingness patterns in certain laboratory values and has been validated in baseline datasets.
- A deep attention module for encoding all available patient data, which has been shown to improve CIF characterization for laboratory values.
- Leveraging transformer event encoder in downstream tasks for better representation learning of patient's EHR

II. Background

A. Temporal point process

A temporal point process is a stochastic process that models a sequence of events in a continuous time domain. Consider event sequence data $\mathcal{D} = \{\mathcal{S}_i\}_{i=1}^N$ where N is the total number of samples and each sample $\mathcal{S}_i$ is represented as a sequence of events $\mathcal{S}_i = \{(t_i, e_i)\}_{j=1}^L$, where L is the total number of occurred events, t_j is the event's timestamp, and $e_j \in \mathbb{R}^M$ is the binary representation of event marks (multi-class or multi-label). Furthermore, the history of events at time t is denoted as $\mathcal{H}_t = \{(t_j, e_j) : t_j < t\}$.

The core idea of the point process framework is the definition of conditional intensity functions (CIFs) which is the probability of the occurrence of an event of type m in an infinitesimal time window $[t, t + dt)$:

$$\lambda_m^*(t) = \lim \frac{P(\text{event of type m in } [t, t + \Delta t) | \mathcal{H}_t)}{\Delta t} \tag{1}$$

Here, $*$ denotes conditioning on the history of events$\mathcal{H}_t$. The Multivariate Hawkes process is the traditional approach to characterize CIFs by assuming a fixed form of intensity to account for the additive influence of an event in the past:

$$\lambda_m^*(t) = \mu_m + \sum_{(t', e') \in \mathcal{H}_t} \phi(t - t') \tag{2}$$

where $\mu \geq 0$, aka base intensity, is an exogenous component that is independent of history, while $\phi(t) > 0$, excitation function, is an endogenous component depending on the history that shows the mutual influences. The excitation function can be characterized using exponentials, or linear combination of M basis functions [11].

B. Parameter Estimation

Based on the conditional intensity function 1, it is straightforward to derive conditional probability density function $p_m^*(t)$ in the interval $(t_j, t_{j+1}]$:

$$p_m^*(t) = \lambda_m^*(t) \exp\left[-\sum_{m=1}^{M} \int_{t_j}^{t_{j+1}} \lambda_m^*(s) ds\right] \tag{3}$$

The parameters of the point process can be learned by Maximum Likelihood Estimation (MLE) framework. However, more advanced methods such as adversarial learning, and reinforcement learning have also been proposed.

In the multi-class setting, the log-likelihood (LL) of a point process for a single event sequence $\mathcal{S}_i$ is defined as:

$$\begin{aligned}\log p_{mc}(\mathcal{S}_i) &= \sum_{j=1}^{L}\sum_{m=1}^{M} \mathbb{1}(e_j = m) \log p_m^*(t_j) \\ &= \sum_{j=1}^{L}\sum_{m=1}^{M} \mathbb{1}(e_j = m) \log \lambda_m^*(t_j) \\ &\quad - \sum_{m=1}^{M} \left(\int_{t_1}^{t_L} \lambda_m^*(s)\, ds\right)\end{aligned} \tag{4}$$

Here, $\mathbb{1}$ is the indicator function. The log-likelihood can be understood using the following two facts. First, the quantity $\lambda_k^*(t)dt$ corresponds to the probability of observing an event of type k in the infinitesimal interval $[t_j, t_j + dt]$ conditioned on the past events $\mathbb{H}_{t_j}$. Second, we can compute the probability of not observing any events of type k in the rest of the interval $[t_1, t_L]$ as $\exp\left(-\int_{t_1}^{t_L} \lambda_m^*(s)ds\right)$.

However, in many cases, such as in EHRs, it is common to have co-occurring events. To handle this issue, [3] proposed using a binary cross-entropy function:

$$\begin{aligned}\log p_{ml}(\mathcal{S}_i) &= \log p_{mc}(\mathcal{S}_i) \\ &+ \sum_{m=1}^{M} (1 - \mathbb{1}(e_j = m)) \log\left(1 - p_m^*(t_j)\right)\end{aligned} \tag{5}$$

It is important to note that the main advantage of point processes lies in their ability to model non-event likelihoods in the form of integrals. If we neglect the integrals, we would end up with the cross-entropy and binary cross-entropy loss in the multi-class and multi-label settings, respectively, for predicting the next mark given the history of events.

