Neural Temporal Point Processes for Forecasting Directional Relations in Evolving Hypergraphs

Tony Gracious, Arman Gupta, Ambedkar Dukkipati

Department of Computer Science and Automation, Indian Institute of Science, Bangalore - 560012, INDIA
{tonygracious, arman, ambedkar}@iisc.ac.in

Abstract

Forecasting relations between entities is paramount in the current era of data and AI. However, it is often overlooked that real-world relationships are inherently directional, involve more than two entities, and can change with time. In this paper, we provide a comprehensive solution to the problem of forecasting directional relations in a general setting, where relations are higher-order, i.e., directed hyperedges in a hypergraph. This problem has not been previously explored in the existing literature. The primary challenge in solving this problem is that the number of possible hyperedges is exponential in the number of nodes at each event time. To overcome this, we propose a sequential generative approach that segments the forecasting process into multiple stages, each contingent upon the preceding stages, thereby reducing the search space involved in predictions of hyperedges. The first stage involves a temporal point process-based node event forecasting module that identifies the subset of nodes involved in an event. The second stage is a candidate generation module that predicts hyperedge sizes and adjacency vectors for nodes observing events. The final stage is a directed hyperedge predictor that identifies the truth by searching over the set of candidate hyperedges. To validate the effectiveness of our model, we compiled five datasets and conducted an extensive empirical study to assess each downstream task. Our proposed method achieves a performance gain of 32% and 41% compared to the state-of-the-art pairwise and hyperedge event forecasting models, respectively, for the event type prediction.

Code & Datasets — https://tinyurl.com/hypernodetpp
Extended version — https://arxiv.org/pdf/2301.12210

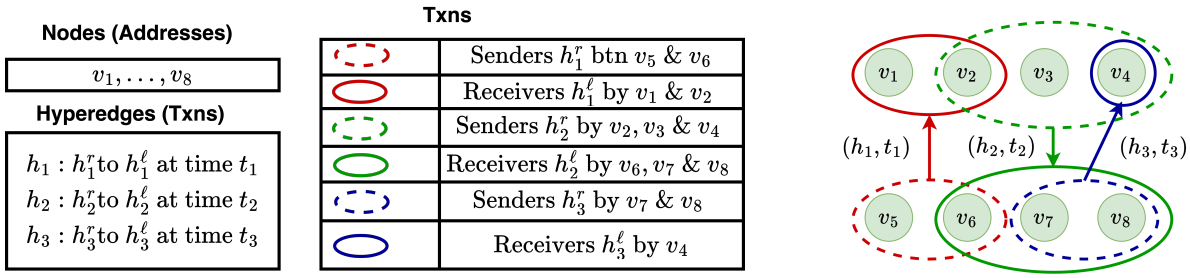
1 Introduction

The formation and evolution of real-world networks result from intricate relationships among entities. These relations can be characterized as (i) higher-order, (ii) directional, and (iii) temporal. For example, interactions via email involve a variable number of users engaged in directed relations that evolve with time. While there is a substantial body of literature focused on representation learning of networks, each study typically addresses these aspects in isolation. It is extremely challenging to study these networks in their most general form by considering all the above characteristics. This paper provides a comprehensive solution to this problem.

Temporal network representations entail learning a model that aggregates historical data into finite-dimensional vectors on each node or entity. These vectors can subsequently be utilized to predict future interactions among the entities. The previous research has characterized relationships as pairwise edge formations within a network, employing temporal graph neural network-based models to derive representations (S. Gupta and Dukkipati 2019; Trivedi et al. 2019; Xu et al. 2020; Liu, Ma, and Li 2022; Cao et al. 2021; Rossi et al. 2020; Zhou et al. 2022). However, this setting has very limited applicability as most real-world relations, as mentioned earlier, are higher-order with groups of entities involved in each relation and sizes of the groups can vary in a network. These problems have been modeled and studied as hypergraphs in a static case (Ghoshdastidar and Dukkipati 2017a,b). While (Gracious and Dukkipati 2023) is the first work that studies evolving hypergraphs, in this paper we study this problem in a very general setting considering that most real-world relations can be directional involving groups of interacting entities (Kim et al. 2022). Such examples can be seen in sports such as football or cricket, where two groups compete. Similarly, in citation networks, authors draw upon the works of other authors to support their arguments. Hence, in this work, we model models relations as edge formation in a directed hypergraph.

An example of relationships in a Bitcoin transaction network as directed hyperedges is depicted in Figure 1. Here, we can see that Bitcoin transactions involve two groups of entities: one group representing the senders and another group representing the receivers. The goal is to create models that can track the evolution of users in the Bitcoin network, represented by addresses, over time so that we can forecast transactions. Using learned dynamic representations, these models can then be used to detect anomalous transactions occurring within the network, such as money laundering (Wu et al. 2022).

