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) is challenging for model development. In addition, the pattern of missingness for certain clinical variables are not at random as it is determined by clinicians decision and the state of patient. Point process is a mathematical framework for handling event sequence data which is also consistent with the irregular sampling. We propose *TEEDAM*, a deep neural network that can learn patient representation from irregulary sample time series as well as informative missingness pattern of certain laboratory variables. We performed various experiments to show the effectiveness of event and state encoding for characterization of conditional intensity functions as well as downstream prediction task. Results show that in some cases learning from patterns may improve the performance of prediction task.**

***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 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.

missingness in tabular and time series.

[sources of missingness] Sources of missingness in EHRs must be carefully understood. For example, lab measurements are typically ordered as part of a diagnostic work-up, meaning that the presence of a datapoint conveys information about the patient's state

[conseqs of IRR] A typical EHR consist of tabular and time series data. Depending on the desired outcome, we can formulate the problem in static or longitudinal setting. There are various consequences asscociated with irregularly sampled data. Many machine learning models are not inherently compatible with

imputation for static setting

imputation for time series.

inherently compatible methods.

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 their own 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 imputation, 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 missig or irregularly-sampled data such as Gaussian process and recurrent neural networks. Deep learning models that are compatible with irregularly sampled time series are important in many real-world applications, such as stock market predictions, speech recognition, and medical diagnosis. These models need to be able to handle data that is not evenly spaced, as is often the case 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 have the ability to 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 be used to extract features from the data and then pass the

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).

processed data to an RNN for further analysis. Additionally, Attention Mechanisms can be integrated into RNNs and CNNs to help the model focus on important features in the data. Overall, these deep learning models offer a powerful toolset for handling irregularly sampled time series and provide useful insights into this type of data. These models does not explicitly take into account the fact wether the absence of some datapoints could convey some unique information.

Point processes are mathematical models used to describe 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 function and the corresponding log likelihood which simultanously models the occurence of events using the history of past events.[GAP] More recently, neural Point Processes have been developed for better characteriztion of CIFs by leveraing 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. Traditioaly, a point process encodes timestample and type of events, however, it is common in healtcare to have additional sources of information that could be useful for CIF characterization. For example the absolute value of patients vital signs might be predictive for next event prediction.[GAP]

our model TEEDAM is a deep learning model for EHR that can simultanously encodes all of events and values and optimizes for a point process objective function as well as a desired outcome. We have performed extensive experiments on existing event sequence data to show the effectivness of our improved transformer event encoder against the existing methods. In addition, we show that encoding states can reduce point process NLL in two EHRs dataset (p12,p19) and in some cases it can lead to better performance for next event prediction. Finally, we provide explanations regarding the interpretablity of out model thanks to the utilized attention mechanisms.

the contributions of this paper are:

- cont1
- cont2

explain structure of paper

II. Background

A. Temporal point process

A temporal point process [13, 14] is a stochastic process that models a sequence of events in a continuous time domain. Consider an event sequence data $\mathcal{D} = \{\mathcal{S}_i\}_{i=1}^{N}$ where N is 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 occured events, t_j is the event's timestamp, and $e_j \in \mathbb{R}^M$ is the binary represntation 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 occurence 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. Multivariate Hawkes process is the traditional approach to characterize CIFs by assumming a fixed form of intensity to account for the additive influence of an history event:

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

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 ge characterized using exponentials, linear combination of M basis functions [**xuLearningGrangerCausality2016**].

B. 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 CIFs using a different decoder architectures $\lambda^*_m(t) = Dec(h_j; \phi)$ for $t \in (t_j, t_{j+1}]$.

Initial works has used recurrent encoders such as RNN, GRU or LSTM [refs] in which the hidden state gets updated after arrival of new event as $h_{j+1} = Update(h_j, (t_j, e_j))$. The main advantage is that they allow us to compute history embeddings in $O(L)$ time, however, they are prone to neglect long-term inter-event influences. On the other hand, set aggregation encoders directly encodes all past events into a history embedding. adopting attention mechanism is one way that can capture long-term influences and can be trained in parallel as well. For example, [**zuoTransformerHawkesProcess2020a**] proposed transformer hawkes process (THP) that adopts a transformer architecture for event encoding.

