Appendix: Self-Supervised Contrastive Pre-Training for Multivariate Temporal Point Processes

Anonymous Author(s)
Affiliation
Address
email

1 Synthetic Generators

We generated datasets from Hawkes process and Proximal Graphical Event Model (PGEM). We describe the parameters used in our experiments to generate event datasets.

Hawkes-Exp. We use a standard library to simulate 10-dimensional event sequences from the Hawkes dynamics with exponential decay[1].The parameters which control the generation are baseline rate, decay coefficent, adjacency (infectivity matrix) and end time. The four parameters for models A, B, C, D, E and F in our study are different from each other, and thus represent a variety of generating mechanisms which are implied by differing baseline rates, decaying patterns, triggering behaviors and sequence time horizons. The actual values for the 6 models used in our experiments are listed in the following for the sake of reproducibility.

A. baseline = [0.1097627 , 0.14303787, 0.12055268, 0.10897664, 0.08473096, 0.12917882, 0.08751744, 0.1783546 , 0.19273255, 0.0766883], decay = 2.5, infectivity = [[0.15037453, 0.10045448, 0.10789028, 0.17580114, 0.01349208, 0.01654871, 0.00384014, 0.1581418 , 0.14779747, 0.16524382], [0.1858717 , 0.1517864 , 0.08765006, 0.14824807, 0.02246419, 0.12154197, 0.02722749, 0.17942359, 0.0991161 , 0.07875789], [0.05024778, 0.14705235, 0.0866379 , 0.10796424, 0.0035688 , 0.11730922, 0.11625704, 0.11717598, 0.17924869, 0.12950002], [0.06828233, 0.08300669, 0.13250303, 0.01143879, 0.12664085, 0.12737611, 0.03995854, 0.02448733, 0.05991018, 0.0690806], [0.10829905, 0.0833048 , 0.18772458, 0.01938165, 0.03967254, 0.03063796, 0.12404668, 0.04810838, 0.0885677 , 0.04642443], [0.03019353, 0.02096386, 0.1246585 , 0.02624547, 0.03733743, 0.07003299, 0.15593352, 0.01844271, 0.1591532 , 0.01825224], [0.18546165, 0.08901222, 0.18551894, 0.11487999, 0.14041038, 0.00744305, 0.05371431, 0.02282927, 0.05624673, 0.02255028], [0.06039543, 0.07868212, 0.01218371, 0.13152315, 0.10761619, 0.05040616, 0.09938195, 0.01784238, 0.10939112, 0.1765038], [0.06050668, 0.1267631 , 0.02503273, 0.13605401, 0.0549677 , 0.03479404, 0.11139803, 0.00381908, 0.15744288, 0.00089182], [0.12873957, 0.05128336, 0.13963744, 0.18275114, 0.04724637, 0.10943116, 0.11244817, 0.10868939, 0.04237051, 0.18095826]] and end-time = 10.

B. Baseline = [8.34044009e-02, 1.44064899e-01, 2.28749635e-05, 6.04665145e-02, 2.93511782e-02, 1.84677190e-02, 3.72520423e-02, 6.91121454e-02, 7.93534948e-02, 1.07763347e-01], decay = 2.5, adjacency = [[0. , 0. , 0.03514877, 0.15096311, 0. , 0.11526461, 0.07174169, 0.09604815, 0.02413487, 0.], [0. , 0. , 0. , 0. , 0.15066599, 0.15379789, 0.01462053, 0.00671417, 0. , 0.], [0.01690747, 0.07239546, 0.16467728, 0. , 0.11894528, 0. , 0.11802102, 0.14348615, 0.00314406, 0.12896239], [0.17000181, 0. , 0. , 0. , 0. , 0. , 0. , 0.0504772 , 0.04947341, 0.], [0. , 0.11670321, 0.03638242, 0. , 0. , 0.00917392, 0. , 0.0252251 , 0.10131151, 0.1203002], [0. , 0.07118317, 0.11937903, 0.07120436, 0. , 0. , 0. , 0.08851807, 0.16239168,

