Point-process-based Representation Learning for Electronic Health Records

Hojjat Karami[1], Anisoara Ionescu[1] and David Atienza[2]

Abstract— **Irregular sampling of time series in electronic health records (EHRs) presents a challenge for the development of machine learning models. Additionally, the pattern of missing data in certain clinical variables is not random, but depends on the decisions of clinicians and the state of the patient. Point process is a mathematical framework for analyzing event sequence data that is consistent with irregular sampling patterns. To tackle the challenges posed by EHRs, we propose a transformer event encoder with point process loss that encodes the pattern of laboratory tests in EHRs. We conduct experiments on two real-world EHR databases to evaluate our proposed approach. Firstly, we learn the transformer event encoder jointly with an existing state encoder in a self-supervised learning approach which gives superior performance in negative log-likelihood and future event prediction. Additionally, we propose an algorithm for aggregating attention weights that can reveal the interaction between the events. Secondly, we transfer and freeze the learned transformer event encoder to the downstream task for the outcome prediction (mortality and sepsis shock), where it outperforms state-of-the-art models for handling irregularly-sample time series. Our results demonstrate that our approach can improve representation learning in EHRs and can be useful for clinical prediction tasks.**

***Clinical relevance*—The transformer event encoder presented in this study can be utilized for analyzing irregularly sampled time series patterns and other event sequences naturally found in modern electronic health records (EHRs). As a result, this approach provides a promising avenue for facilitating data analysis in the field of computer science and healthcare.**

***keywords*: Electronic Health Records (EHRs), Point Process, irregular sampling, informative missingness, self-supervised learning**

I. INTRODUCTION

Machine learning has the potential to revolutionize healthcare by leveraging the 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.

Irregular sampling is one of the data challenges for machine learning (ML) when using EHRs. EHR data is often collected at different times and frequencies, depending on a patient's needs and visit schedules, which can result in uneven and irregularly sampled time series. Asynchronous and incomplete observation of certain clinical variables can also be regarded as missingness in the data. 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 [1].

Irregularly sampled data is a challenge for generic machine learning (ML) models, often requiring imputation or aggregation techniques in pre-processing. In contrast, end-to-end models like Gaussian process [2], Recurrent Neural Networks [3, 4], and transformers [5, 6] are preferred because they can handle irregularly-sampled time series without the need for dealing with missingness in pre-processing.

Point process is a mathematical framework for describing the distribution of events in time or space that naturally works with asynchronous events. In healthcare, medications, ICD codes, or laboratory measurements can be regarded as a sequence of events that are ordered by clinicians. More recently, Neural Point Processes (NPPs) have been developed to better characterize conditional intensity functions (CIFs) by leveraging the representation power of deep neural networks [7, 8].

This study utilized a transformer event encoder (TEE) with a point process loss function in a self-supervised learning approach to learn the interaction between pattern laboratory measurements. An aggregation algorithm was proposed to extract event interaction from attention weights in the transformer. The pre-trained TEE was then transferred to improve the outcome prediction in two EHRs dataset.

II. METHODOLOGY

A. problem formulation

An irregularly sampled data can be denoted as $\mathcal{D} = \{\mathcal{E}_i, \mathcal{U}_i\, S_i, y_i\}_{i=1}^{N}$ where N is the number of samples, S_i and y_i are static covariates and label (if applicable) respectively. The irregularly sampled time series is represented as a 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. Additionally, $\mathcal{E}_i = \{(t_j, e_j)\}_{j=1}^{L}$ represents event sequence data that is available in the patient EHRs.

B. Proposed Model

The model comprises two modules: A transformer event encoder (TEE) for encoding $\mathcal{E}_i$, and a deep attention module (DAM) for encoding $\mathcal{U}_i$. Each of these modules encodes the available data until t_j to produce two embedded vectors: h_j for TEE and y_j for DAM. The concatenated vector $[h_j||y_j]$ is then utilized for any self-supervised or supervised learning task (see Fig. 1).