Another approach is the marked case, which assumes that the marks and timestamps are conditionally independent given $\mathcal{H}_t$:

$$\begin{aligned}\log p_{marked}(\mathcal{S}_i) &= \sum_{j=1}^{L}\sum_{m=1}^{M} \mathbb{1}(e_j = m) \log p^*(e_j = m) \\ &+ \sum_{j=1}^{L} \lambda^*(t_j) - \int_{t_1}^{t_L} \lambda^*(t')dt'\end{aligned} \tag{6}$$

This marked case is basically an autoencoder for next mark prediction with a single dimension point process for timestamps only.

C. Neural temporal point process

Encoder-decoder architectures have proven to be effective in many applications. The main idea of a neural temporal point process (NTPP) is to first encode the history of events until t_j using a neural network architecture $h_j = Enc(\mathcal{H}_{j+1}; \theta)$. Then it tries to estimate m CIF using a different decoder architecture $\lambda_m^*(t) = Dec(h_j; \phi)$ for $t \in (t_j, t_{j+1}]$.

Initial works have utilized recurrent encoders such as RNN, GRU, or LSTM, where the hidden state is updated after the arrival of a new event as $h_{j+1} = Update(h_j, (t_j, e_j))$. The main advantage of these models is their ability to compute history embeddings in $O(L)$ time. However, they are susceptible to ignoring long-term inter-event dependencies. In contrast, set aggregation encoders encode all past events directly into a history embedding. One approach to capture long-term dependencies is to use an attention mechanism, which can be trained in parallel and is more computationally efficient. For instance,[13] proposed Transformer Hawkes Process (THP), which adopts a transformer architecture for event encoding.

The learned event embeddings can be later used for estimating conditional intensity functions, cumulative conditional intensity functions or probability density functions.

D. Deep learning for irregularly sampled data

An irregularly sampled data can be denoted as $\mathcal{D} = \{\mathcal{U}_i\}_{i=1}^N$ where N is number of samples and each sample is represented as sequence of tuples $\mathcal{U}_i = \{(t_p, k_p, v_p)\}_{p=1}^P$ where P is the total number of observations and t_p, k_p, v_p represents the time, modality and value of p-th observation respectively.

Recurrent neural networks can naturally deal with sequential data. These models can be made compatible with irregularly sampled data by adding a strategy to consider time.

III. Related Works

We focus on Neural point process models based on attention mechanisms that can potentially address the problems of slow serial computing and loss of long-term information. In addition, attention weights bring interpretability and can show peer influences between events. Self-attentive Hawkes Process (SAHP) [12] proposes a multi-head attention network as the history encoder. In addition, they use a sofplus function that can capture both excitation and inhibition effects. Similarly, Transformer Hawkes Process (THP) [13] adopts the transformer architecture [9] with time encodings for event embeddings. They have introduced softplus function that can only capture mutual excitations between events. In an interesting study [3], researchers studied different combinations of encoders (self-attention and GRU) and decoders and simulated on various datasets for comparison. They demonstrated that attention-based TPPs appear to transmit pertinent EHR information and perform favorably compared to existing models. One gap in the current literature is that they don't have any means for encoding additional information that can be useful for the characterization of CIFs. For example, in EHRs there are many information in addition to clinical events that could be useful

Recurrent neural networks have been modified to consider irregularly sampled time series. For example, GRU-D [1] adapts GRU to consider missingness patterns in the form of feature masks and time intervals to achieve better prediction results. RETAIN [2] is based on a two-level neural attention model that is specifically designed for EHRs data. SeFT [5] is based on recent advancements in differentiable set function learning, extremely parallelizable with a beneficial memory footprint, thus scaling well to large datasets of long time series and online monitoring scenarios. Their use of aggregation function is similar to transformers that compute the embeddings of set elements independently, leading to lower runtime and memory complexity of $O(n)$. Although these models are nearly end-to-end that eliminate the need for an imputation pipeline, it is still unclear how much they are affected by the missingness pattern in EHRs. In addition, they do not explicitly model the missingness pattern in the data.

IV. Proposed Model

The proposed model, NPP4EHR, consists of two modules for encoding a dataset with both event sequence data and irregularly sampled time series. The schematic of the model is depicted in 2.