[1]https://x-datainitiative.github.io/tick/

0.10083865], [0. , 0.02363421, 0.02394394, 0.1388041 , 0.06836732, 0.02842716, 0.15945428, 0. , 0. , 0.12481123], [0.15185513, 0. , 0.1290996 , 0. , 0. , 0.15401787, 0. , 0.16587219, 0.11405672, 0.10687992], [0. , 0. , 0.07734744, 0.09943488, 0.07016556, 0.04074891, 0. , 0. , 0.00049346, 0.10609756], [0.05615574, 0.09061013, 0.15230831, 0.06142066, 0.15619243, 0.10716606, 0.00271994, 0.15978585, 0.11877677, 0.17145652]] and end-time = 40.

C. baseline = [0.08719898, 0.00518525, 0.1099325 , 0.08706448, 0.08407356, 0.06606696, 0.04092973, 0.12385419, 0.05993093, 0.05336546], decay = 2.5, adjacency = [[0. , 0.09623737, 0. , 0. , 0. , 0.14283231, 0. , 0. , 0. , 0.], [0. , 0. , 0. , 0. , 0. , 0. , 0. , 0. , 0. , 0.], [0. , 0. , 0. , 0. , 0. , 0. , 0.14434229, 0.10548788, 0. , 0.], [0. , 0. , 0.16178088, 0. , 0. , 0. , 0. , 0. , 0. , 0.], [0. , 0. , 0. , 0. , 0. , 0. , 0. , 0. , 0. , 0.], [0.14554646, 0. , 0. , 0. , 0. , 0.09531432, 0. , 0. , 0. , 0.], [0. , 0. , 0. , 0. , 0. , 0. , 0. , 0. , 0. , 0.04899491], [0. , 0. , 0. , 0.07048057, 0. , 0.07546073, 0. , 0. , 0. , 0.], [0.05697369, 0. , 0. , 0. , 0. , 0. , 0.11709833, 0. , 0.03100542, 0.], [0. , 0. , 0. , 0. , 0. , 0. , 0.04016475, 0. , 0.10768499, 0.06297179]] and end-time = 80.

D. Baseline = [0.11015958, 0.14162956, 0.05818095, 0.10216552, 0.17858939, 0.17925862, 0.02511706, 0.04144858, 0.01029344, 0.08816197], decay = [[8., 9., 2., 7., 3., 3., 2., 4., 6., 9.], [2., 9., 8., 9., 2., 1., 6., 5., 2., 6.], [5., 8., 7., 1., 1., 3., 5., 6., 9., 9.], [8., 6., 2., 2., 2., 6., 6., 8., 5., 4.], [1., 1., 1., 1., 3., 3., 8., 1., 6., 1.], [2., 5., 2., 3., 3., 5., 9., 1., 7., 1.], [5., 2., 6., 2., 9., 9., 8., 1., 1., 2.], [8., 9., 8., 5., 1., 1., 5., 4., 1., 9.], [3., 8., 3., 2., 4., 3., 5., 2., 3., 3.], [8., 4., 5., 2., 7., 8., 2., 1., 1., 6.]], adjacency = [[0. , 0.14343731, 0. , 0.11247478, 0. , 0. , 0. , 0.09416725, 0. , 0.], [0. , 0. , 0. , 0. , 0. , 0. , 0. , 0.15805746, 0. , 0.08262718], [0. , 0. , 0.03018515, 0. , 0. , 0. , 0. , 0. , 0. , 0.], [0. , 0.11090719, 0.08158544, 0. , 0.11109462, 0. , 0. , 0. , 0. , 0.], [0. , 0. , 0.14264684, 0. , 0. , 0.11786607, 0. , 0. , 0.01101593, 0.], [0.04954495, 0. , 0. , 0. , 0. , 0.12385743, 0. , 0. , 0.02375575, 0.05345351], [0. , 0.14941748, 0.02618691, 0. , 0.13608937, 0. , 0.06263167, 0. , 0.04097688, 0.14101171], [0. , 0.11902986, 0. , 0.04889382, 0. , 0. , 0.01569298, 0.03678315, 0. , 0.], [0.05359555, 0. , 0. , 0.09188512, 0. , 0.14255311, 0. , 0. , 0. , 0.], [0.12792415, 0.05843994, 0.16156482, 0.11931973, 0. , 0.00774966, 0.00947755, 0. , 0. , 0.]], and end-time = 50.