[1]Hojjat Karami and Anisoara Ionescu are with laboratory of movement analysis and measurement (LMAM), EPFL, Switzerland. Emails: {hojjat.karami, anisoara.ionescu}@epfl.ch

[2]David Atienza is with embedded systems laboratory (ESL), EPFL, Switzerland. Email: david.atienza@epfl.ch

Here, we only explain the TEE module. The DAM is proposed in [6] and is based on recent advances in differentiable set function learning [9, 10]. We refer the reader to the respective references for a better understanding.

Transformer Event Encoder. We use a transformer event encoder (TEE) similar to THP [7] with a few modifications. In the first step, we embed all event marks as $E_{emb} = E \times W_{emb}$, where $E \in \mathbb{R}^{L\times M}$ 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 the 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 the following transformation formula:

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

Here, $\mathcal{T}$ represents the maximum time scale and d_{time} is the time embedding dimension. This transformation closely resembles positional encodings in transformers [11], where the index is replaced by the timestamp. Unlike THP and the original positional encoding [11], which assume $d_{\text{emb}} = d_{\text{time}}$ and add the time encoding to the event embedding, we propose concatenating these two vectors before providing them as input to the transformer block:

$$X_{ev} = [E_{emb}||Z] \in \mathbb{R}^{L\times(d_{emb}+t_{emb})} \tag{2}$$

Finally, we use the standard transformer architecture to encode the embedded events matrix X_{ev} into the encoded matrix $H = (h_1, ..., h_j, ..., h_L)$.

It is important to use the correct attention mask matrix to prevent information leakage from the future to the past. The vector h_j should contain the available information up until the j-th event, which will be used to parameterize the CIFs in the future. To accomplish this, an upper triangular masking matrix m_0 is needed, with the elements above the diagonal masked with one.

We propose a second modification to our approach by introducing a masking parameter, denoted as w. This involves shifting the left column of the input matrix, m_0, by w to obtain the masked matrix, m_w. The resulting output, h_j, will contain information only about the first $(j - w)$ events. This type of masking could be useful in preventing overfitting.

Once we obtain a representation of a sample using embedded events and states, we can parameterize conditional intensity functions (CIFs) of the events. In neural point process literature, many approaches have been proposed to decode either conditional or cumulative intensity functions. We will use a decoder similar to [8] as it can model both exciting and inhibiting effects for CIFs:

$$\mu_{m,j} = gelu(h_j W_{m,\mu} + y_j W_{m,\mu}), \tag{3}$$
$$\eta_{m,j} = gelu(h_j W_{m,\eta} + y_j W_{m,\eta}), \tag{4}$$
$$\gamma_{m,j} = gelu(h_j W_{m,\gamma} + y_j W_{m,\gamma}). \tag{5}$$

The *gelu* function is a nonlinear activation that incorporates the Gaussian Error Linear Unit. Through empirical evidence, it has been demonstrated to outperform other activation functions when utilized for self-attention [12]. Finally, we can express the intensity function as follows:

$$\lambda_m(t) = \text{softplus}(\mu_{m,j} + (\eta_{m,j} - \mu_{m,j}) \exp(-\gamma_{m,j}(t - t_j))), \tag{6}$$

for $t \in (t_j, t_{j+1}]$, where the $softplus$ ensures the CIF to be positive. Finally, we define the loss function in the marked case, which assumes that the marks and timestamps are conditionally independent given $\mathcal{H}_t$:

$$\log p_{marked}(\mathcal{S}_i) = \sum_{j=1}^{L} \sum_{m=1}^{M} \mathbb{1}(e_j = m) \log p^*(e_j = m) + \sum_{j=1}^{L} \lambda^*(t_j) - \int_{t_1}^{t_L} \lambda^*(t')dt' \tag{7}$$

This marked case is essentially an autoencoder for the next mark prediction, with a single-dimensional point process for timestamps only. We use Mont Carlo sampling to compute the integral in equation 7.

