HyperHawkes: Hypernetwork based Neural Temporal Point Process

Manisha Dubey, P.K. Srijith, Maunendra Sankar Desarkar

Abstract

Temporal point process serves as an essential tool for modeling time-to-event data in continuous time space. Despite having massive amounts of event sequence data from various domains like social media, healthcare etc., real world application of temporal point process faces two major challenges: 1) it is not generalizable to predict events from unseen sequences in dynamic environment 2) they are not capable of thriving in continually evolving environment with minimal supervision while retaining previously learnt knowledge. To tackle these issues, we propose HyperHawkes, a hypernetwork based temporal point process framework which is capable of modeling time of occurrence of events for unseen sequences. Thereby, we solve the problem of zero-shot learning for time-to-event modeling. We also develop a hypernetwork based continually learning temporal point process for continuous modeling of time-to-event sequences with minimal forgetting. In this way, HyperHawkes augments the temporal point process with zero-shot modeling and continual learning capabilities. We demonstrate the application of the proposed framework through our experiments on two real-world datasets. Our results show the efficacy of the proposed approach in terms of predicting future events under zero-shot regime for unseen event sequences. We also show that the proposed model is able to predict sequences continually while retaining information from previous event sequences, hence mitigating catastrophic forgetting for time-to-event data.

Introduction

Various applications in daily life like earthquake occurrences, social networks, financial transactions, user activity logs etc. are associated with collection of discrete asynchronous events where event occurrences are represented with timestamps. Each event sequence, consisting of a series of timestamps, is associated with a separate entity. For example, in social media, each user can be associated with the time of posting a tweet, and each tweet can be viewed as an event. Similarly, in financial transactions, each stock can be associated with the time of buy-sell order. The ability to model such sequences is of vital importance to create intelligent systems. These sequences often contain rich information, which can predict the future evolution of the sequences.

A principled mathematical framework to model such sequences in continuous time space is temporal point process (Valkeila 2008). Hawkes process (Hawkes 1971), a self exciting point process, has a rich literature in terms of theoretical importance and has been widely used in a wide array of practical applications like epidemic modeling (Diggle, Rowlingson, and Su 2005), earthquake prediction (Hainzl, Steacy, and Marsan 2010), financial modeling (Bacry, Mastromatteo, and Muzy 2015), crime prediction (Mohler et al. 2011) etc. Recent works improve the performance of standard classic Hawkes process by considering neural networks for modeling such event sequences (Mei and Eisner 2016; Du et al. 2016; Xiao et al. 2017; Omi, Ueda, and Alhara 2019). Neural Hawkes processes have proved to learn complex dependencies as against their classical counterparts. The Neural Hawkes process is one of the cornerstones of recent progress in time-to-event modeling.

Despite having improved performance, the neural Hawkes process is challenged by two potential limitations on the practical side. Neural Hawkes process typically needs to be trained on a large time-to-event dataset for the specific domain or entity. This restricts the prediction for a new and unseen entity with limited or no data and can be very crucial for certain applications. Moreover, the process of data acquisition for the time of occurrence of events for a new sequence is expensive. Besides, such a process can be time consuming for some sequences which may have low frequency of occurrence, and hence may take a long time to produce massive amounts of data which will be used to predict future time of occurrences. Secondly, real-world event occurrences happen sequentially in continuous streams. Therefore, a realistic and challenging problem is to continually learn time-to-event models in an ever-changing environment while retaining previous learnt knowledge.

Motivated by the above limitations, we consider a practical and under-explored setting for time-to-event modeling, called zero-shot event modeling. We also consider a continual learning setup where time-to-event prediction tasks arrive sequentially in an online manner. We aim to develop neural Hawkes process models which could generalize to time-to-event prediction tasks with no data and can continually learn while retaining previous knowledge. To this end, we introduce HyperHawkes, a hypernetwork based Hawkes process to generate sequence-specific parameters for the neural Hawkes process. Hypernetwork is essentially a metanetwork which can generate parameters for the neural
Hawkes process network for modeling continuous events. By employing hypernetwork based learning, we improve the model’s generalization ability to predict unseen sequences using the sequence descriptors. By incorporating descriptor-conditioned hypernetwork, we enable learning at the level of time-to-event sequence by learning event-sequence-specific parameters, hence being able to predict unseen sequences with the help of a descriptor. For a more pragmatic setup, we augment our model to consider continually arriving sequences where each sequence can be considered as a separate task. For continually learning the event sequences, we recast the descriptor-conditioned hypernetwork to include a hypernetwork output regularizer. This regularizer will penalize the changes in previously learnt parameters, hence retaining previously learnt time-to-event modeling capabilities. We provide two variations to the proposed approach, allowing to encompass 1) zero-shot modeling 2) continual learning capabilities within the framework of neural Hawkes process. To the best of our knowledge, there is no prior work on zero-shot or continually learning time-to-event modeling.