E. Baseline = [0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2], decay = [[9., 4., 9., 9., 1., 6., 4., 6., 8., 7.], [1., 5., 8., 9., 2., 7., 3., 3., 2., 4.], [6., 9., 2., 9., 8., 9., 2., 1., 6., 5.], [2., 6., 5., 8., 7., 1., 1., 3., 5., 6.], [9., 9., 8., 6., 2., 2., 2., 6., 6., 8.], [5., 4., 1., 1., 1., 1., 3., 3., 8., 1.], [6., 1., 2., 5., 2., 3., 3., 5., 9., 1.], [7., 1., 5., 2., 6., 2., 9., 9., 8., 1.], [1., 2., 8., 9., 8., 5., 1., 1., 5., 4.], [1., 9., 3., 8., 3., 2., 4., 3., 5., 2.]], adjacency= [[0.02186539, 0.09356695, 0. , 0.16101355, 0.11527002, 0.09149395, 0. , 0. , 0.15672219, 0.], [0. , 0. , 0.14241135, 0. , 0.11167029, 0. , 0. , 0. , 0.0934937 , 0.], [0. , 0. , 0. , 0. , 0. , 0. , 0. , 0. , 0.15692693, 0.], [0.08203618, 0. , 0. , 0.02996925, 0. , 0. , 0. , 0. , 0. , 0.], [0. , 0. , 0.11011391, 0.08100189, 0. , 0.11029999, 0. , 0. , 0. , 0.], [0. , 0. , 0. , 0.14162654, 0. , 0. , 0.11702301, 0. , 0. , 0.01093714], [0. , 0.04919057, 0. , 0. , 0. , 0. , 0.12297152, 0. , 0. , 0.02358583], [0.05307117, 0. , 0.14834875, 0.02599961, 0. , 0.13511597, 0. , 0.06218368, 0. , 0.04068378], [0.1400031 , 0. , 0.11817848, 0. , 0.0485441 , 0. , 0. , 0.01558073, 0.03652006, 0.], [0. , 0.05321219, 0. , 0. , 0.0912279 , 0. , 0.14153347, 0. , 0. , 0.]] and end-time = 50.

F. Baseline = [0.21736198, 0.11134775, 0.16980704, 0.33791045, 0.00188754, 0.04862765, 0.26829963, 0.3303411 , 0.05468264, 0.23003733], decay = [[5., 1., 7., 3., 5., 2., 6., 4., 5., 5.], [4., 8., 2., 2., 8., 8., 1., 3., 4., 3.], [6., 9., 2., 1., 8., 7., 3., 1., 9., 3.], [6., 2., 9., 2., 6., 5., 3., 9., 4., 6.], [1., 4., 7., 4., 5., 8., 7., 4., 1., 5.], [5., 6., 8., 7., 7., 3., 5., 3., 8., 2.], [7., 7., 1., 8., 3., 4., 6., 5., 3., 5.], [4., 8., 1., 1., 6., 7., 7., 6., 7., 5.], [8., 4., 3., 4., 9., 8., 2., 6., 4., 1.], [7., 3., 4., 5., 9., 9., 6., 3., 8., 6.]], adjacency = [[0.15693854, 0.04896059, 0.0400508 , 0. , 0. , 0.13021228, 0. , 0.10699903, 0. , 0.15329807], [0.0784283 , 0. , 0. , 0. , 0. , 0. , 0.00310706, 0.0090892 , 0.07758874, 0.], [0.01672489, 0. , 0. , 0.07851303, 0. , 0. , 0.12848331, 0.08859293, 0. , 0.], [0.09984995, 0. , 0. , 0.10541925, 0. , 0.08032527, 0. , 0. , 0. , 0.], [0.14642469, 0.06629365, 0. , 0. , 0. , 0. , 0.11891738, 0.04166225, 0.09808829, 0.17638655], [0.00976324, 0.1100343 , 0.02003261, 0. , 0. , 0.05993539, 0.09739541, 0. , 0. , 0.], [0. , 0. , 0. , 0.04672133, 0.16916 , 0. , 0.17341419, 0.12078975, 0.14441602, 0.], [0. , 0.17305542, 0.06927975, 0. , 0.03408974, 0. , 0.08457162, 0.03787486, 0.10863292, 0.], [0.07186225, 0.05760593, 0. , 0. , 0.08042031, 0.04403479, 0.1033595 , 0.17046747, 0. , 0.05083523], [0. , 0.10029222, 0. , 0.1022067 , 0. , 0. , 0.0588527 , 0. , 0.03530513, 0.]], and end-time = 40.