The key advantage of our proposed model is to combine a transformer-based event encoder (TEE) with a deep attention module (DAM) that can handle an irregularly sampled time series for any downstream prediction task. The data is represented as $\mathcal{D} = \{(\mathcal{S}_i, \mathcal{U}_i)\}_{i=1}^{N}$, where $\mathcal{S}_i$ and $\mathcal{U}_i$ are event sequence data and irregular time series for the i-th sample as described in previous section.

A. Event Encoder

We use a transformer event encoder similar to THP [13] with minor modifications for time encoding. In the first step, we embed all event marks as $E_{emb} = E \times W_{emb}$, where $E_i \in \mathbb{R}^{L \times M}$ is the binary encoding matrix of all event marks (multi-label or multi-class), and $W_{emb} \in \mathbb{R}^{M \times d_{emb}}$ is the trainable embedding matrix. In addition, we encode vector of timestamps $t = [t_1, t_2, ..., t_L]$ to $Z = [z(t_1), z(t_2), ..., z(t_L)] \in \mathbb{R}^{L \times d_{time}}$ using following transformation formula:

$$[z(t_j)]_k = \begin{cases} \cos\left(\frac{t_j}{\mathcal{T}^{(k-1)/d_t}}\right) & \text{if } k \text{ is odd} \\ \sin\left(\frac{t_j}{\mathcal{T}^{k/d_t}}\right) & \text{if } k \text{ is even} \end{cases} \tag{7}$$

Here, $\mathcal{T}$ is representing the maximum time scale. This transformation is very similar to positional encodings in transformers [9], where the index is substituted by the timestamp. Contrary to THP and original positional encoding that considers $d_{emb} = d_{time}$ and adds up time encoding to the event embedding, we propose to concatenate these two vectors:

$$X_{ev} = [W_{emb}, Z] \in \mathbb{R}^{L \times (d_{emb} + t_{emb})} \tag{8}$$

Finally, we use the standard transformer architecture to encode embedded events X_{ev} into the encoded matrix $H = (h_1, ..., h_j, ..., h_L)$. It is important to use an appropriate mask matrix to prevent information leakage from the future to the past. In this case, h_j should contain all available information until the occurrence of j-th event.

B. State Encoder

We use a deep attention module [5] for encoding all additional information including irregularly sampled time series. Each observation is represented by $u_p = (z(t_p), v_p, m_p)$ where $z(t_p)$ is the same transformation for time encoding in equation 7.

We define $\mathcal{U}_p$ to be the set of the first p observations. The goal is to calculate the attention weight $a(\mathcal{U}_p, u_k), k \leq p$ that is the relevance of k-th observation u_k to the first p observed values $\mathcal{U}_p$. This is achieved by computing an embedding of the set elements using a smaller set functions f', and projecting the concatenation of the set representation and the individual set element into d-dimensional space:

$$f'(\mathcal{U}_p) = g'\left(\frac{1}{|p|}\sum_{u_k \in \mathcal{U}_p} h'(u_k; \theta'); \rho'\right) \tag{9}$$

Here, we compute the mean of the first p observations after passing the first p observations through a multilayer perceptron (MLP) neural network $(h'(u_k; \theta'))$. Finally, a second transformation using g' is performed to obtain embeddings $f'(\mathcal{U}_p) \in \mathbb{R}^{d_{g'}}$.

Then we can compute key values (K_p) using key matrix $W^k \in \mathbb{R}^{(d_{g'}+d_s) \times d_{prod}}$:

$$K_p = [f'(\mathcal{U}_p), u_p]^T W^K \tag{10}$$

Then using a query vector $w^q \in \mathbb{R}^{d_{prod}}$, we compute the desired attnetion weight in this way:

$$a(\mathcal{U}_p, u_k) = softmax(\frac{K_p.w^q}{\sqrt{d}}) \tag{11}$$

Finally, we compute a weighted aggregation of set elements using attention weights similar to equation 9:

$$y_p = f(\mathcal{U}_p) = g\left(\sum_{u_k \in \mathcal{U}_p} a(\mathcal{U}_p, u_k) h(u_k; \theta); \rho\right)$$

We regard $y_p \in \mathbb{R}^{d_g}$ as the representation of first p observations.

The matrix $Y = [y_1, y_2, ..., y_P] \in \mathbb{R}^{P \times d_g}$ keeps the entire represenations of the data. To use state information for CIFs characterization, we down-sample Y to $Y' = [y'_1, y'_2, ..., y'_L] \in \mathbb{R}^{L \times d_g}$ where $y'_j = y_p$ where $p = \text{argmax}_{t_p \leq t_j} p$.