Our contributions can be summarized as follows:

- We propose two novel problems of zero-shot learning and continually learning from the paradigm of time-to-event modeling.
- We propose HyperHawkes, a descriptor-conditioned hypernetwork based neural Hawkes process which can generate event sequence specific parameters, hence learning at the level of sequence. We present two variants of HyperHawkes considering architecture of neural Hawkes process.
- The proposed methods can be used for predicting time of occurrences of unseen sequences, hence performing zero-shot time-to-event modeling.
- We augment the model with continual learning abilities by employing hypernetwork based regularization parameter, hence avoiding catastrophic forgetting for successively appearing time-to-event sequences.
- We present an experimental setup for evaluating zero shot learning and continual learning for time-to-event modeling. We demonstrate the effectiveness of the proposed models on these setups for two real-world datasets.

**Related Work**

**Hawkes Process**

Hawkes process (Hawkes 1971) is a point process (Valkeila 2008) with self-triggering property, i.e. occurrence of previous events trigger occurrences of future events. Hawkes process has been used in earthquake modeling (Hainzl, Steacy, and Marsani 2010), crime forecasting (Mohler et al. 2011), social media (Rizoiu et al. 2017) finance (Bacry, Mastromatteo, and Muzy 2015; Embrechts, Liniger, and Lin 2011) and epidemic forecasting (Diggle, Rowlingson, and Su 2005; Chiang, Liu, and Mohler 2021). They provide a solid mathematical framework for modeling event sequences. Earlier works on point process modeling specify a parametric form for the intensity function characterizing the point process. However, parametric models may not be capable of capturing the complex event dynamics. To address this, several research works were proposed (Du et al. 2016; Mei and Eisner 2017; Omi, Ueda, and Aihara 2019; Zuo et al. 2020) where the intensity function is modeled using neural networks, which are better at learning complex event dynamics. Recently, (Zuo et al. 2020) and (Zhang et al. 2020) proposed to use positional encodings in transformer language models (Vaswani et al. 2017) to model point processes. There are some efforts for learning from small data for Hawkes process (Xie et al. 2019; Salehi et al. 2019). However, they are based on the statistical Hawkes process model (not the neural Hawkes process) and are not applicable to a zero shot learning setting.

**Hypernetwork, Continual Learning and ZSL**

- **Zero-Shot Learning:** ZSL (Palatucci et al. 2009; Lampert, Nickisch, and Harmeling 2009) aims to predict classes which are not in training samples. Such classes are known as unseen classes and classes which are in training samples are known as seen classes. A few methods to address zero-shot learning is through mapping function (Frome et al. 2013), generative model (Felix et al. 2018) and graph neural network (Wang, Ye, and Gupta 2018). Huge literature has addressed the problem of zero-shot learning across various domains of vision, natural language processing tasks such as text classification, relation extraction etc.

- **Continual Learning:** aims to create a learning paradigm which is able to model a stream of tasks while avoiding catastrophic forgetting. Different techniques (Kirkpatrick et al. 2017; Li and Hoiem 2017; Lopez-Paz and Ranzato 2017; Von Oswald et al. 2019) have been proposed in this regard by consolidating knowledge either in various spaces like data, weight or meta space. (Von Oswald et al. 2019) performs continual learning through hypernetwork which learns task conditioned weights of base model.

- **Hypernetworks:** They have been introduced as a metanetwork which can generate weights for another network (Ha, Dai, and Le 2017). It is used for various tasks like meta learning (Zhao et al. 2020), neural architecture search (Zoph and Le 2016), natural language understanding (He et al. 2022) etc.