Existing works on directed hyperedge prediction use hypergraph neural networks based unsupervised learning (Wei et al. 2022; Lee and Shin 2023; Bai, Zhang, and Torr 2021; Hwang et al. 2022). These models cannot be applied to the proposed problem as hypergraphs cannot be constructed



(a) Transactions (b) Model

Figure 1: Bitcoin transactions (Txns) are modeled as a temporal directed hypergraph graph with eight nodes to represent addresses, and three hyperedges to represent transactions. Here, t_i is the time with $t_3 > t_2 > t_1$, and a hyperedge is represented as a tuple $h_i = (h_i^r, h_i^\ell)$ with h_i^r the right hyperedge are the senders' addresses and h_i^ℓ the left hyperedge are receiver addresses.

from historical relations as they exist only for an instant of time. Alternately, set prediction-based scoring functions are used for hyperedge prediction. These models use a deep learning-based permutation invariant architecture to make predictions from node representations. These works are applicable only for undirected hyperedges (Zhang, Zou, and Ma 2019). and bipartite hyperedges (Sharma et al. 2021). These models can capture only self-connection information in the case of the undirected and cross-connection in the case of the bipartite. Unlike these, a directed hyperedge can represent three types of information: two self-connections among the left and right groups and a cross-connection between these groups. Now, to model the evolution of nodes, we use temporal node representations. Previous work on higher-order relation modeling uses it for forecasting undirected and bipartite hyperedges events (Gracious and Dukkipati 2023). These works achieve this using a sequential recurrent neural network-based model that updates node representations when an event occurs. One of the main disadvantages of this approach is that it will not allow the model to do batch processing of the data, as each sample depends on the previous samples. Further, for simulating the future in directed temporal graphs with nodes $\mathcal{V}$, one needs to compare all the ${}^{|\mathcal{V}|}\mathcal{P}_2$ pairwise combinations of nodes in the network. In the case of directed hypergraphs, there are a maximum of $2^{|\mathcal{V}|}$ combinations of nodes in the left and the right hyperedges.

In this paper, for the first time, we solve the problem of learning the representation of a higher-order, directional, and temporal network by proposing a model called *Directed HyperNode Temporal Point Process*, **DHyperNodeTPP**. Unlike previous works that forecast future events by an exhaustive search, we use a generative model based on the Temporal Point Process (TPP) to forecast candidate hyperedges (Hawkes 1971; Mei and Eisner 2017; Zuo et al. 2020). This is achieved by predicting event times for nodes followed by forecasting the projected adjacency matrix along with the distribution of hyperedge sizes of the nodes that are then used for generating candidate hyperedges. Further, to process batches of data, we use a temporal message-passing technique to store the features of events on each node and then use them to update the node representation before the next batch. Appendix H.1 shows the advantage of using batch processing to reduce the training time, enhancing model scalability to datasets with many samples. The following are the main contributions of our work. **(1)** A temporal point process model for forecasting directed hyperedges in a scalable way; **(2)** A temporal node representation learning approach that uses graph attention networks and batch processing; **(3)** A directed hyperedge prediction model; **(4)** Creation of five real-world temporal-directed hypergraph datasets from open-source data; **5)** Extensive experiments showing the advantage of our model on event forecasting over existing modeling techniques.

Related Works. The early works in temporal relation forecasting focus only on pairwise edge prediction. These involve discrete time models like DySAT (Sankar et al. 2020) and NLSM (Gracious et al. 2021), which divide the time into snapshots of uniform length and predict edges in the future snapshot by observing the history. Our method focuses on building continuous time models since discretization involves information loss and requires domain knowledge. JODIE (Kumar, Zhang, and Leskovec 2019) and TGAT (Xu et al. 2020) use temporal graph network to aggregate history into node representations, and these are trained to predict edge formation as a binary classification. Even though they can predict the presence of an edge at a particular time, they cannot estimate the time at which the event will occur. The TPP is a popular technique for modeling temporal networks that can do both. DyRep (Trivedi et al. 2019) and DSPP (Cao et al. 2021) use the TPP to model the time distribution of edge formation. Note that none of these above-mentioned works can model higher-order relations that are abundantly available in the real world. The importance of higher-order analysis is empirically shown in the work by Benson et al. (2018). They focus on the problem of higher-order relations between three nodes, mainly on the issue of distinguishing between the formation of the open triangle and close triangle relations. This is extended to a deep learning based method by Liu, Ma, and Li (2022). Recently, (Gracious and Dukkipati 2023) have addressed the problem of hyperedge forecasting using a TPP model, but this work does not consider the scalability issues and direction information in the relations.

2 Problem Definition

Hypergraph. A directed hypergraph is denoted by $\mathcal{G} = (\mathcal{V}, \mathcal{H})$, where $\mathcal{V} = \{v_1, v_2, \ldots, v_{|\mathcal{V}|}\}$ is the set of nodes, $\mathcal{H}$ is the set of valid hyperedges. In this, each hyperedge $h = (h^r, h^\ell) \in \mathcal{H}$ is represented by two subset of nodes, $h^r \subset \mathcal{V}$ the right hyperedge of size $k^r = |h^r|$, and $h^\ell \subset \mathcal{V}$ the left hyperedge of size $k^\ell = |h^\ell|$. The maximum size of right and left hyperedge is denoted by $k^r_{max} = \max_{h \in \mathcal{H}} |h^r|$ and $k^\ell_{max} = \max_{h \in \mathcal{H}} |h^\ell|$, respectively.