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

C. Parameter Estimation

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

$$p^*_m(t) = \lambda^*_m(t) \exp\left[-\sum_{m=1}^{M} \int_{t_j}^{t_{j+1}} \lambda^*_m(t')dt'\right] \tag{3}$$

The paramters of point process can be learnt by Maximum Likeihood Estimation (MLE) framework. However, more advanced methods such as adversarial learning, reinforcement learning have also been proposed.

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

$$\begin{aligned}\log p_{mc}(\mathcal{S}_i) &= \sum_{j=1}^{L}\sum_{m=1}^{M} \mathbb{1}(e_j = m)\log p^*_m(t_j)\\ &= \sum_{j=1}^{L}\sum_{m=1}^{M} \mathbb{1}(e_j = m)\log \lambda^*_m(t_j)\\ &\quad - \sum_{m=1}^{M}\left(\int_0^T \lambda^*_m(s)\,ds\right) \quad (4)\end{aligned}$$

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 eevents of type k in the rest of the interval $[0, T]$ as $\exp\left(-\int_0^T \lambda^*_m(s)ds\right)$.

However, in many cases such as EHRs it is common to have co-occuring events. [**enguehardNeuralTemporalPoint2020**] proposed to use a binary cross entropy function:

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

It should be noted that the real advantage of point process is modeling non-event likelihoods in the form of integrals. If we neglect the integrals, we would achieve the cross-entropy and binary cross entropy loss in the multi-class and multi-label settings respectively for the prediction of next mark given history of events.

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

$$\begin{aligned}\log p_{marked}(\mathcal{S}_i) &= \sum_{j=1}^{L}\sum_{m=1}^{M} 1(e_j = m)\log p^*(e_j = m)\\ &\quad + \sum_{j=1}^{L}\lambda^*(t_j) - \int_{t_0}^{t_L}\lambda^*(t')dt' \quad (6)\end{aligned}$$

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

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 based on attention mechanism that can potentially addresses 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) [**zhangSelfAttentiveHawkesProcess2020**] proposes a multihead attention network as the history encoder. In addition, they use a sofplus function that can capture both excitaion and inhibition effects. Similarly, Transformer Hawkes Process (THP) [**zuoTransformerHawkesProcess2020a**] adopts the transformer architecture [**vaswaniAttentionAllYou2017**] with time encodings for event embeddings. They have introduced softplus function that can only capture mutual excitations between events. In a interesting study, reseraches studied different combination of encoders (self attention and GRU) and decoders and simulated on various datasets for comparison [**enguehardNeuralTemporalPoint2020**]. They demonsterated that attention-based TPPs appear to transmit pertinent EHR information and perform favourably compared to existing models. One gap in the current literature is that they don't have any mean for encoding additional information that can be useful for characterization of CIFs.

Recurrent neural networks have been modified to consider irregulary sampled time series. For example, GRU-D [**cheRecurrentNeuralNetworks2018**] adapts GRU to consider missingness pattern in the form of feature masks and time intervals to achieve better prediction results. RETAIN [**choiRETAINInterpretablePredictive2017**] is based on a two-level neural attention model that is specifically designed for EHRs data. SeFT [**hornSetFunctionsTime2020**] 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 computes 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 eliminates the need for imputation pipeline, it is still unclear how much they are affected by the missigness pattern in EHRs. In addition, they do not explicitly model the missingness pattern in the data.

IV. Proposed Model

The proposed model, TEEDAM, 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 **??**.

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 a any downstream prediciton 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.