Despite having several literature in all these domains, to the best of our knowledge, there is no work in the intersection of zero-shot learning and time-to-event modeling. Also, there is no effort along the lines of continual learning for time-to-event modeling. Therefore, we address novel and essential problems in this direction which can benefit several applications.

**Preliminary**

**Problem Definition**

- **Zero-shot Learning for time-to-event modeling:** Assume we are given a collection of N seen sequences \( D^S = \{ (T^1, d^1), (T^2, d^2), ..., (T^N, d^N) \} \) where \( d^i \) represents the descriptor of the \( i^{th} \) sequence or meta-information and \( T^i \) represents the times of occurrence of \( n^i \) events in
the $i^{th}$ sequence, i.e. $\mathcal{T}^i = \{t_j^i\}_{j=1}^{n_i}$. Our goal is to predict
time of event occurrences for $N$ unseen sequences
$\mathcal{D}^U = \{(\mathcal{T}^1, d^1), (\mathcal{T}^2, d^2), ..., (\mathcal{T}^N, d^N)\}$ with the help
of the sequence descriptor.

- **Continual learning for time-to-event modeling:** Assume we are given a collection of $N$ sequences $\mathcal{D} = \{(\mathcal{T}^1, d^1), (\mathcal{T}^2, d^2), ..., (\mathcal{T}^N, d^N)\}$ where $d^i$ represents
the sequence descriptor and $\mathcal{T}^i$ represents the times of
occurrence of $n^i$ events in the $i^{th}$ sequence, i.e. $\mathcal{T}^i = \{t_j^i\}_{j=1}^{n_i}$ and we assume these sequences arrive one after the other in the order of their index. Our goal is to continually
learn the sequences while avoiding catastrophic forgetting
from the previous sequences. So, we aim to learn
a NHP model which will be able to predict the future
event occurrences in all the sequences ($\mathcal{T}^i, d^i$) where
$i \leq j$ after training on the $j^{th}$ sequence ($\mathcal{T}^j, d^j$).

### Hawkes Process

Point processes are useful to model the distribution of points
over some space and are defined using an underlying intensity
function. A Hawkes process [Hawkes 1971] is a point
process with self-triggering property i.e. occurrence of previ-
ous events trigger occurrences of future events. Conditional
intensity function for univariate Hawkes process at time $t_j$
for the $i^{th}$ sequence is defined as

$$\lambda(t_j^i) = \mu_i + \sum_{k=1}^{j-1} k(t_j^i - t_k^i)$$

(1)

where $\mu_i$ is the base intensity function and $k(\cdot)$ is the
triggering kernel function capturing the influence from previ-
ous events. The summation represents the effect of all events
prior to time $t_j^i$ which will contribute to computing the in-
tensity at time $t_j^i$. The probability density function at time $t_j^i$
given the past event times as $\{t_1^i, t_2^i, ..., t_{j-1}^i\}$, is obtained
as follows:

$$p(t_j^i|t_1^i, t_2^i, ..., t_{j-1}^i) = \lambda(t_j^i) \exp\left\{-\int_{t_{j-1}^i}^{t_j^i} \lambda(t)\,dt\right\},$$

(2)

where the exponential term in the right-hand side represents the
probability that no events occur in $[t_{j-1}^i, t_j^i]$.

### Neural Hawkes Process

Standard Hawkes process assumes a parametric form for the
intensity function which is not generalizable to every event
prediction problem. The influences between the events can be
complex and need not be exponentially decaying. Various
recent works introduced neural Hawkes processes [Du et al. 2016; Meil and Eisenet 2016; Omi, Ueda, and Aihara 2019] which models the intensity function as a nonlinear
function of history using a neural network. The central idea
of these works is to use recurrent neural networks (RNN)
to model intensity function which captures the influence of past events. So, conditional intensity function is modeled as:

$$\lambda(t_j^i) = f(h_j^i)$$

where $h_j^i$ represents a hidden state updated using RNN and
$f(\cdot)$ is a positive valued function ensuring positivity of the
intensity function.