Temporal Events. The sequence of events occurring in a hypergraph till time $t \in \mathbb{R}^+$ is denoted by $\mathcal{E}(t) = \{(e_1, t_1), \ldots, (e_n, t_n), \ldots\}$. Here, event $e_n = \{h_{n,m}\}_{m=1}^{L_n}, h_{n,m} \in \mathcal{H}$, denotes the L_n concurrent hyperedges occurring at time $t_n \leq t$ with n as the index of the event and m as the index of the concurrent hyperedge. Then for each e_n, we create its projected adjacency matrices $\mathbf{A}^r_n, \mathbf{A}^\ell_n \in \{0,1\}^{|\mathcal{V}| \times |\mathcal{V}|}$ indicating the pairwise adjacency matrices for nodes in right hyperedge. In these, $\mathbf{A}^r_n[i,j] = 1$ if $\exists \{v_i, v_j\} \subset h^r, h = (h^r, h^\ell) \in e_n$ else 0, and $\mathbf{A}^\ell_n[i,j] = 1$ if $\exists\ v_i \in h^r$ and $v_j \in h^\ell, h = (h^r, h^\ell) \in e_n$ else 0. The rows of $\mathbf{A}^r_n$ and $\mathbf{A}^\ell_n$ denoted by $\mathbf{a}^r_{n,i}$ and $\mathbf{a}^\ell_{n,i}$, respectively, are the adjacency vectors of node v_i. In addition to this, we create the size matrices of right $\mathbf{K}^r_n \in \{0,1\}^{\mathcal{V} \times k^\ell_{max}}$ and left hyperedges $\mathbf{K}^\ell_n \in \{0,1\}^{|\mathcal{V}| \times k^r_{max}}$ with respect to the nodes in right hyperedge. Here, $\mathbf{K}^r_n[i,k] = 1$ if $\exists k = |h^r|$ and $\mathbf{K}^\ell_n[i,k] = 1$ if $\exists k = |h^\ell|\ \forall h \in e_{n+1}$ and $v_i \in h^r$. The size vectors for a node v_i is denoted by $\mathbf{k}^\ell_{i,n}$ and $\mathbf{k}^r_{i,n}$. These are the ith rows of respective matrices.

Goal. The aim is to learn the probability distribution $\mathrm{P}^*(e_{n+1}, t_{n+1}) = \mathrm{P}((e_{n+1}, t_{n+1}) | \mathcal{E}(t_n))$ over the time (t_{n+1}), and type (e_{n+1}) of the next event given the history, $\mathcal{E}(t_n)$. A naive implementation has the following likelihood, $\mathrm{P}^*(e_{n+1}, t_{n+1}) = \mathrm{P}^*(t_{n+1}) \prod_{h \in \mathcal{H}} [\mathrm{P}^*(h|t_{n+1})]^{\mathbb{I}_{h \in e_{n+1}}} \times [1 - \mathrm{P}^*(h|t_{n+1})]^{\mathbb{I}_{h \notin e_{n+1}}}$. This is obtained by assuming all the relations inside event e_{n+1} are independent, given the time of the event and history. The main challenge in generating samples based on the above likelihood is that the hyperedge prediction requires a search over a huge number of candidates, $|\mathcal{H}|$. Alternately, we can further reduce this by introducing a candidate generation model that will forecast likely hyperedges ($\mathcal{H}^c_{n+1}$),

$$\begin{aligned}&\mathrm{P}^*(e_{n+1}, t_{n+1}) \\ &= \mathrm{P}^*(\cup e^r_{n+1}, t_{n+1}) \times \mathrm{P}^*(\mathcal{H}^c_{n+1} | \cup e^r_{n+1}, t_{n+1}) \times \\ &\quad \prod_{h \in \mathcal{H}^c_{n+1}} [\mathrm{P}^*(h|t_{n+1})]^{\mathbb{I}_{h \in e_{n+1}}} [1 - \mathrm{P}^*(h|t_{n+1})]^{\mathbb{I}_{h \notin e_{n+1}}}. \quad (1)\end{aligned}$$

Let $\cup e^r_{n+1} = \cup_{m=1}^{L_n} h^r_{n+1,m}$ be the set of all the nodes in the right hyperedge. In this model, we first predict all the right nodes that are observing events at time t_{n+1}, followed by a candidate hyperedge generation module that outputs $\mathcal{H}^c_{n+1}$, and finally, we use the directed hyperedge predictor to get the ground truth. Here, we assume that the right hyperedge represents the source of the relation; for email exchange, the right node is the sender, and citation networks have the right nodes as the paper's authors. The entire block diagram for the proposed model is shown in Figure 2.

3 Model

We follow a sequential generative process by dividing the relation forecasting into three different modules that do time forecasting, candidate generation, and hyperedge prediction. These models use the temporal representations of nodes $\mathbf{V}(t) \in \mathbb{R}^{|\mathcal{V}| \times d}$ as input. Here d is the dimension of representation, and $\mathbf{v}_i(t) \in \mathbb{R}^d$ denotes the representation of node v_i. The architecture used to learn this is explained in Section 3.2. In the following section, MLP_*s are multilayer perceptron functions, and architectures for these are explained in Appendix C.