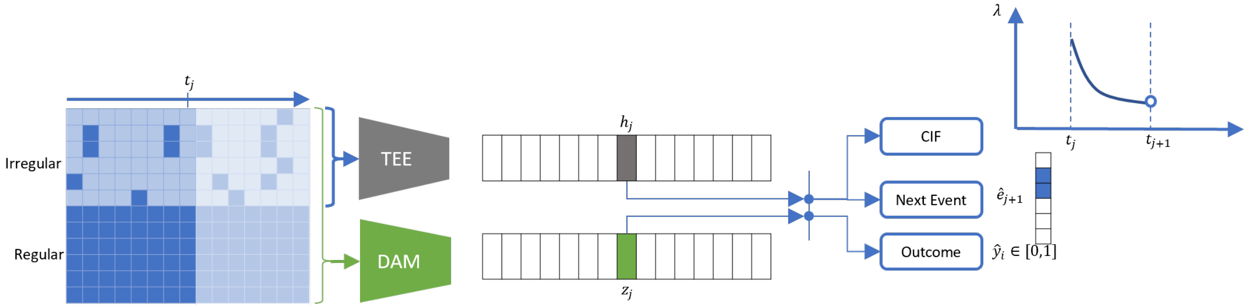

Fig. 1. Magnetization as a function of applied field. It is good practice to explain the significance of the figure in the caption.

A. Event Encoder

We use a transformer event encoder similar to THP [**zuoTransformerHawkesProcess2020a**] with minor modifications for event embedding.

In the first step, we embed all event marks $E_{emb} = E \times W_{emb}$ 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 maximum time scale. This is very similar to positional encodings in transformers where the index is substitued by the time stamp. In 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 this two vectors:

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

Finally, we use the standard transformer encoder to encode embedded events X_{emb} tothe encoded matrix $H = (h_1, ..., h_j, ..., h_L)$. It is important to use appropriate mask matrix to prevent information leakage from future to the past. In this case, h_j containes the all available information until occurence of j-th event.

B. State Encoder

We use the a deep attention module [**hornSetFunctionsTime2020**] for encoding all additional information including irregularly sampled time series. Each observation of a sample can be represented by $u_k = (z(t_k), v_k, m_k)$ where $z(t_k)$ is the same time encoding similar to equation **??**.

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 s_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:

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

Here, we compute the mean of first p observations after passing first p observations through a multilayer perceptron (MLP) nueral 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:

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

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

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

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 represntions 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 $p = \operatorname{argmax}_{t_p \leq t_j} p$.

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

C. All formulas

D. 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 [sahp] as it can model both exciting and inhibiting effects for modeling CIFs.

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

Finally, we can express the intensity function as follows:

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

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

E. 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 **??**,**??** or **??**. $\mathcal{L}_{state}$ could be any desired loss functioin such as mean square error or corss entropy depending on the task. β is a coefficient that can be optimized during hyperparamter tuning.

V. Experiments

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

Datasets

Synthea(*Syn*). We used the Synthea simulator (Walonoski et al., 2018) which generates patient-level EHRs using human expert curated Markov processes. Here, we reused the already processed version of this data by [ntpp].

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”

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.

Furthermore, we consider two EHRs provided by physionet challenge to investigate the advantage of irregular sample and point process modeling in the same time.

Physionet 2012 Mortality Prediction Challenge (*P12*). The 2012 Physionet challenge dataset (Goldberger et al., 2000),contains 12, 000 ICU stays each of which lasts at least 48 h. 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 40k patients in ICU. Clinical data consist of demographics, vital signs and laboratory values as well as sepsis label in a one-hour time grid. Our objective is to predict the timestamp of next lab sampling events as well as measured variables (event marks) given the patient history.

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. As mentioned before, the log-likelihood of the point process will reduce to this one if we ignore the integral term.
- PP(marked) is a marked point process (equation **??**) where we assume marks and time stamps are independent.
- PP(multi-class or multi-label) uses multi-class or multi-label loss.

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. As a result, we consider two scenarios:

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

In the last series of experiments, we investigate the utility of event encoding in a 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.