PGEM. We implement the PGEM generator by previous work [1] and generate 5-dimensional event datasets governed by the PGEM dynamics, which is signified by its piecewise conditional intensity function for each event type at a given time instance of a parental configuration observed at a particular window. The parameters are the conditional intensity (lambdas) for each event type given parental states, parental configuration (parents), windows for each parental state (windows) and end time which is 100 across all models. The parameters for models A, B, C, D and E are listed in the following. We point out to the readers that the 5 models share different graphs, which is implied by the parents for each event type (node).

A has the following parameters: 'parents': 'A': [], 'B': [], 'C': ['B'], 'D': ['A', 'B'], 'E': ['C'], 'windows': 'A': [], 'B': [], 'C': [15], 'D': [15, 30], 'E': [15], 'lambdas': 'A': (): 0.2, 'B': (): 0.05, 'C': (0,): 0.2, (1,): 0.3, 'D': (0, 0): 0.1, (0, 1): 0.05, (1, 0): 0.3, (1, 1): 0.2, 'E': (0,): 0.1, (1,): 0.3

B has the follow parameters: 'parents': 'A': ['B'], 'B': ['B'], 'C': ['B'], 'D': ['A'], 'E': ['C'], 'windows': 'A': [15], 'B': [30], 'C': [15], 'D': [30], 'E': [30], 'lambdas': 'A': (0,): 0.3, (1,): 0.2, 'B': (0,): 0.2, (1,): 0.4, 'C': (0,): 0.4, (1,): 0.1, 'D': (0,): 0.05, (1,): 0.2, 'E': (0,): 0.1, (1,): 0.3.

C has the follow parameters: 'parents': 'A': ['B', 'D'], 'B': [], 'C': ['B', 'E'], 'D': ['B'], 'E': ['B'], 'windows': 'A': [15, 30], 'B': [], 'C': [15, 30], 'D': [30], 'E': [30], 'lambdas': 'A': (0, 0): 0.1, (0, 1): 0.05, (1, 0): 0.3, (1, 1): 0.2, 'B': (): 0.2, 'C': (0, 0): 0.2, (0, 1): 0.05, (1, 0): 0.4, (1, 1): 0.3, 'D': (0,): 0.1, (1,): 0.2, 'E': (0,): 0.1, (1,): 0.4.

D has the follow parameters: 'parents': 'A': ['B'], 'B': ['C'], 'C': ['A'], 'D': ['A', 'B'], 'E': ['B', 'C'], 'windows': 'A': [15], 'B': [30], 'C': [15], 'D': [15, 30], 'E': [30, 15], 'lambdas': 'A': (0,): 0.05, (1,): 0.2, 'B': (0,): 0.1, (1,): 0.3, 'C': (0,): 0.4, (1,): 0.2, 'D': (0, 0): 0.1, (0, 1): 0.3, (1, 0): 0.05, (1, 1): 0.2, 'E': (0, 0): 0.1, (0, 1): 0.02, (1, 0): 0.4, (1, 1): 0.1.