Node Event Model. Given the history, $\mathcal{E}(t_n)$, the task is to model the probability distribution of the time of occurrence of events on nodes. This is the product of likelihood of next event occurring at time $t_{n+1} = \Delta t_{n+1} + t_n$ for nodes in $\cup e^r_{n+1}$ and event not occurring for nodes not in $\cup e^r_{n+1}$ for the interval $[t_n, t_{n+1}]$. $\mathrm{P}^*(\cup e^r_{n+1}, t_{n+1}) = \prod_{v_i \in \cup e^r_{n+1}} \mathrm{P}^*_i(\Delta t_{n+1}) \prod_{v_i \notin \cup e^r_{n+1}} \mathrm{S}^*_i(\Delta t_{n+1})$. Here, S^*_i is the survival function that models the probability of events not occurring in the interval $[t_n, t_{n+1}]$. We use $\mathrm{Lognormal}$ distribution to model event times, $\Delta t_{n+1} \sim \mathrm{Lognormal}(\mu_{n+1,i}, s_t^2)$. Here, variance s_t is a hyperparameter, and $\mu_{n+1,i}$ is parameterized by a neural network that takes representation of v_i at time t_n as the input, $\mu_{n+1,i} = \mathrm{MLP}_t(\mathbf{v}_i(t_n))$. Then the log-likelihood becomes,

$$\begin{aligned}\mathcal{LL}^{n+1}_t = &\sum_{v_i \in \cup e^r_{n+1}} \frac{(\log(\Delta t_{n+1}) - \mu_{n+1,i})^2}{2 s_t^2} \\ &- \sum_{v_i \notin \cup e^r_{n+1}} \log\left(1 - \Phi\left(\frac{\log \Delta t_{n+1} - \mu_{n+1,i}}{s_t}\right)\right).\end{aligned}$$

Here, $\Phi(.)$ is the cumulative density function of the standard normal. The second loss component due to the survival function is approximated using negative sampling during training.

Candidate Generation. This is achieved by predicting the projected adjacency, $\mathbf{a}^r_{n+1,i}, \mathbf{a}^\ell_{n+1,i}$, and size vectors, $\mathbf{k}^r_{n+1,i}, \mathbf{k}^\ell_{n+1,i}$, for node v_i observing event from previous module, $v_i \in \cup e^r_{n+1}$. We use independent $\mathrm{Bernoulli}$ distribution to model the adjacency vectors of v_i, $\mathbf{a}^r_{n+1,i} \sim \mathrm{Bernoulli}(\sigma(\boldsymbol{\theta}^r_{n+1,i}))$, $\mathbf{a}^\ell_{n+1,i} \sim \mathrm{Bernoulli}(\sigma(\boldsymbol{\theta}^\ell_{n+1,i}))$. Here, $\boldsymbol{\theta}^r_{n+1,i} = \mathrm{MLP}_{ar}(\mathbf{v}_i(t_n))$ and $\boldsymbol{\theta}^\ell_{n+1,i} = \mathrm{MLP}_{a\ell}(\mathbf{v}_i(t_n))$. Similarly, we model the size distribution for node v_i, $\mathbf{k}^r_{n+1,i} \sim \mathrm{Bernoulli}(\sigma(\boldsymbol{\kappa}^r_{n+1,i}))$, $\mathbf{k}^\ell_{n+1,i} \sim \mathrm{Bernoulli}(\sigma(\boldsymbol{\kappa}^\ell_{n+1,i}))$. Here, $\boldsymbol{\kappa}^r_{n+1,i} = \mathrm{MLP}_{sr}(\mathbf{v}_i(t_n))$, and $\boldsymbol{\kappa}^\ell_{n+1,i} = \mathrm{MLP}_{sl}(\mathbf{v}_i(t_n))$. Then we can write log-likelihood for size and adjacency prediction as,

$$\begin{aligned}\mathcal{LL}^{n+1}_k = &\sum_{v_i \in \cup e^r_{n+1}, s=r,l} \langle \mathbf{k}^s_{n+1,i}, \log(\sigma(\boldsymbol{\kappa}^s_{n+1,i})) \rangle \\ &+ \langle (\mathbf{1} - \mathbf{k}^s_{n+1,i}), \log(\mathbf{1} - \sigma(\boldsymbol{\kappa}^s_{n+1,i})) \rangle, \quad (2)\end{aligned}$$

$$\mathcal{LL}_a^{n+1} = \sum_{v_i \in \cup e_{n+1}^r, s=r,\ell} \langle \mathbf{a}_{n+1,i}^s, \log(\sigma(\boldsymbol{\theta}_{n+1,i}^s) \rangle + \langle (\mathbf{1} - \mathbf{a}_{n+1,i}^s), \log(\mathbf{1} - \sigma(\boldsymbol{\theta}_{n+1,i}^s)) \rangle. \quad (3)$$

Hyperedge Predictor. Given the candidate hyperedges $\mathcal{H}_{n+1}^c$, the the probability of observing e_{n+1} at time t_{n+1} is, $\mathrm{P}^*(e_{n+1}|\mathcal{H}_{n+1}^c, t_{n+1}) = \prod_{h \in \mathcal{H}_{n+1}^c} \sigma(\lambda_h(t_{n+1}))^{\mathbb{I}_{h \in e_{n+1}}} (1 - \sigma(\lambda_h(t_{n+1})))^{\mathbb{I}_{h \notin e_{n+1}}}$. Here, $\lambda_h(t)$ is parameterized by a neural network that takes representations of the nodes in h at time t as input, $\lambda_h(t) = f(\{\mathbf{v}_i(t)\}_{v_i \in h^r}, \{\mathbf{v}_i(t)\}_{v_i \in h^\ell})$. The architecture of $f(.)$ is explained in Section 3.1. The log-likelihood is,

$$\mathcal{LL}_h^{n+1} = \sum_{h \in \mathcal{H}_{n+1}^c} \mathbb{I}_{h \in e_{n+1}} \log \sigma(\lambda_h(t_{n+1})) + \mathbb{I}_{h \notin e_{n+1}} \log(1 - \sigma(\lambda_h(t_{n+1}))). \quad (4)$$

While training, the candidate hyperedges are the combination of the true hyperedges and negative hyperedges generated by negative sampling. This is done by replacing either the left or right of true hyperedge with a hyperedge of randomly sampled size and filled with random nodes.