Without loss of generality, we can consider multiple heads by adding an additional dimension to keys and queries.

C. Event Decoder

Once we obtain a representation of a patient using embedded events and states, we can try to parameterize conditional intensity functions (CIFs) of the events.

In neural point process literature, many approaches have been propose to decode either conditional or cumulative intensity function. We will use a decoder similar to [12] as it can model both exciting and inhibiting effects for modeling CIFs.

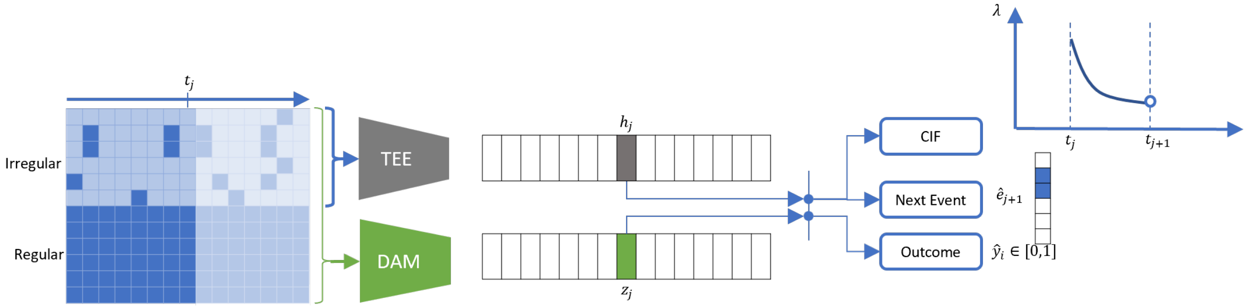

Fig. 1. Magnetization as a function of applied field. It is good practice to explain the significance of the figure in the caption.

$$\mu_{m,j} = gelu(h_j W_{m,\mu} + y'_j W_{m,\mu}), \tag{12}$$
$$\eta_{m,j} = gelu(h_j W_{m,\eta} + y'_j W_{m,\eta}), \tag{13}$$
$$\gamma_{m,j} = gelu(h_j W_{m,\gamma} + y'_j W_{m,\gamma}), \tag{14}$$

Finally, we can express the intensity function as follows:

$$\lambda_m(t) = \text{softplus}(\mu_{m,j} + (\eta_{m,j} - \mu_{m,j}) \exp(-\gamma_{m,j}(t - t_j))), \tag{15}$$

for $t \in (t_j, t_{j+1}]$,where the $softplus$ is used to constrain the intensity function to be positive.

D. Loss Function

We define a multi-objective loss function as $\mathcal{L} = \mathcal{L}_{CIF} + \beta\mathcal{L}_{state}$., where $\mathcal{L}_{CIF}$ could be one of equations 4,5 or 6, and $\mathcal{L}_{state}$ could be any desired loss function such as mean square error or cross-entropy depending on the task. β is a coefficient that can be optimized during hyperparameter tuning.

V. Experiments

We perform various experiments to show the effectiveness of each component in our model.

Datasets

Synthea(*SYN*). We used the Synthea simulator [10] which generates patient-level EHRs using human expert-curated Markov processes. Here, we reused the already processed version of this data by [3].

ReTweets (*RT*). The Retweets dataset contains sequences of tweets, where each sequence contains an origin tweet (i.e., some user initiates a tweet), and some follow-up tweets. We record the time and the user tag of each tweet. Further, users are grouped into three categories based on the number of their followers: “small”, “medium”, and “large”. We use two versions of this dataset for multi-class [12] and multi-label [3] scenario.

Stackoverflow (*SO*) is a question-answering website. The website rewards users with badges to promote engagement in the community, and the same badge can be rewarded multiple times to the same user. We collect data in a two-year period, and we treat each user’s reward history as a sequence. Each event in the sequence signifies receipt of a particular medal. We used the same processed dataset in the literature [13].

Furthermore, we consider two EHRs provided by physionet:

Physionet 2012 Mortality Prediction Challenge (*P12*). The 2012 Physionet challenge dataset [8],contains 12, 000 ICU stays each of which lasts at least 48 hours. For each stay, a set of general descriptors (such as gender or age) are collected at admission time. Depending on the course of the stay and patient status, up to 37 time series variables were measured (e.g. blood pressure, lactate, and respiration rate). While some modalities might be measured in regular time intervals (e.g. hourly or daily), some are only collected when required; moreover, not all variables are available for each stay.