Attention Aggregation. The attention matrix of the TEE for each sample can be used for model interpretability. Additionally, we can aggregate attention matrices of a group of samples to extract an influence matrix that reveals the interaction between events. Consider the attention matrix of a sample, $A^p_{L_p\times L_p}$ where the sum of the elements in each row equals one. we multiply each row by the number of unmasked events to compensate for different event counts. We define the allowable event interaction matrix C^p such that $C^p_{ij} = 1$ if $A^p_{ij} > 0$. Similarly, the active event interaction matrix I^p specifies the significant attention values such that $I^p_{ij} = 1$ if $A^p_{ij} > \epsilon$. Here, we assume $\epsilon = 3$.

Finally, we can compute the aggregated allowable interaction C^{agg} and aggregated influence matrix I^{agg} for N samples:

$$C^{agg}_{mn} = \left(\sum_{p=1}^{N} \sum_{j=2}^{L_p} \sum_{i=1}^{j} C^p_{ij} \mathbb{1}(e^p_j = n)\mathbb{1}(e^p_i = m)\right) \tag{8}$$

$$I^{agg}_{mn} = \left(\sum_{p=1}^{N} \sum_{j=2}^{L_p} \sum_{i=1}^{j} I^p_{ij} \mathbb{1}(e^p_j = n)\mathbb{1}(e^p_i = m)\right) / C^{agg}_{mn} \tag{9}$$

Here, C^{agg}_{mn} can be interpreted as the number of times the event n occurs before the event m. Similarly, I^{agg}_{mn} reveals the fraction of C^{agg}_{mn} in which event n plays a significant role in the prediction of event m.

C. Baselines & Training

We compare our method to the following approaches which are particularly adapted for irregularly-sampled time

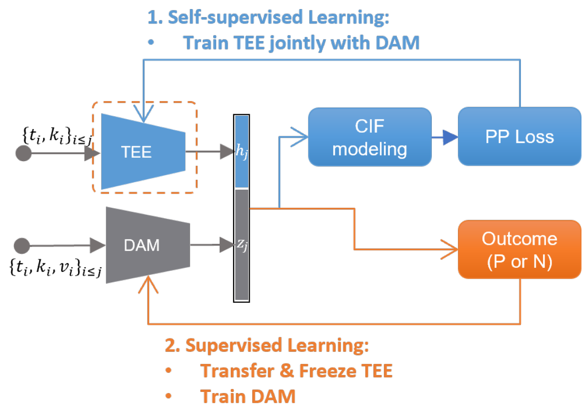

Fig. 1. Training procedure

series: Transformer [11], GRU-D [3], SeFT [6], IP-NET [13], mTAND [14], and RainDrop [5].

Our experiment employed the identical 80-10-10 train-validation-test split as described in RainDrop [5]. To ensure consistency, we execute 20 epochs for all models, capturing the parameters with the highest AUROC in the validation set, which we subsequently apply to make predictions for testing samples. A batch size of 128 and a learning rate of 0.001 are utilized. The masking parameter w is also set to 0. The training set was upsampled so that each batch contains equal number of positive and negative examples.

D. Datasets

We utilize two intensive care unit (ICU) electronic health record (EHR) datasets for analysis. The 2012 Physionet challenge dataset (Physio12) [15] consists of 12,000 ICU stays, each with at least 48 hours duration. Patients' demographic information is included, along with 36 irregularly observed sensor readings. The objective is to predict mortality (prevalence of 14%). The second dataset is the 2019 Physionet challenge dataset (Physio19) [16], which includes clinical data from almost 40,000 ICU patients, sharing similar variables compared to Physio12. The focus of this dataset is on predicting sepsis mortality (prevalence of 4.2%).

III. EXPERIMENTS & RESULTS

The overall training procedure (self-supervised learning followed by supervised learning) is depicted in Fig. 1 which we explain in more details.