Metrics

We report log-likelihood normalized by the number of events (LL/#events) as a goodness of fit for CIFs characterization [**zhangSelfAttentiveHawkesProcess2020**, **zuoTransformerHawkesProcess2020a**]. 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).

Training Details

To be completed.

VI. Results and Discussion

In this section, we present our results regarding the advantage of state and event encoding.

A. Preliminary comparison

Table **??** shows the performance metrics of TEEDAM 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 literature [**zhangSelfAttentiveHawkesProcess2020**, **zuoTransformerHawkesProcess2020a**].

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.

RT(MC) is the only dataset in which the point process with multi-label loss (equation **??**) performs better indicating the real advantage of modeling non-event likelihood of point process in this dataset. In general, we would recommend to compare future works in this area with this simple baseline.

B. State encoding for CIF characterization

Table **??** 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 **??** 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 s

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 the patterns. One application could be time series generation.

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

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(0.19)	24.12	23.89	26
ReTweet (MC)	LL/#events	ND	ND					-3.89	-4.56	-4.57	NR
	F1-score	62.48	61.65	58.45	35.59	36.38		58.29	53.92	53.86	NR
Synthea (ML)	LL/#events	ND	ND					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	1.589	1.355	-1.587	-1.656	ND	ND	ND	NR
	AUROC	69.12	67.86	62.5	60.4	73.9	71.8	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	sc	1	0.55 (0.02)	0.55 (0.02)	0.55 (0.02)	0.55 (0.02)	0.55 (0.02)	0.55 (0.02)
		2	0.55 (0.02)	0.55 (0.02)	0.55 (0.02)	0.55 (0.02)	0.55 (0.02)	0.55 (0.02)
		3	0.55 (0.02)	0.55 (0.02)	0.55 (0.02)	0.55 (0.02)	0.55 (0.02)	0.55 (0.02)
	mc1	1	0.55 (0.02)	0.55 (0.02)	0.55 (0.02)	0.55 (0.02)	0.55 (0.02)	0.55 (0.02)
		2	0.55 (0.02)	0.55 (0.02)	0.55 (0.02)	0.55 (0.02)	0.55 (0.02)	0.55 (0.02)
		3	0.55 (0.02)	0.55 (0.02)	0.55 (0.02)	0.55 (0.02)	0.55 (0.02)	0.55 (0.02)
	mc2	-	0.55 (0.02)	0.55 (0.02)	0.55 (0.02)	0.55 (0.02)	0.55 (0.02)	0.55 (0.02)
	seft	-	0.55 (0.02)	0.55 (0.02)	0.55 (0.02)	0.55 (0.02)	0.55 (0.02)	0.55 (0.02)
P19	sc	1	0.55 (0.02)	0.55 (0.02)	0.55 (0.02)	0.55 (0.02)	0.55 (0.02)	0.55 (0.02)
		2	0.55 (0.02)	0.55 (0.02)	0.55 (0.02)	0.55 (0.02)	0.55 (0.02)	0.55 (0.02)
		3	0.55 (0.02)	0.55 (0.02)	0.55 (0.02)	0.55 (0.02)	0.55 (0.02)	0.55 (0.02)
	mc1	1	0.55 (0.02)	0.55 (0.02)	0.55 (0.02)	0.55 (0.02)	0.55 (0.02)	0.55 (0.02)
		2	0.55 (0.02)	0.55 (0.02)	0.55 (0.02)	0.55 (0.02)	0.55 (0.02)	0.55 (0.02)
		3	0.55 (0.02)	0.55 (0.02)	0.55 (0.02)	0.55 (0.02)	0.55 (0.02)	0.55 (0.02)
	mc2	-	0.55 (0.02)	0.55 (0.02)	0.55 (0.02)	0.55 (0.02)	0.55 (0.02)	0.55 (0.02)
	seft	-	0.55 (0.02)	0.55 (0.02)	0.55 (0.02)	0.55 (0.02)	0.55 (0.02)	0.55 (0.02)

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