Loss Function. The complete likelihood for event sequence $\mathcal{E}(t_N)$ is, $\mathrm{P}(\mathcal{E}(t_N)) = \prod_{n=0}^{N-1} \mathrm{P}^*(e_{n+1}, t_{n+1})$. For training, we minimize the negative log-likelihood,

$$\mathcal{NLL} = -\left[\sum_{n=0}^{N-1} \mathcal{LL}_t^{n+1} + \mathcal{LL}_k^{n+1} + \mathcal{LL}_a^{n+1} + \mathcal{LL}_h^{n+1}\right].$$

3.1 Architecture of Hyperedge Predictor

We now propose an architecture that utilizes all three types of information in a directed hyperedge to predict true hyperedges from candidate hyperedges. Given a directed hyperedge $h = (h^r, h^\ell)$ with $h^r = \{v_{1^r}, \ldots, v_{k^r}\}$ and $h^\ell = \{v_{1^\ell}, \ldots, v_{k^\ell}\}$, we use the cross-attention layer (CAT) between the sets of nodes to create cross-dynamic hyperedge representation,

$$\mathbf{d}_{i^r}^{ch} = \mathrm{CAT}(\{\mathbf{v}_{1^r}(t), \ldots, \mathbf{v}_{k^r}(t)\}, \{\mathbf{v}_{1^\ell}(t), \ldots, \mathbf{v}_{k^\ell}(t)\}),$$

for node v_{i^r}. These dynamic hyperedge representations are added to the original representation to get $\mathbf{z}_{i^r}(t) = \mathbf{v}_{i^r}(t) + \mathbf{d}_{i^r}^{ch}$. Then we pass these through a self-attention layer (SAT) to get self-dynamic hyperedge representation, $\mathbf{d}_{i^r}^{sh} = \mathrm{SAT}(\{\mathbf{z}_{1^r}(t), \ldots, \mathbf{z}_{k^r}(t)\})$. The complete dynamic hyperedge representations are obtained by combining them, $\mathbf{d}_{i^r}^{h} = \mathbf{d}_{i^r}^{sh} + \mathbf{d}_{i^r}^{ch}$. We also create static hyperedge representation $\mathbf{s}_{i^r}^h = \mathbf{W}_{s^r} \mathbf{v}_{i^r}(t)$, where $\mathbf{W}_{s^r} \in \mathbb{R}^{d \times d}$ is a learnable parameter. Note that the terms dynamic and static in this section are used with respect to the hyperedge, not the time. Then we calculate the Hadamard power of the difference between static and dynamic hyperedge representation pairs followed by a linear layer, and the final score $\mathcal{P}^{h^r}$ is calculated as shown below,

$$\mathcal{P}^{h^r} = \frac{1}{k^r} \sum_{i^r=1}^{k^r} \mathbf{W}_{o^r}[(\mathbf{d}_{i^r}^h - \mathbf{s}_{i^r}^h)^2 || (\mathbf{d}_{i^r}^{ch} - \mathbf{s}_{i^r}^h)^2] + b_{o^r}.$$

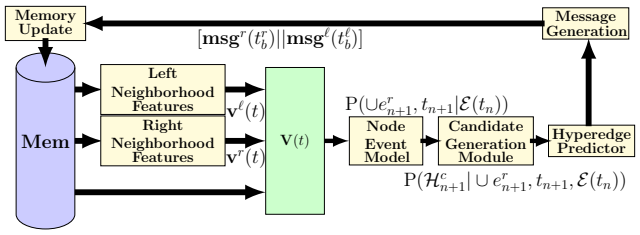


Figure 2: Neural Architecture of DHyperNodeTPP: We calculate the temporal node representation $\mathbf{V}(t)$ by combining the entries from the Memory module with information from recent relations where the node is involved in the left and right hyperedges. These temporal node representations are given as input to forecast nodes where events occur, followed by candidate hyperedge generation. Then the hyperedge prediction decoder in Section 3.1 is used to find the observed hyperedges.

Here, $||$ is the concatenation operator, $\mathbf{W}_{o^r} \in \mathbb{R}^{1 \times 2d}, b_{o^r} \in \mathbb{R}$ are the learnable parameters of the output layer. This equation models the cross and self connections in the right hyperedge. Similarly, we model left hyperedge to find $\mathcal{P}^{h^\ell}$ with a different set of parameters and combine them to predict links, $f(\{\mathbf{v}_i(t)\}_{v_i \in h^r}, \{\mathbf{v}_i(t)\}_{v_i \in h^\ell}) = \mathcal{P}^{h^r} + \mathcal{P}^{h^\ell}$. The model's entire block architecture and details of CAT and SAT are shown in Appendix D.