Physionet 2019 Sepsis Early Prediction Challenge (*P19*). This dataset contains clinical data of about 40,000 patients in ICU [7]. Clinical data consist of demographics, vital signs and laboratory values as well as sepsis label in a one-hour time grid.

Experimental Setup

In the first series of experiments, we compare our proposed model for point process modeling in three scenarios:

- *AE(next mark)* is a simple auto-encoder for predicting the next event from event embeddings without CIF characterization. The loss function is cross-entropy or binary cross-entropy for multi-class and multi-label datasets respectively. This scenario provides a baseline for other point process-based scenarios.
- *PP(marked)* is a marked point process (6) where we assume marks and time stamps are independent.
- *PP(multi-class or multi-label)* uses multi-class (4) or multi-label loss (5).

To show the utility of time concatenation, we also report the metrics for the summation case.

In the second series, we want to investigate the effectiveness of state encoding in addition to the events for real-world EHR datasets P12 and P19 in a multi-label setting. We regard the occurrence of certain laboratory variables as events that are based on clinicians’ decisions (see Appendix). As a result, we consider two scenarios:

- *TEE*: Here, we only use the transformer event encoder (TEE) for CIFs characterization.
- *TEE+DAM*: We further encode time stamps and values of all clinical variables through the deep attention module (DAM).

- *TEE+noise*: To further investigate that the improved performance in the second scenario is not due to larger vector representation, we replace the DAM representations by a Gaussian noises vector ($\epsilon \sim \mathcal{N}(0, 1)$).

In the last series of experiments, we investigate the utility of event encoding in a downstream supervised learning task. In particular, we try to predict mortality and sepsis shock as a binary outcome in P12 and P19 respectively. As a result, we compare (TEE+DAM) against the baseline (DAM) for outcome prediction.

We further split healthcare datasets (P12 and P19) into three settings to compare for generalizability across different centers:

- single-center (sc): Training data is from the same center as the test data.
- external evaluation (mc2): Training data is from all centers excluding the test data center:
- multi-center (mc1): Training data is from all available center

Metrics

We report log-likelihood normalized by the number of events (LL/#events) as a goodness of fit for CIFs characterization [12, 13]. For the next event type prediction, we report the weighted measure of F1-score and area under the receiver operating characteristic curve (AUROC) in the multi-class and multi-label setting respectively. In the supervised learning task for binary prediction, we report F1-score and area under the precision-recall curve (AUPRC).

We use t-SNE (t-Distributed Stochastic Neighbor Embedding) for showing learned representations in the downstream tasks which is a machine learning algorithm used for visualizing high-dimensional data in a lower-dimensional space [6].

To show the similarity of events in the learned representation, we first compute the measurement density of each laboratory variable during patient stay:

$$d^i_m = \frac{1}{t_L} \sum_{j=1}^{L} \mathbb{1}(e_j = m), \tag{16}$$

The mean of cosine similarities between the measurement density of a desired patient and its 10 nearest neighbors (in the embedding space) is computed (10-nnps).

Training Details

To be completed.

VI. Results and Discussion

A. Preliminary comparison

Table I shows the performance metrics of NPP4EHR for different datasets in various scenarios. We can see that in most cases time concatenation leads to better results in terms of LL/#events and F1-score/AUROC. While adding time encodings to the event embeddings is the default practice in the point process literature [12, 13] as well as natural language processing, we can see that time concatenation would allow model to achieve better results.

Another interesting fact is that the simple auto-encoder for next event type prediction achieves better results for SO,RT(MC) and SYN(ML) compared to the point process scenario. These datasets have been widely used in the point process literature to show the effectiveness of point process, however, we show that this simple baseline may perform better for some datasets. We remark again that point process loss functions (4,5,6) will reduce to the loss function of AE(next mark) if we neglect the integral term.

RT(MC) is the only dataset in which the point process with multi-label loss (equation 5) performs better indicating the real advantage of modeling non-event likelihood of the point process in this dataset. For future research in the neural point process literature, we would recommend comparing model performance with this simple autoencoder model.

B. State encoding for CIF characterization

Table II shows the result for the utility of state encodings for the estimation of negative likelihood. We see that in all cases, state encoding leads to higher AUROC for the next event type prediction and LL/#event as well. It is intuitive that in a hospital, the absolute value of patient states can be useful for ordering future laboratory events. We have evaluated our model in a healthcare database, however, it can be further evaluated on other event sequence data where additional information is available.