### Proposed Model

We propose **HyperHawkes**, a hypernetwork based neural
Hawkes process for time-to-event modeling. For time-to-
event modeling, we consider the neural Hawkes process
(NHP) [Omi, Ueda, and Aihara 2019] as the base model.
Integrating Hypernetwork with neural Hawkes process, we
introduce descriptor-conditioned hypernetwork to generate
weights for each sequence which can perform time-to-event
modeling. The descriptor-conditioned hypernetwork learns
separate weights of NHP for each sequence. We leverage
this framework for zero-shot event modeling where the hy-
pernetwork produces weights for unseen tasks using the
sequence descriptor and NHP predicts future events. Inspired by [Von Oswald et al. 2019], we also use this framework for
continual learning of tasks by using a hypernetwork based
regularizer. We discuss each of these pieces in detail in fur-
ther subsections.

### Base Model: Neural Hawkes Process

In particular we employ neural Hawkes process [Omi, Ueda,
and Aihara 2019] as base model for time-to-event modeling.
It uses a combination of recurrent neural network and feed-
forward neural network to model the intensity function. We
represent history by using hidden representations generated by
recursive neural networks (RNNs) at each time step. The
hidden representation $h_j^i$ at time $t_j^i$ is obtained as

$$h_j^i = RNN(\tau_j^i, h_{j-1}^i; W_j^i) = \sigma(r_j^i V_j^i + h_{j-1}^i U_j^i + b_j^i)$$

(3)

where $\tau_j^i = t - t_j^i$ and $W_j^i$ represents the parameters associated with RNN for the $i^{th}$ sequence such as input weight matrix $V_j^i$, recurrent weight matrix $U_j^i$, and bias $b_j^i$. $h_j^i$ is obtained by repeated application of the RNN block on a se-
quence formed from previous $M$ inter-arrival times. This is
used as input to a feed-forward neural network to compute the
intensity function (hazard function) and consequently the
cumulative hazard function for computing the likelihood of event occurrences. In the proposed model, input to the
feed-forward neural network is I) the hidden representation
generated from RNN II) elapsed time from the most recent
event. We model the conditional intensity as a function of the
elapsed time from the most recent event as

$$\lambda(t_j^i H_j^i) = \lambda(t - t_j^i | h_j^i)$$

(4)

where $\lambda(\cdot)$ is a non-negative function referred to as a hazard function. Therefore, we define cumulative hazard function in terms of inter-event interval $\tau_j^i = t - t_j^i$ as $\Phi(\tau_j^i | h_j^i) = \int_0^{\tau_j^i} \lambda(s | h_j^i)\,ds$. Cumulative hazard function is modeled using
a feed-forward neural network (FNN)

$$\Phi(\tau_j^i | h_j^i) = FN\,NN(\tau_j^i, h_j^i; W_j^i).$$

(5)

However, we need to fulfill two properties of cumulative hazard function. Firstly, it has to be a monotonically increasing
function of $\tau_j^i$ and secondly, it has to be positive valued.
We achieve these by maintaining positive weights and posi-
tive activation functions in the neural network [Chilinski and
Silva 2020; Omi, Ueda, and Aihara 2019]. The hazard func-
tion itself can then be obtained by differentiating the cumu-
lative hazard function with respect to $\tau$ as

$$\lambda(\tau_j^i | h_j^i) = \frac{\partial}{\partial \tau_j^i} \Phi(\tau_j^i | h_j^i)$$

(6)
The log-likelihood of observing event times is defined as follows using the cumulative hazard function:

\[
\log p(\{t^i_j\}_{j=1}^{n^i}; W^i) = \sum_{j=1}^{n^i} \log p(t^i_j | h^i_{j-1}; W^i) = \sum_{j=1}^{n^i} \left( \log \frac{\partial}{\partial \tau^i_j} \Phi(\tau^i_j | h^i_{j-1}; W^i) - \Phi(\tau^i_j | h^i_{j-1}; W^i) \right)
\]

where \(\tau^i_j - \tau^i_{j-1}\) and \(W^i = \{W^i_r, W^i_f\}\) represents the combined weights associated with RNN and FNN. In NHP, the weights of the networks are learnt by maximizing the likelihood given by (7). The gradient of the log-likelihood function is calculated using backpropagation.