3.2 Temporal Node Representation

The temporal node representation of node v_i has the following form, $\mathbf{v}_i(t) = \tanh(\mathbf{W}_s \mathbf{Mem}_i + \mathbf{W}^r \mathbf{v}_i^r(t) + \mathbf{W}^\ell \mathbf{v}_i^\ell(t) + \mathbf{b}_v)$. Here, $\mathbf{W}_s \in \mathbb{R}^{d \times d}, \mathbf{W}^r \in \mathbb{R}^{d \times d}, \mathbf{W}^\ell \in \mathbb{R}^{d \times d}$ and $\mathbf{b}_v \in \mathbb{R}^d$ are learnable parameters. The first term, $\mathbf{Mem}_i \in \mathbb{R}^d$, is the historical events information for node v_i stored in the Memory Module, as explained in the following section. The second term, $\mathbf{v}_i^r(t)$, is the right neighborhood features calculated based on the recent higher-order relations h where the node v_i is present in the right hyperedge, $v_i \in h^r$. Similarly, we calculate $\mathbf{v}_i^l(t)$ based on the relations where a node is present in the left hyperedge. The architectures used for each of these components are shown below.

Memory Module. $\mathbf{Mem} \in \mathbb{R}^{|\mathcal{V}| \times d}$ stores the historical event information for each node in the network till time t. This module is initialized with zero values, and when a node is involved in a relation, its corresponding entries are updated based on the output of the Message Generation Module. However, if we update the memory with each event as done in the previous work, (Gracious and Dukkipati 2023), we will not be able to scale the model to a large number of samples, as batch processing of the data is not possible. So, we divide the events into batches of size $\mathcal{B}$ while maintaining their temporal order. Then, we aggregate the relation features of each node in the batch using the message generation stage, which is used to update the memory entries of the node using a recurrent neural network, as explained in the memory update stage.

Message Generation. We calculate features for each node in all the hyperedges in a batch $(h_b, t_b)_{b=1}^{\mathcal{B}}$. For node v_i in the right hyperedge h_b^r of (h_b, t_b), the features are the concatenation of its temporal node representation $\mathbf{v}_i(t)$, dynamic hyperedge representation $\mathbf{d}_i^h$, and Fourier features (Xu et al. 2020) calculated from the duration $t - t_i^p$ since the last memory update of that node at time t_i^p. This is stored as a message vector, $\mathbf{msg}_i^r(t_b) = \left[\mathbf{v}_i(t)||\mathbf{d}_i^h||\boldsymbol{\psi}(t - t_i^p)\right]$. Here, $\boldsymbol{\psi}(t - t_i^p) \in \mathbb{R}^d$ is the Fourier features based on functional encoding for the time $t - t_i^p$. This is achieved by learning a mapping function, $\boldsymbol{\psi}(t) = [\cos(\omega_1 t + \phi_1), \ldots, \cos(\omega_d t + \phi_d)]$ with parameters $\{\omega_i\}_{i=1}^d$, and $\{\phi_i\}_{i=1}^d$ inferred from the data. Similarly, we calculate the message vectors $(\mathbf{msg}^l(.))$ for nodes in the left hyperedge.

Memory Update. The messages from the previous section are used to update the memory entries of the corresponding nodes before the next batch. In our work, we use a GRU (Cho et al. 2014) based recurrent neural networks for memory updating as, $\mathbf{Mem}_i = \text{GRU}([\mathbf{msg}_i^r(t_b^r)||\mathbf{msg}_i^\ell(t_b^\ell)], \mathbf{Mem}_i)$, and $t_i^p = max(t_b^r, t_b^\ell)$. Here, t_b^r, t_b^ℓ are the latest event times for node v_i in previous batch. The corresponding features will have zero value for a node if the messages are unavailable on the left or right.

Neighborhood Features. To incorporate higher-order neighborhood information into the node representation, we use a graph attention network to update the node representation with features from relevant historical events. Further, it also helps to avoid the staleness of vectors in the Memory Module due to the absence of recent events involving the node by extracting information from other nodes that are previously involved with it (Kazemi et al. 2020). For a node v_i at time t, we find the recent $\mathcal{N}$ relations involving the node in the right hyperedge, and we denote them as $\mathcal{N}_{h^r}(t)$. Then for each hyperedge $(h, t) \in \mathcal{N}_{h^r}(t)$, we calculate the hyperedge representation as shown below, $\mathbf{h}^r(t_i) = \frac{1}{|h^r|}\sum_{v_{i^r} \in h^r} \mathbf{W}_h^r \mathbf{Mem}_{i^r} + \frac{1}{|h^\ell|}\sum_{v_{i^\ell} \in h^\ell} \mathbf{W}_h^\ell \mathbf{Mem}_{i^\ell}$. Similarly, we find the recent $\mathcal{N}$ relations where nodes are in the left hyperedge, $\mathcal{N}_{h^\ell}(t)$, and calculate hyperedge representation $\mathbf{h}^\ell(t_i)$. Then, the right neighborhood features are calculated using the graph attention layer as shown below,

$$\begin{aligned}
&\mathbf{C}(t) = [\mathbf{h}^r(t_1)||\boldsymbol{\psi}(t - t_1), \ldots, \mathbf{h}^r(t_\mathcal{N})||\boldsymbol{\psi}(t - t_\mathcal{N})], \\
&\mathbf{q}(t) = \mathbf{Mem}_i||\boldsymbol{\psi}(0), \\
&\mathbf{v}^r(t) = \text{MultiHeadAttention}(\mathbf{q}(t), \mathbf{C}(t), \mathbf{C}(t)). \qquad (5)
\end{aligned}$$

The $\text{MultiHeadAttention}$ uses the node memory vectors for query $(\mathbf{q}(t))$, and recent relation representation as keys $(\mathbf{C}(t))$ and values $(\mathbf{C}(t))$ (Vaswani et al. 2017). Similarly, we use a separate $\text{MultiHeadAttention}$ layer to calculate $\mathbf{v}^\ell(t)$.