C. Event encoding in supervised learning

Table III indicates the result for Mortality/sepsis prediction task across different settings and hospital centers. Generally, we can see that in the mc1 setting, has the best performance because of the larger training dataset from other centers and the test center. In addition, event encoding module is useful for 2/5 cases while in 1/5 cases degrades the performance.

In general, it might seem problematic to rely on the missingness pattern for outcome prediction as it can hurt generalizability when transferring to a new environment with a different pattern. We regard this approach as a double-edged sword that could improve performance in some cases, especially in a new environment with a similar pattern, but it can also degrade the performance if the target environment has a completely different pattern.

D. Learned represention

Fig 1 visualizes the t-SNE plot for the learned embeddings of [] with and without event encodings. Besides, we visualize the sampling pattern of an example patient and its neighbors. As can be seen, in TEEDAM neighbors have more similar pattern compared to the baseline. Although event encoding does not necessarily leads to better performance, we can learn a better representation that considers both absolute values and patterns. The 10 nearest neighbor pattern similarity (10-nnps) of positive labels for TEE+DAM (0.903±0.058) is signifcantly higher than (DAM) $(0.832 \pm 0.095)(pvalue \leq 0.001)$. One application could be time series generation.

TABLE I
ADD CAPTION

		TEEDAM									
		AE (next mark)		PP(single+mark)		PP (MC/ML)					
Dataset	Metric	concat	sum	concat	sum	concat	sum	Latent	SAHP	THP	GRU-CP
SO (MC)	LL/#events	*ND*	*ND*	-0.56	-0.57	-2.04	-2.05	-1.54	-1.86	-1.84	*NR*
	F1-score	**38.46**	36.91	36.04	34.91	32.65	31.67	28.34	24.12	23.89	26
ReTweet (MC)	LL/#events	*ND*	*ND*	-6.91	-8.036	-11	-23.38	-3.89	-4.56	-4.57	*NR*
	F1-score	**62.69**	61.39	53.91	38.63	53	35.17	58.29	53.92	53.86	*NR*
Synthea (ML)	LL/#events	*ND*	*ND*	-2.405	-2.599	-7.254	-6.58	*ND*	*ND*	*ND*	*NR*
	AUROC	**89.58**	89.19	64.98	63.95	60.95	60.65	*ND*	*ND*	*ND*	0.85(.014)
ReTweet (ML)	LL/#events	*ND*	*ND*	2.045	1.863	-1.517	-1.601	*ND*	*ND*	*ND*	*NR*
	AUROC	69.63	68.54	69.31	68.05	**74.22**	71.27	*ND*	*ND*	*ND*	0.611(0.001)

TABLE II
ADD CAPTION

		LL/#Events			AUROC		
Dataset	setting	TEE	TEE+DAM	TEE+noise	TEE	TEE+DAM	TEE+noise
P12	sc	-0.2345	**-0.0019**	0.2365	67.76	**71.66**	69.99
	mc1	0.1984	**0.3623**	0.2252	74.88	**81.39**	74.52
	mc2	0.1634	**0.2337**	0.0858	78.67	**80.24**	72.03
P19	sc	-0.9734	**-0.8531**	-1.047	79.3	**82.52**	72.13
	mc1	-0.9182	**-0.7641**	-0.937	77.58	**87.09**	73.11
	mc2	-1.199	**-1.04**	-1.135	64	**73.31**	64.96