**HyperHawkes: Hypernetwork based Neural Hawkes Process**

Hypernetwork is a meta-network which produces parameters used by other networks (Ha, Dai, and Le 2017). As discussed in the above section, the neural Hawkes process comprises of two building blocks - RNN and FNN. So, we use hypernetwork to produce weights for these two components. We use a feed-forward neural network (FNN) to produce parameters \(W^i = \{W^i_r, W^i_f\}\) associated with the NHP. Since the nature of the RNN parameters \(W^i_r\) and FNN parameters \(W^i_f\) are different, we use two different types of hypernetworks, \(f_r(\cdot)\) producing \(W^i_r\) and \(f_f(\cdot)\) producing \(W^i_f\). Given a sequence description \(d^i\), parameters for the RNN are generated as follows:

\[
W^i_r = f_r(d^i; \theta_{f_r})
\]

where \(\theta_{f_r}\) denotes the parameters of the hypernetwork (weight vectors of a neural network). Note that the hypernetwork parameters are the same across the sequences. The descriptor \(d^i\) is used to generate the sequence specific parameters. As discussed above, the cumulative hazard function is a monotonically increasing function of \(\tau^i_j\) and is positive-valued. The hypernetwork which will generate parameters for cumulative hazard function has to fulfill these properties. This can be achieved when hypernetwork generates only positive weights for which we use a positive activation function.

Hypernetwork \(f_\ell(\cdot)\) for FNN can be written as:

\[
W^i_f = f_f(d^i; \theta_{f_f})
\]

where \(\theta_{f_f}\) denotes parameters of hypernetwork \(f_\ell(\cdot)\). We propose the following variants of HyperHawkes -

- **HyperHawkes-FNN**: Hypernetwork is considered only for the FNN modeling the cumulative hazard function.
- **HyperHawkes-FNN-RNN**: This variant uses two separate hypernetworks, one to model the RNN modeling the history and the second to model the FNN.

Overview of the proposed architecture is shown in Fig 1.
The difference in training lies in the fact that by maximizing this log-likelihood, we will get weights of the hyperparameter network rather than the neural Hawkes process. So, we calculate \( \theta_{fr} \) and \( \theta_{ft} \) using the gradient of the log-likelihood function using backpropagation.

**Prediction** For prediction of events from unseen sequence \( T^s \) from \( D^U \), we use our trained hypernetwork to produce weights \( \{W^c_u, W^a_u\} \) for the neural Hawkes process using sequence descriptor \( d^u \). Neural Hawkes process uses the bisection method (Omi, Ueda, and Aihara 2019) to predict the time of the next event. Bisection method provides the median \( t_s \) of the predictive distribution over next event time using the relation \( \Phi(t_s - t'_j | \theta^d_t; W^c_u, W^a_u) = \log(2) \).

**HyperHawkes for Continual Learning**

In a more realistic setup of event time modeling, sequences appear one after the other and it is unrealistic to store the data and models associated with all the previous sequences. In spite of this, we need to predict correctly on these past sequences though we have only data from new sequences. We want the NHP models to retain information from the past sequences while learning from new sequences. The standard training of the NHP model adapts them to the new sequence data (by updating parameters to optimize the loss to new sequence data) and results in forgetting what it has learnt from past sequences. The inability of the neural network models to retain knowledge from past data is known as catastrophic forgetting and continual learning techniques have been proposed to address this. Though it is studied in the vision community, to the best of our knowledge we could not find any work on the time-to-event prediction problem and with NHP models.

We address the novel problem of learning the time-to-event sequences continually while retaining knowledge from past time-to-event sequences. Inspired by (Von Oswald et al. 2019), we use descriptor conditioned hypernetworks for continually learning from event sequences. Ideally, we want our model to remember the parameters of the neural Hawkes process for each sequence. A naive approach to achieve this is through storing and replaying over previous data, which is obviously memory expensive and unrealistic. However, HyperHawkes, being conditioned on the sequence descriptor, can be modified to handle this problem. The direct use of the HyperHawkes training through (10) would result in hypernetworks forgetting the generation of the NHP parameters corresponding to past event sequences. We overcome this by incorporating a regularization on the hypernetwork parameters such that it penalizes any change to the NHP parameters produced from old sequences.