A. Self-supervised learning

In this work, we jointly train both TEE and DAM with a defined decoder loss in equation (7). Two evaluation metrics are reported, negative log-likelihood normalized by the number of events (NLL/#events) [8], and area under the receiver operating characteristic curve (AUROC) for future event prediction. We also compare our model with the case where only TEE is used to demonstrate the effectiveness of DAM for CIF characterization.

The results for the utility of state encodings in two healthcare datasets (Physio12, Physio19) are presented in Table I. Our analysis indicates that state encoding using DAM consistently results in higher AUROC to predict the future event type and a higher LL/#event ratio in all cases.

It is also intuitive that, in the hospital setting, the absolute value of patient states could prove useful in determining the order of future laboratory events. Furthermore, this is the first study which demonstrates that an additional module could prove useful in neural point process modeling.

TABLE I
PERFORMANCE OF OUR MODEL IN THE SELF-SUPERVISED LEARNING TASK

		Metric	
Dataset	**Model**	**AUROC**	**LL/#events**
Physio12	TEE	0.80(0.04)	-1.34(0.04)
	TEE+DAM	**0.84(0.02)**	**-1.30(0.03)**
Physio19	TEE	0.78(0.05)	-0.87(0.26)
	TEE+DAM	**0.86(0.02)**	**-0.72(0.27)**

Another advantage of the proposed method is the use of attention mechanisms in both event and state encoders. Fig.2 shows the t-SNE plot for the learned representations of an example ($[h_j||y_j]$) in the self-supervised learning task, where each data point is colored by its true label (red is positive). It is interesting to see that although we have not trained our model for the prediction label, the prevalence of positive examples is different in the two subgroups shown in the plot (25% vs 4%). This means that the sampling pattern could be informative of the label. Additionally, we have shown the aggregated influence matrix (I^{agg}) for the two subgroups. Here, each event represents 10 most frequent patterns of laboratory values (show in bottom of 2). We can see that the general structure of influence matrix seems to be the same, however, in the second group (G2), P9 and P10 has more influence on other events.

B. Supervised learning

In the second step, we transfer the pre-trained TEE from the first step and freeze all its parameters (1). We then optimize the DAM with respect to the label loss, which is a binary cross-entropy loss function. We freeze the TEE to ensure that the missingness pattern in the data does not affect our model's ability to label accurately.

Table II presents the results for the downstream tasks in Physio12 and Physio19. It can be observed that the AUROC and AUPRC consistently improve in Physio12, as well as the AUROC in Physio19.

IV. CONCLUSION

In this work, we propose a transformer event encoder that utilizes the missingness pattern of irregularly sampled time series in electronic health records (EHRs). We have demonstrated its effectiveness in both self-supervised and supervised learning tasks using two real-world databases.

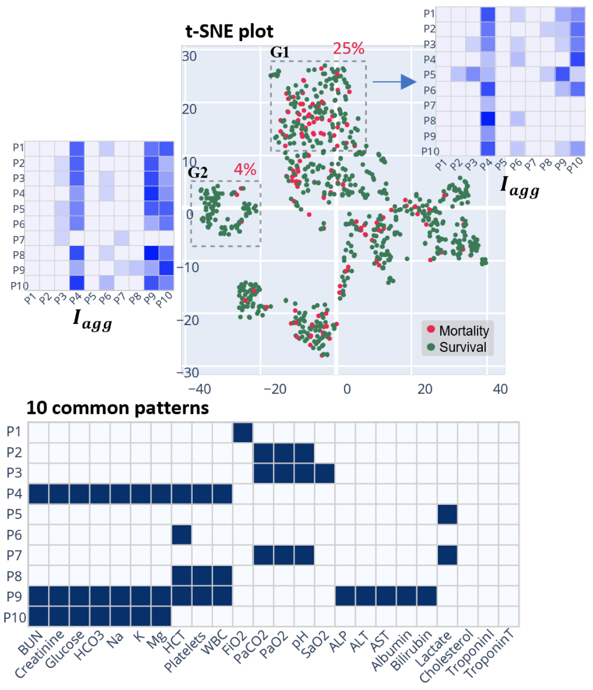

Fig. 2. t-SNE plot for the learned representations in the self-supervised learning step. The aggregated influence matrix is shown for two subgroups