TABLE III
ADD CAPTION

			F1		AUPRC		AUROC	
Dataset	Setting	Center	DAM	TE+DAM	DAM	TE+DAM	DAM	TE+DAM
P12	mc1	1	0.53 (0.04)	0.52 (0.03)	0.53 (0.06)	0.53 (0.04)	0.85 (0.03)	0.85 (0.03)
		2	0.56 (0.02)	0.55 (0.03)	0.56 (0.05)	0.56 (0.03)	0.86 (0.01)	0.87 (0.01)
		3	0.55 (0.04)	**0.57 (0.03)**	0.59 (0.05)	0.58 (0.03)	0.86 (0.02)	0.87 (0.01)
	mc2	1	0.52	0.52	0.53	0.52	0.85	0.85
		2	0.54	0.54	0.53	**0.55**	0.85	0.86
		3	0.52	0.54	0.56	0.56	0.85	0.86
	sc	1	**0.49 (0.05)**	0.46 (0.01)	**0.48 (0.05)**	0.44 (0.02)	0.82 (0.03)	0.81 (0.02)
		2	**0.52 (0.03)**	0.5 (0.03)	**0.51 (0.05)**	0.47 (0.03)	0.83 (0.01)	0.82 (0.01)
		3	0.53 (0.03)	0.52 (0.04)	0.51 (0.05)	**0.53 (0.05)**	0.84 (0.03)	0.84 (0.02)
	seft	-	0.51	0.51	0.51	0.51	0.85	0.85
P19	mc1	1	0.7 (0.02)	0.69 (0.01)	0.78 (0.02)	**0.78 (0.02)**	0.92 (0.02)	0.93 (0.01)
		2	0.62 (0.04)	0.62 (0.03)	0.7 (0.03)	0.69 (0.03)	0.94 (0.01)	0.94 (0.01)
	mc2	1	0.45	**0.59**	0.68	0.67	0.88	0.87
		2	0.62	0.61	0.53	**0.55**	0.92	0.91
	sc	1	0.68 (0.02)	0.68 (0.03)	0.77 (0.02)	0.76 (0.03)	0.93 (0.0)	0.93 (0.0)
		2	0.61 (0.02)	0.61 (0.01)	0.68 (0.01)	0.67 (0.01)	0.94 (0.0)	0.94 (0.0)

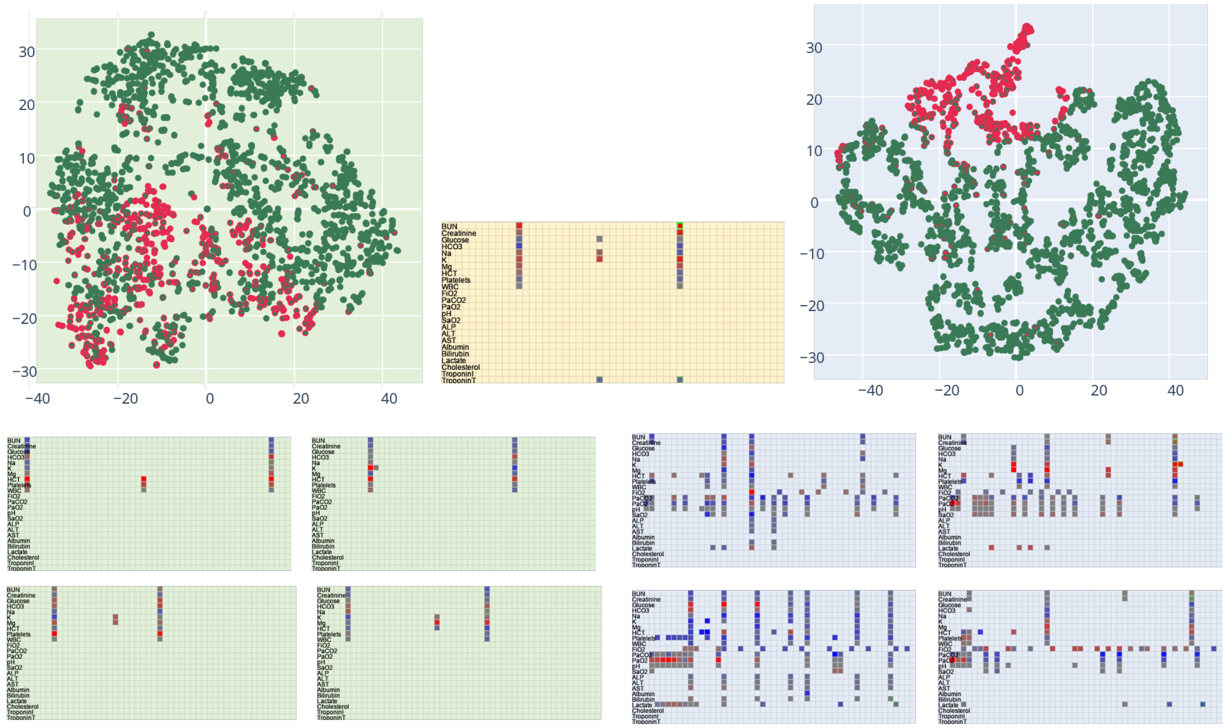

Fig. 2. Magnetization as a function of applied field. It is good practice to explain the significance of the figure in the caption.