Given a sequence description \( d^s \) for the descriptor \( T^s \), our descriptor conditioned hypernetwork \( f_r(\cdot) \) can generate parameters \( W^c \) and \( f_l(\cdot) \) can generate parameters \( W^a \). To perform continual learning, we use regularization to penalize changes in \( \{W^c, W^a\} \) generated for past sequences in order to retain information from those sequences and to learn continually. The regularization is applied to the hypernetwork parameters while learning a new event sequence, and this prevents adaptation of the hypernetwork parameters completely to the new event sequence. For a new event sequence \( T^s \) and its corresponding descriptor \( d^s \), the hypernetwork parameters are learnt by minimizing the following continual learning loss over events in the sequence:

\[
\sum_{j=1}^{n_s} \log p(t_j^s | \mathcal{H}_j^s; (f_r(d^s); \theta_{fr}), f_l(d^s; \theta_{ft})) + \frac{\beta}{s-1} \sum_{c=1}^{s-1} \left( \| f_r(d^c; \theta_{fr}) - f_r(d^s; \theta_{fr}) \|^2 + \| f_l(d^c; \theta_{ft}) - f_l(d^s; \theta_{ft}) \|^2 \right)
\]

where \( \{\theta_{fr}, \theta_{ft}\} \) represents the stored hypernetwork parameters after learning until sequence \( s-1 \) and \( \{\theta_{fr}, \theta_{ft}\} \) represent the hypernetwork parameters learnt considering the event sequence \( s \) and regularization to avoid forgetting. The regularization term ensures that the newly learnt hypernetwork parameters will be able to produce the required main network parameters from the past event sequences given the sequence descriptor without forgetting and the regularization constant \( \beta \) captures the importance associated with it. So, in this way, we try to retain the information from previous sequences at a meta-level. By including a simple regularization term within the framework of HyperHawkes, our model is capable of learning sequences continually without forgetting knowledge learnt from previous sequences. We are able to achieve this because of the use of sequence-conditioned hypernetwork on the top of neural Hawkes process, emphasizing its usefulness for continual learning over event sequences in addition to zero shot learning.

**Experiments**

**Datasets**

Due to paucity of standard datasets for event modeling tasks which contain meta descriptions as well, we use the following two datasets: 1) **Yelp**[^1]. This is a dataset comprising of business information and their check-in information. Each business is associated with 82 attributes like Wheelchair Accessible, Accepts Insurance, By Appointment Only, Business Category, Business Timings etc. Also, they are associated with latitude-longitude pairs. Moreover, these businesses are associated with a fine-grained category. For higher granularity, we convert them into 22 broad categories using the hierarchy mentioned in their website[^2]. Using these attributes, we create a vector of length 1229 representing a business. This vector acts as a descriptor of the business. We select businesses with more than 5000 check-ins. For continual learning, we have considered another sample of dataset consisting of business with more than 10k checkins, hence considering 26 sequences of business. This is done to reduce the number of sequences for better visualization of performance of each sequence. 2) **Meme**[^3]. This dataset ([Leskovec, Backstrom, and Kleinberg 2009][^4]) tracks the popular phrases and quotes which appear most frequently over time in news media and blogs. Each meme is associated with the

[^1]: https://www.kaggle.com/datasets/yelp-dataset/
[^2]: https://www.yelp.com/developers/documentation/v3/all_category_list
[^3]: https://snap.stanford.edu/data/memetracker9.html
[^4]: https://snap.stanford.edu/data/memetracker9.html
content and timestamps when they were quoted in the media. We select top 200 English phrases and doc2vec representation of the meme content is considered as the descriptor of length 100. We have considered memes from April, 2009 with an average number of events as 970.

**Baselines**

To the best of our knowledge, the proposed problem statement is the first work along this direction. Therefore, we propose our own baselines as - **1) FNHP:** This includes fully neural Hawkes process (Omi, Ueda, and Aihara 2019). This approach doesn’t incorporate the sequence descriptor. **2) FNHP-Descriptor:** In this variant, we use concatenated descriptor and time as input to RNN and FNN. For Continual learning setup, we compare against HyperHawkes without any regularization as baseline.