4 Experiments

Datasets. Table 1 shows the statistics of both static and temporal directed hypergraph datasets (Yin et al. 2017; Chodrow and Mellor 2019; Gehrke, Ginsparg, and Kleinberg 2003; Wu et al. 2022; Jin et al. 2017). Here, $|\mathcal{V}|$ denotes the number of nodes, $|\mathcal{E}(T)|$ denotes the number of hyperedges, $|\mathcal{H}^r|$ denotes the number of unique right hyperedges, $|\mathcal{H}^l|$ denotes the number of unique left hyperedges, and T is the time span of the dataset. For the static datasets, there is no time span feature. A more detailed description of each dataset is provided in Appendix D.

Datasets	$\lvert\mathcal{V}\rvert$	$\lvert\mathcal{E}(T)\rvert$	$\lvert\mathcal{H}^r\rvert$	$\lvert\mathcal{H}^\ell\rvert$	T
Enron-Email	183	10,311	1,003	89	99,070
Eu-Email	800	208,403	11,897	744	69,459,254
Twitter	2,130	9,889	1,218	2,321	17,277
Hepth	451	9,882	8,384	1,352	21,532
ML-Arxiv	659	18,558	2,995	17,014	62,741
Bitcoin	7,806	231,071	23,901	6,706	1,416,317,420
iAF1260b	1,668	2,084	2,010	1,985	N/A
iJO1366	1,805	2,253	2,174	2,146	N/A
USPTO	16,293	11,433	6,819	6,784	N/A

Table 1: Datasets used for Temporal and Static Directed Hypergraphs along with their vital statistics.

Baseline Models. HyperNodeTPP is an undirected version of our model DHyperNodeTPP with right and left hyperedge merged into a single hyperedge, $h = h^r \cup h^\ell$. This uses the same temporal node representations with only a single type of neighborhood features and HyperSAGNN architecture for hyperedge prediction (Zhang, Zou, and Ma 2019). HGBDHE and HGDHE are models taken from previous work for higher-order relation forecasting by (Gracious and Dukkipati 2023). TGN and GAT are pairwise models developed for forecasting pairwise relations. TGN is developed for forecasting edges in a temporal graph (Rossi et al. 2020), and GAT (Veličković et al. 2018) forecast edges in a static graph.

We create two tasks to evaluate our model's event forecasting capabilities and to compare its performance against baseline methods. (i) Event Type Prediction: The goal of this task is to predict the type of event e_{n+1} occurring at time t_{n+1} given the history $\mathcal{E}(t_n)$. For evaluating the performance, we find the position of true hyperedges against candidate negative hyperedges by ordering them in descending order of $\lambda_h(t)$. Here, negative hyperedges are calculated by replacing the entire left or right hyperedges with hyperedges of randomly sampled nodes and sizes. Then MRR is calculated as follows, $\text{MRR} = \frac{1}{N}\sum_{n=1}^{N}\frac{1}{r_{n+1}}$. (ii) Event Time Prediction: The goal of this task is to predict the next time of event t_{n+1} given the history $\mathcal{E}(t_n)$ for the nodes of interest. It can be calculated using the time estimate $\Delta t_{n+1,i} = \exp(\mu_{n+1,i})$ for each node $v_i \in \cup e_{n+1}^r$. For evaluating the performance, we find the Mean Absolute Error (MAE) between true values t^{true} and estimated value $\hat{t}_{n,i}$ as, $\text{MAE} = \frac{1}{N}\sum_{n=1}^{N}\left[\sum_{v_i \in \cup e_n^r}\frac{1}{|\cup e_n^r|}|t_n^{\text{true}} - \hat{t}_{n,i}|\right]$. Additionally, the performance of predicting projected adjacency vectors, $\mathbf{a}_{n,i}^r, \mathbf{a}_{n,i}^\ell$, at time t is calculated by the proportion of

Methods		GAT	TGN	HGDHE	HGBDHE	HyperNodeTPP(ours)	DHyperNodeTPP(ours)
Enron-Email	MAE	N/A	N/A	35.77 ± 1.61	13.92 ± 0.35	$\mathbf{4.15 \pm 0.01}$	4.18 ± 0.02
	MRR	40.16 ± 7.15	42.22 ± 0.87	35.00 ± 3.94	36.09 ± 1.96	61.85 ± 0.01	$\mathbf{61.94 \pm 0.01}$
Eu-Email	MAE	N/A	N/A	20.58 ± 3.34	18.57 ± 1.62	12.23 ± 0.03	$\mathbf{12.22 \pm 0.02}$
	MRR	66.81 ± 0.02	69.15 ± 0.01	62.42 ± 1.79	55.34 ± 1.21	$\mathbf{75.95 \pm 0.01}$	68.05 ± 0.03
Twitter	MAE	N/A	N/A	21.58 ± 3.79	8.16 ± 0.70	$\mathbf{1.18 \pm 0.01}$	1.20 ± 0.01
	MRR	44.88 ± 0.05	55.20 ± 0.03	69.87 ± 0.72	70.19 ± 0.95	$\mathbf{84.47 \pm 0.01}$	82.12 ± 0.00
HepTh	MAE	N/A	N/A	16.19 ± 3.01	8.86 ± 0.08	1.25 ± 0.03	$\mathbf{1.20 \pm 0.01}$
	MRR	33.79 ± 8.95	51.70 ± 1.37	57.98 ± 0.82	57.40 ± 3.00	45.18 ± 0.02	$\mathbf{79.01 \pm 0.01}$
ML-Arxiv	MAE	N/A	N/A	29.94 ± 3.77	17.29 ± 0.57	1.25 ± 0.00	$\mathbf{1.24 \pm 0.00}$
	MRR	22.49 ± 4.31	37.58 ± 1.12	26.07 ± 0.29	28.13 ± 0.78	29.49 ± 0.00	$\mathbf{52.05 \pm 0.01}$
Bitcoin	MAE	N/A	N/A	109.71 ± 4.34	$\mathbf{63.16 \pm 1.55}$	75.98 ± 1.44	75.73 ± 2.60
	MRR	93.61 ± 4.02	93.34 ± 1.48	91.03 ± 0.00	85.06 ± 0.03	97.50 ± 0.09	$\mathbf{98.72 \pm 0.05}$