E has the follow parameters: 'parents': 'A': ['A'], 'B': ['A', 'C'], 'C': ['C'], 'D': ['A', 'E'], 'E': ['C', 'D'], 'windows': 'A': [15], 'B': [30, 30], 'C': [15], 'D': [15, 30], 'E': [15, 30], 'lambdas': 'A': (0,): 0.1, (1,): 0.3, 'B': (0, 0): 0.01, (0, 1): 0.05, (1, 0): 0.1, (1, 1): 0.5, 'C': (0,): 0.2, (1,): 0.4, 'D': (0, 0): 0.05, (0, 1): 0.02, (1, 0): 0.2, (1, 1): 0.1, 'E': (0, 0): 0.1, (0, 1): 0.01, (1, 0): 0.3, (1, 1): 0.1.

2 Real Datasets

Cosmetics and **Electronics** contain user-level online transactions in an electronics and cosmetics store respectively. While the original cosmetics dataset from Kaggle contain multiple months of transactions, we used the one from Dec, 2019. Similarly, we also used electronics dataset from Dec, 2019. Both share the same four types of events: 'view', 'cart','remove-from-cart' and 'purchase. The two datasets involve transaction events in seconds. To optimize computation for transformer models, we filtered out sequences longer than 300 events and shorter than 30 and scaled the timestamps into [0,1] to avoid numerical issues. We test the algorithms on Electronics after pre-training on Cosmetics.

Defi-Mainnet and **Defi-Polygon**. Defi Mainnet is built on the more widely used ethereum blockchain. Polygon is a scalable sidechain of Ethereum that allows for much faster and low fee transactions than the original Ethereum Blockchain. The difference in fee structure produces quite different dynamics in the two different AAVE lending protocols. DeFi-Polygon has many more users and transactions per user, but much less total value locked than Defi Mainnet. The polygon users are much more likely to engage in risky but potentially profitable "Yield Farming" transactions. Foundation methods would be very useful in modeling the many other lending protocols in AAVE or other lending platforms which are new or less popular and thus have less transactions. The 6 types of actions user performs are : 'borrow', 'collateral', 'deposit', 'liquidation', 'redeem', 'repay'. The origin timestamps are mined in Unix Timestamp. We filtered out sequences longer than 300 events and shorter than 30 and scaled the timestamps into [0,1] to avoid numerical issues. We test the algorithms on Mainnet after pre-training on Polygon.

ACLED-India and **ACLED-Bangladesh** contains sequences where each sequence involves an actor involving some armed conflict (i.e. riots and protests) in the respective country. The former involves events happening from 2016 to 2022, and the latter 2010-2021. The unit of each timestamp is 'days'. There are 6 types of events in the ACLED-India which are: 'Battles', 'Explosions/Remote violence', 'Riots', 'Violence against civilians','Protests',and 'Strategic developments'. The 4 overlapping types in ACLED-Bangladesh are 'Battles', 'Explosions/Remote violence', 'Riots',and 'Violence against civilians'. We filtered out sequences longer than 300 events and shorter than 2 and scaled the timestamps into [0,1] to avoid numerical issues. We test the algorithms on Bangladesh after pre-training on India.

A descriptive summary of the above datasets are shown in Table 1.

Table 1: Properties of 6 real datasets.

Dataset	# classes	# seqs.	Avg. length	# events	Data Type
Defi-Mainnet	6	20539	32	654844	Financial
Defi-Polygon	6	33597	85	2856453	Financial
Electronics	4	9993	20	195726	E-Commerce
Cosmetics	4	19301	39	752109	E-Commerce
ACLED-India	6	111	17	1934	Political
ACLED-Bangladesh	4	97	17	1697	Political