**Implementation Details**

For zero-shot setup, we perform a 60-20-20 split where 60% of sequences are considered as seen sequence and 20% for validation unseen sequence and rest 20% as test unseen sequence. We have used a single layer with 32 units for hypernetwork where we use softplus activation function for modeling cumulative hazard function. For the neural Hawkes process, we consider a recurrent neural network with one layer and 16 units and 2-layer feed-forward neural network with 16 units in each layer. We use Adam optimizer with learning rate, $\beta_1$ and $\beta_2$ as 0.0001, 0.90 and 0.99 for the reported results. We perform single step lookahead prediction where we use actual time of occurrence of events as past events for the historical information to predict future events. We have performed all the experiments on Intel(R) Xeon(R) Silver 4208 CPU @ 2.10GHz, GeForce RTX 2080 Ti GPU and 128 GB RAM. We have implemented our code in Tensorflow 2.2.0 (Abadi et al. 2015). For continual learning setup, we test on regularization parameters in range [0.001-0.9]. For Yelp, reported results are for $\beta$ is set as 0.5 for HyperHawkes-FNN, 0.6 for HyperHawkes-FNN-RNN and for Meme $\beta$ is 0.6 for HyperHawkes-FNN and 0.1 for HyperHawkes-FNN-RNN. More implementation details and hyperparameter settings are presented in the supplementary.

**Experimental Setup**

We consider these experimental setups to evaluate the performance of our model - **1) Zero-Shot:** Training is done on seen sequences and testing is done on unseen sequences. **2) Generalized Zero-Shot:** In this, testing is done by randomly sampling 20% of events from seen sequences and unseen sequences. **3) Standard Event Modeling:** Training is done on the first 70% of the events from all sequences. Testing is done on the last 20% events for unseen sequences. Mean negative log-likelihood (MNLL) and mean absolute error (MAE) are considered as evaluation metrics for both zero-shot and continual learning setup. Lower MNLL and MAE indicates better performance.

**Results and Analysis**

**Zero-Shot Learning**

Results for zero-shot setup are presented in Table 1. A better model is expected to have lower MNLL and MAE. We can observe that the proposed method HyperHawkes performs better than the baselines in terms of both evaluation metrics MNLL and MAE. We can observe that zero-shot setup, which consists of predictive performance for unseen sequences, exhibits significantly lower MNLL and MAE for HyperHawkes-FNN as compared to FNHP and FNHP-Descriptor. Also, HyperHawkes-FNN-RNN yields lower MAE for Meme dataset, however, MNLL for both the variants of the proposed methods is close. Comparing the results of generalized zero-shot setup where we consider instances from seen sequences as well as unseen sequences, we can observe that HyperHawkes-FNN-RNN performs better for both the datasets. The final section of the table discusses results for standard event modeling where we test on the last 20% of the events of unseen sequences. HyperHawkes-FNN-RNN performs well here as well except for MAE for Yelp dataset. We can also observe that HyperHawkes-FNN-RNN performs slightly better for generalized zero-shot and standard event modeling setup. Moreover, it is interesting to note that inclusion of sequence descriptors within FNHP in FNHP-Descriptor has not helped much in prediction of unseen sequences. In fact, in various cases, it has performed worse than FNHP itself. This confirms the necessity of an event sequence specific framework which can predict better for unseen sequences as well. Therefore, our results indicate

| Experimental Setup | Dataset | MNLL | MAE |
|--------------------|---------|------|-----|
|                    |         | FNHP-Descriptor | HyperHawkes-FNN | HyperHawkes-FNN-RNN | FNHP-Descriptor | HyperHawkes-FNN | HyperHawkes-FNN-RNN |
| Zero-shot           | Yelp    | -4.2023 -3.3334 | -5.7045 | -5.4934 | 0.0025 0.0015 | 0.0013 0.0014 |
|                     | Meme    | -3.9011 -2.9421 | -5.4339 | -5.3403 | 0.0056 0.0072 | 0.0036 0.0018 |
| Generalized zero-shot | Yelp    | -4.9279 -4.2075 | -5.2473 | -5.6869 | 0.0025 0.0016 | 0.0021 0.0014 |
|                     | Meme    | -4.2735 -4.0250 | -5.2475 | -5.2922 | 0.0036 0.0033 | 0.0030 0.0026 |
| Standard event modeling | Yelp    | -4.2700 -3.8543 | -4.9030 | -4.9475 | 0.0047 0.0042 | 0.0024 0.0025 |
|                     | Meme    | -4.6991 -2.9633 | -2.5325 | -4.8796 | 0.0046 0.0072 | 0.0078 0.0027 |
that the proposed variants of HyperHawkes perform consistently and significantly better than the baselines for both the datasets.