TABLE II

PERFORMANCE OF OUR MODEL COMPARED TO THE STATE-OF-THE-ART METHODS IN THE SUPERVISED LEARNING STEP

	Physio12		Physio19	
	AUROC	AUPRC	AUROC	AUPRC
Transformer	83.3(0.7)	47.9(3.6)	82.7(2.6)	41.4(5.9)
GRU-D	81.9(2.1)	46.1(4.7)	83.9(1.7)	**46.9(2.1)**
SeFT	85.1(0.4)	52.4(1.1)	81.2(2.3)	41.9(3.1)
mTAND	84.2(0.8)	48.2(3.4)	80.4(1.3)	32.4(1.8)
IP-Net	82.6(1.4)	47.6(3.1)	84.6(1.3)	38.1(3.7)
Raindrop	82.6(1.7)	44.0(0.3)	75.0(4.8)	47.2(6.8)
Ours	**85.5(0.3)**	**53.1(2.1)**	**86.6(2.0)**	45.7(5.6)

It is important to note that the TEE does not require any extra information, rather it utilizes the pattern of laboratory values. Another benefit of a TEE encoder is its ability to handle additional data in the form of event sequence data, such as medication details and clinical events. Although we did not have access to additional data in this dataset, in other datasets like MIMIC-IV, numerous data points exist that could be utilized through TEE.

ACKNOWLEDGMENT

This work was supported by the European Union's Horizon 2020 research programme and innovation under the DIGIPREDICT project (GA 101017915).

REFERENCES

[1] 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.

[2] Ahmed M. Alaa, Scott Hu, and Mihaela Schaar. "Learning from Clinical Judgments: Semi-Markov-Modulated Marked Hawkes Processes for Risk Prognosis". In: *Proceedings of the 34th International Conference on Machine Learning*. International Conference on Machine Learning. PMLR, July 17, 2017, pp. 60–69.

[3] 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.

[4] Edward Choi et al. "Doctor AI: Predicting Clinical Events via Recurrent Neural Networks". In: *Proceedings of the 1st Machine Learning for Healthcare Conference*. Machine Learning for Healthcare Conference. PMLR, Dec. 10, 2016, pp. 301–318.

[5] Xiang Zhang et al. *Graph-Guided Network for Irregularly Sampled Multivariate Time Series*. Mar. 15, 2022. arXiv: 2110.05357 [cs]. preprint.

[6] 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.

[7] 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.

[8] 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.

[9] Manzil Zaheer et al. "Deep Sets". In: *Advances in Neural Information Processing Systems*. Vol. 30. Curran Associates, Inc., 2017.

[10] Edward Wagstaff et al. "On the Limitations of Representing Functions on Sets". In: *Proceedings of the 36th International Conference on Machine Learning*. International Conference on Machine Learning. PMLR, May 24, 2019, pp. 6487–6494.

[11] Ashish Vaswani et al. "Attention Is All You Need". In: *Advances in Neural Information Processing Systems*. Vol. 30. Curran Associates, Inc., 2017.

[12] Dan Hendrycks and Kevin Gimpel. *Gaussian Error Linear Units (GELUs)*. July 8, 2020. DOI: 10.48550/arXiv.1606.08415. arXiv: 1606.08415 [cs]. preprint.

[13] Satya Narayan Shukla and Benjamin M. Marlin. *Interpolation-Prediction Networks for Irregularly Sampled Time Series*. Sept. 13, 2019. DOI: 10.48550/arXiv.1909.07782. arXiv: 1909.07782 [cs, stat]. preprint.

[14] Satya Narayan Shukla and Benjamin Marlin. "Multi-Time Attention Networks for Irregularly Sampled Time Series". In: International Conference on Learning Representations. Sept. 28, 2020.

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

[16] 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.