3 Model Implementation and (Pre)Training

Pretraining. Our pretraining model is based on the code adaptation from [2], which can be found in the Supplementary Material. The entire procedure is detailed in Algorithm 1 where a combined loss $\mathcal{L}_{event} = \mathcal{L}_{predict} + \lambda \mathcal{L}_{contrast}$ is minimized for each batch where $\mathcal{L}_{predict}, \mathcal{L}_{contrast}$ are from equation 3 and 4 respectively. We first randomly sample void event instances in-between every consecutive real events and then insert such void events. Then for each augment sequence, we randomly mask 15% of the events and split into train and dev subsets of sequences. We train our model using stochastic gradient descent and employ the Adam optimizer for optimization. The default transformer architecture used for pretraining includes the following specifications: the multi-headed self-attention module consists of 4 blocks, the dimension of the value vector after attention is applied is 512, the number of attention heads is 4, the hidden layer of the feed-forward neural network has a dimension of 1024, the value vector has a dimension of 512, the key vector has a dimension of 512, and the dropout rate is set at 0.1. Our training process spans 100 epochs with a learning rate of 0.0001. In all experiments, the hyper-parameters γ and λ are set to 1, and 0.01.The temporal parameter, τ, is fixed at 1. It is worth pointing out that equation 4 can be simplified to a more concise form by absorbing τ into λ in practice. The outputs of the transformer model, denoted as $\mathbf{H}_i$'s, are linearly mapped to numerical values and probabilities in the regression and classification losses presented in equation 3.

Fine-tuning. For the fine-tuning process, we employ a feed-forward network consisting of three hidden layers, each with a dimension of 512. The representations of the training and development subsets are acquired from the pre-trained model and are held constant throughout the fine-tuning stage. We utilize the Adam optimizer with a learning rate chosen from the range of $0.001, 0.002$, along with a batch size of 32, for a total of 100 epochs. To prevent overfitting, we implement early stopping if there is no improvement in the loss L_{event} for 10 consecutive epochs. The learning rate is determined based on the model's performance on the development subset of the target data. Furthermore, in all experiments, the value of α in equation 5 is fixed at 0.01.

Combined Position Encoding (PE) and Temporal Encoding (TE). We present a straightforward yet illustrative example demonstrating the effectiveness of the combined encoding scheme, which

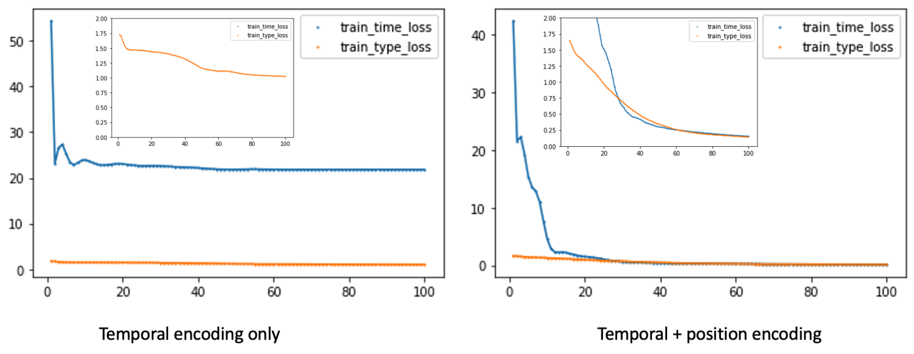

Figure 1: The effect of combined TE + PE in training.

Algorithm 1: Pretraining of Event-former

```
Input: Given dataset $S$ with $D$ sequences, each sequence $S_d$ with length $d_l$, $\{(t_i, y_i)\}_{i=1}^{d_l}$,
       batch size $b$
$S' = []$
for $d \leftarrow 1$ to $D$ do
    seq' = []
    for $i \leftarrow 1$ to $d_l - 1$ do
        $t'_i \sim Unif$(t$_i$,$t_{i+1}$)
        seq'.append(($t'_i$, null))
    end
    $seq_{new}$ = merge($S_d$, seq')
    S'.append( $seq_{new}$ )
end
for $d \leftarrow 1$ to $D$ do
    for $i \leftarrow 1$ to $d'_l - 1$ do
        mask $\sim Bern(0.15)$
        if $mask == 1$ then
            (0, null) $\leftarrow (t'_i, y'_i)$
        end
    end
end
split(S'$_m$) :=S'$_{m,tr}$ $= \{S'_{m,k}\}_{k=1}^{K}$, $S'_{m,dev}$
for $epoch \leftarrow 1$ to $N$ do
    for $iteration \leftarrow 1$ to $\lceil \frac{K}{b} \rceil$ do
        $B' \sim$S'$_{m,tr}$; Sample a batch of sequences
        compute $\mathcal{L}_{event}(B')$ (Eq. 3)
        back-propagate with gradient $\nabla_{\theta,\phi}\mathcal{L}_{event}(B')$ ( $\theta$: Transformer model parameters, $\phi$:
          Weights and bias for regression and classification.)
        update parameters of network $\theta, \phi$
    end
    evaluate $L_{event}(S'_{m,dev})$, stop training if not improving in 5 epochs
end
Return: Optimal parameters $\phi^*$, $\theta^*$
```