E. Model interpretability

one advantage of proposed method is use of attention mechanisms in both event and state encoder. Fig 1 shows the attention mechanism

VII. Conclusion

data generation with similar patterns
pattern similarity index

References

[1] Zhengping Che et al. “Recurrent Neural Networks for Multivariate Time Series with Missing Values”. In: *Scientific Reports* 8.1 (Dec. 2018), p. 6085. ISSN: 2045-2322. DOI: 10.1038/s41598-018-24271-9.

[2] Edward Choi et al. “RETAIN: An Interpretable Predictive Model for Healthcare Using Reverse Time Attention Mechanism”. Feb. 26, 2017. arXiv: 1608.05745 [cs]. URL: http://arxiv.org/abs/1608.05745 (visited on 01/03/2022).

[3] Joseph Enguehard et al. “Neural Temporal Point Processes For Modelling Electronic Health Records”. In: *Proceedings of the Machine Learning for Health NeurIPS Workshop*. Machine Learning for Health. PMLR, Nov. 23, 2020, pp. 85–113. URL: https://proceedings.mlr.press/v136/enguehard20a.html (visited on 05/06/2022).

[4] Marzyeh Ghassemi et al. “A Review of Challenges and Opportunities in Machine Learning for Health”. In: *AMIA Summits on Translational Science Proceedings* 2020 (May 30, 2020), pp. 191–200. ISSN: 2153-4063. pmid: 32477638. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7233077/ (visited on 12/31/2021).

[5] Max Horn et al. “Set Functions for Time Series”. In: *Proceedings of the 37th International Conference on Machine Learning*. International Conference on Machine Learning. PMLR, Nov. 21, 2020, pp. 4353–4363. URL: https://proceedings.mlr.press/v119/horn20a.html (visited on 09/26/2022).

[6] Laurens van der Maaten and Geoffrey Hinton. “Visualizing Data Using T-SNE”. In: *Journal of Machine Learning Research* 9.86 (2008), pp. 2579–2605. ISSN: 1533-7928. URL: http://jmlr.org/papers/v9/vandermaaten08a.html (visited on 03/03/2023).

[7] Matthew A. Reyna et al. “Early Prediction of Sepsis From Clinical Data: The PhysioNet/Computing in Cardiology Challenge 2019”. In: *Critical Care Medicine* 48.2 (Feb. 2020), pp. 210–217. ISSN: 0090-3493. DOI: 10.1097/CCM.0000000000004145.

[8] Ikaro Silva et al. “Predicting In-Hospital Mortality of ICU Patients: The PhysioNet/Computing in Cardiology Challenge 2012”. In: ().

[9] Ashish Vaswani et al. “Attention Is All You Need”. In: *Advances in Neural Information Processing Systems*. Vol. 30. Curran Associates, Inc., 2017. URL: https://proceedings.neurips.cc/paper/2017/hash/

3f5ee243547dee91fbd053c1c4a845aa-Abstract.html (visited on 07/07/2022).

[10] Jason Walonoski et al. "Synthea: An Approach, Method, and Software Mechanism for Generating Synthetic Patients and the Synthetic Electronic Health Care Record". In: *Journal of the American Medical Informatics Association* 25.3 (Mar. 1, 2018), pp. 230–238. ISSN: 1527-974X. DOI: 10.1093/jamia/ocx079.

[11] Hongteng Xu, Mehrdad Farajtabar, and Hongyuan Zha. "Learning Granger Causality for Hawkes Processes". In: *Proceedings of The 33rd International Conference on Machine Learning*. International Conference on Machine Learning. PMLR, June 11, 2016, pp. 1717–1726. URL: https://proceedings.mlr.press/v48/xuc16.html (visited on 04/20/2022).

[12] Qiang Zhang et al. "Self-Attentive Hawkes Process". In: *Proceedings of the 37th International Conference on Machine Learning*. International Conference on Machine Learning. PMLR, Nov. 21, 2020, pp. 11183–11193. URL: https://proceedings.mlr.press/v119/zhang20q.html (visited on 05/09/2022).

[13] Simiao Zuo et al. "Transformer Hawkes Process". In: *Proceedings of the 37th International Conference on Machine Learning*. International Conference on Machine Learning. PMLR, Nov. 21, 2020, pp. 11692–11702. URL: https://proceedings.mlr.press/v119/zuo20a.html (visited on 05/06/2022).