**Continual Learning**

Table 2 presents the averaged results over all tasks by enabling HyperHawkes for continual learning. We can observe that averaged performance for the proposed model is better than the case when no regularization is incorporated. Also, we can observe that both the proposed variants perform better than the model without using regularization (corresponds to the case when $\beta$ from Equation 11 is set to 0). Hence the use of regularization within the framework of HyperHawkes supports that proposed method can avoid catastrophic forgetting. Fig 2 displays sequence-wise performance for both the datasets for the proposed variants HyperHawkes-FNN and HyperHawkes-FNN-RNN. 2a) displays average MNLL over previous sequences for Yelp, 2b) Average MAE over previous sequences for Meme, 2c) MNLL for each sequence for Yelp HyperHawkes-FNN, 2d) MNLL for different values of $\beta$ in Eq. 11 for HyperHawkes-FNN in Yelp and Meme.

In this work, we address two novel and practical limitations for time-to-event modeling. Firstly, we address zero-shot event modeling for predicting time of unseen events. Secondly, we propose an approach for continual learning for time-to-event modeling where we learn when event sequences continually and the model learns while retaining previous knowledge. To address both of these issues, we propose HyperHawkes, a descriptor conditioned hypernetwork based neural Hawkes process which can generate event sequence specific parameters. The proposed approach can predict the time of occurrences of events from unseen sequences, hence performing zero-shot time-to-event modeling. Subsequently, we augment HyperHawkes with regularization which can aid in learning time-to-event sequences continually by avoiding catastrophic forgetting. Our experiments on two real-world datasets demonstrate the effectiveness of the proposed approach for both the issues. In this way, we augment the ability of the neural Hawkes process to perform two unexplored and practical tasks of zero-shot and continually learning time-to-event modeling.

**Figure 2:** Plots displaying the performance of HyperHawkes for Continual Learning for the variant HyperHawkes-FNN and HyperHawkes-FNN-RNN. 2a) Average MNLL over previous sequences for Yelp, 2b) Average MAE over previous sequences for Meme, 2c) MNLL for each sequence for Yelp HyperHawkes-FNN, 2d) MNLL for different values of $\beta$ in Eq. 11 for HyperHawkes-FNN in Yelp and Meme.

**Conclusion**

In this work, we address two novel and practical limitations for time-to-event modeling. Firstly, we address zero-shot event modeling for predicting time of unseen events. Secondly, we propose an approach for continual learning for time-to-event modeling where we learn when event sequences continually and the model learns while retaining previous knowledge. To address both of these issues, we propose HyperHawkes, a descriptor conditioned hypernetwork based neural Hawkes process which can generate event sequence specific parameters. The proposed approach can predict the time of occurrences of events from unseen sequences, hence performing zero-shot time-to-event modeling. Subsequently, we augment HyperHawkes with regularization which can aid in learning time-to-event sequences continually by avoiding catastrophic forgetting. Our experiments on two real-world datasets demonstrate the effectiveness of the proposed approach for both the issues. In this way, we augment the ability of the neural Hawkes process to perform two unexplored and practical tasks of zero-shot and continually learning time-to-event modeling.
Xiao, S.; Yan, J.; Yang, X.; Zha, H.; and Chu, S. 2017. Modeling the intensity function of point process via recurrent neural networks. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 31.

Xie, Y.; Jiang, H.; Liu, F.; Zhao, T.; and Zha, H. 2019. Meta learning with relational information for short sequences. Advances in neural information processing systems, 32.

Zhang, Q.; Lipani, A.; Kiniap, O.; and Yilmaz, E. 2020. Self-attentive Hawkes process. In International conference on machine learning, 11183–11193. PMLR.

Zhao, D.; von Oswald, J.; Kobayashi, S.; Sacramento, J.; and Grewe, B. F. 2020. Meta-learning via hypernetworks.

Zoph, B.; and Le, Q. V. 2016. Neural architecture search with reinforcement learning. arXiv preprint arXiv:1611.01578.

Zuo, S.; Jiang, H.; Li, Z.; Zhao, T.; and Zha, H. 2020. Transformer hawkes process. In International conference on machine learning, 11692–11702. PMLR.