incorporates both position encoding and temporal encoding. In our experiment, we trained the model with only two samples, and the results clearly demonstrate improved overall training compared to using temporal encoding alone. Figure 1 visually depicts the compromised optimization when employing only temporal encoding.

4 Baselines Models and Implementation

In our study, we provide a brief overview of the implementation of the baseline models.

T-RMTPP and T-ERPP: The original Recurrent Marked Temporal Point Process (RMTPP) and Event Recurrent Point Process (ERPP) models are based on recurrent neural networks (RNNs) or their variants. Our implementation builds upon a standard codebase [2]. However, to ensure a fair comparison, we replaced the RNN components with transformers. We trained these models using the recommended set of parameters in our experiments.

T-LNM: The original Lognormal Mixture (LNM) model, like the previous models, is RNN-based. To model the density of log (inter-event) times, we utilized 64 mixtures of lognormal components while keeping other parameters at their default setting [3].

[2]https://github.com/woshiyyya/ERPP-RMTPP

[3]https://github.com/shchur/ifl-tpp

THP: The Transformer Hawkes Process (THP) models were implemented using attention-based transformers. We followed the recommended parameter settings for these models in our experiments.[4].

5 Proof Sketch of Theorem 1: Transformer with Combined Temporal Encoding and Position Encoding

We consider a general type of transformer architecture as described by Yun et al. (2019) in their work on transformers. Our proof demonstrates that a transformer with combined temporal encoding and position encoding can serve as a universal approximating function for any continuous function in the context of sequence-to-sequence tasks with compact support. This proof follows a similar structure to the proof of the transformer with position encoding (Theorem 3 in [3]).

Without loss of generality, let's consider a sequence with timestamps $t_1, t_2, \ldots, t_n$. We assume that t_i is an integer value for each $i \in 1, 2, 3, \ldots, n$, and that $t_i < t_j$ for all $i < j$. In case any t_i values are in decimal form, we can multiply them by a constant factor without affecting the dynamics of the event sequence, thereby transforming each given timestamp to an integer.

We define a d-dimensional temporal encoding for the n event epochs as follows:

$$\mathbf{T} = \begin{bmatrix} 0 & t_2 - t_1 & \ldots\, t_n - t_1 \\ 0 & t_2 - t_1 & \ldots\, t_n - t_1 \\ \vdots & \vdots & \vdots \\ 0 & t_2 - t_1 & \ldots\, t_n - t_1 \end{bmatrix}$$

Similarly, a d-dimensional position encoding for the n event epochs is defined as:

$$\mathbf{P} = \begin{bmatrix} 0 & 1 & \ldots\, n \\ 0 & 1 & \ldots\, n \\ \vdots & \vdots & \vdots \\ 0 & 1 & \ldots\, n \end{bmatrix}$$

The combined encoding is given by:

$$\mathbf{PE} + \mathbf{TE} = \begin{bmatrix} 0 & t_2 - t_1 + 1 & \ldots\, t_n - t_1 + 1 \\ 0 & t_2 - t_1 + 1 & \ldots\, t_n - t_1 + 1 \\ \vdots & \vdots & \vdots \\ 0 & t_2 - t_1 + 1 & \ldots\, t_n - t_1 + 1 \end{bmatrix}$$

The strict temporal ordering of the timestamps guarantees that $t_i - t_1 + 1 < t_{i+1} - t_1 + 1$ for all $i \in \{1, 2, 3, \ldots, n-1\}$. This ensures that for all rows, the coordinates are monotonically increasing, and the input values can be partitioned into cubes.

The remaining steps of the proof closely follow those outlined in the proof of Theorem 3 in [3]. This involves replacing n with t_n through quantization performed by feed-forward layers, contextual mapping achieved by attention layers, and function value mapping implemented by feed-forward layers.

6 Clarification, Limitations And Societal Impact

Additional rationale for pre-training and fine-tuning. In the context of dynamic event datasets, the role of labels extends beyond being mere targets for prediction, as observed in static image and text classification tasks. Here, labels in dynamic event datasets also serve as dynamic "features" in addition to their target role. Self-supervised representation learning emerges as a particularly effective approach for achieving a concise representation of the dynamic event history, utilizing event labels and timestamps as raw features. Our objective is to capitalize on a large volume of event sequences to acquire high-quality representations of event dynamics, especially for downstream

[4]https://github.com/SimiaoZuo/Transformer-Hawkes-Process

tasks with limited data availability. Consequently, learning the representation of historical context and transferring a sequence of such representations that encapsulate event dynamics differs from learning representations for tabular, image, or text data. A similar motivation can be observed in time series transfer learning [4], where self-supervised techniques demonstrate their effectiveness in acquiring compact dynamic features from raw historical values in a time series.

Advantages of homogeneous transfer. The homogeneous transfer problem, where pre-training on open domains and target task domains share the same set of event types, is not an oversimplified scenario. In fact, a significant portion of transfer learning research focuses on this type of transfer [5]. In our study, we also consider this scenario, where the event types are shared across pre-training and fine-tuning; however, the datasets themselves may differ, resulting in varying underlying dynamics between datasets. We have presented several practical applications in our paper, supported by experimental evidence and thorough exposition. In the case of alignment of event types, we utilize shared embedding matrices for event types during both pre-training and fine-tuning in our homogeneous transfer setting. Therefore, the challenge of aligning event types does not arise. However, it is worth noting that addressing the alignment issue becomes relevant for heterogeneous transfer, which is an avenue we are currently exploring.

Impact of self-supervised pretraining for TPP. Our self-supervised pretraining approach for temporal point process (TPP) models has certain considerations and potential societal implications that warrant attention. Firstly, the availability and representativeness of large-scale event streams, upon which our pretraining relies, may vary, posing challenges in accessing diverse real-world scenarios. Additionally, self-supervised training can be computationally demanding, necessitating substantial computational resources and time investments. Moreover, we acknowledge the potential risks associated with biases present in some data streams, which could lead to biased model predictions and the potential reinforcement of societal inequalities. Therefore, caution should be exercised when utilizing our model, and careful data curation and preprocessing are essential to mitigate such biases. Furthermore, the widespread adoption of self-supervised training for TPP models has implications for domains like healthcare and finance. To prevent potential negative consequences and societal harm, ethical considerations encompassing privacy protection, fairness, and transparency must be addressed. While self-supervised training holds promising opportunities, its limitations and societal impact demand meticulous consideration to ensure responsible harnessing of its potential and its beneficial application in real-world contexts.

References

[1] Debarun Bhattacharjya, Dharmashankar Subramanian, and Tian Gao. Proximal graphical event models. *Advances in Neural Information Processing Systems*, 31, 2018.

[2] Simiao Zuo, Haoming Jiang, Zichong Li, Tuo Zhao, and Hongyuan Zha. Transformer Hawkes process. In *International Conference on Machine Learning*, pages 11692–11702. PMLR, 2020.

[3] Chulhee Yun, Srinadh Bhojanapalli, Ankit Singh Rawat, Sashank J Reddi, and Sanjiv Kumar. Are transformers universal approximators of sequence-to-sequence functions? *arXiv preprint arXiv:1912.10077*, 2019.

[4] George Zerveas, Srideepika Jayaraman, Dhaval Patel, Anuradha Bhamidipaty, and Carsten Eickhoff. A transformer-based framework for multivariate time series representation learning. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pages 2114–2124, 2021.

[5] Fuzhen Zhuang, Zhiyuan Qi, Keyu Duan, Dongbo Xi, Yongchun Zhu, Hengshu Zhu, Hui Xiong, and Qing He. A comprehensive survey on transfer learning. *Proceedings of the IEEE*, 109(1):43–76, 2021.