Table 2: Results in event type and time prediction tasks. The proposed model DHyperNodeTPP beats baseline models in almost all the tasks. Here, event type prediction is evaluated using MRR %; here, a higher value indicates better performance, and event time prediction is evaluated using MAE, the lower value indicating better performance.

true neighbors in the estimated adjacency vectors, $\hat{\mathbf{a}}^r_{n,i}, \hat{\mathbf{a}}^\ell_{n,i}$ as, Recall $= \frac{1}{2N}\sum_{n=1}^{N}\left[\sum_{v_i \in \cup e^r_n} \frac{1}{|\cup e^r_n|} \frac{\left(\mathbf{a}^r_{n,i}\right)^\intercal \hat{\mathbf{a}}^r_{n,i}}{(\mathbf{a}^r{}_{n,i})^\intercal \mathbf{a}^r_{n,i}}\right] + \frac{1}{2N}\sum_{n=1}^{N}\left[\sum_{v_i \in \cup e^r_n} \frac{1}{|\cup e^r_n|} \frac{\left(\mathbf{a}^\ell_{n,i}\right)^\intercal \hat{\mathbf{a}}^\ell_{n,i}}{(\mathbf{a}^\ell{}_{n,i})^\intercal \mathbf{a}^\ell_{n,i}}\right]$.

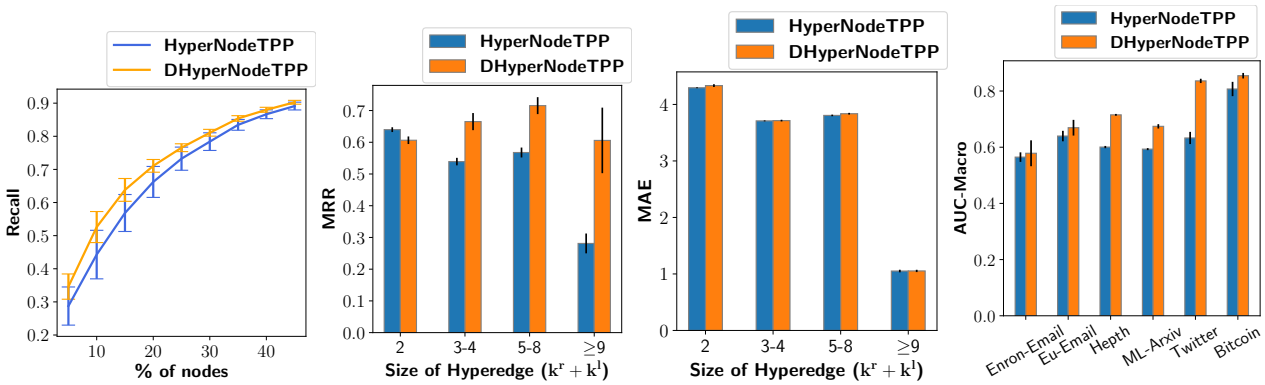
In all our experiments, we use the first 50% of hyperedges for training, 25% for validation, and the remaining 25% for testing. We use the Adam optimizer, and models are trained for 100 epochs with the batch size set to 128 and a learning rate of 0.001. Models have the same representation size of $d = 64$, and we use 20 negative hyperedges for each true hyperedge in the dataset. All the reported scores are the average of ten randomized runs, along with their standard deviation.

4.1 Results

Our proposed model, DHyperNodeTPP, is evaluated against previous works and an undirected baseline model, HyperNodeTPP. The results in Table 2 demonstrate that our models, HyperNodeTPP and DHyperNodeTPP, outperformed previous works, HGDHE and HGBDHE, in event time prediction. This superior performance can be attributed to the fact that our models are trained to predict the next event on the nodes, as opposed to previous models that are trained to predict the next event on a hyperedge, which is more difficult and prone to error as they have to predict events of longer duration. However, in our case, events on nodes are more frequent and have shorter periods, hence the smaller error in the event time prediction.

Further, one can observe that the temporal models perform much better than the static model GAT. This shows the need for temporal models for real-world relation forecasting. In our work, we achieve this using a memory based temporal node representation learning technique as explained in Section 3.2. We can also observe that models that use attention-based temporal representation, HyperNodeTPP, and DHyperNodeTPP, perform better than non-attentional models, HGBDHE and HGDHE, by comparing the performance in event type prediction tasks. This justifies using neighborhood features for temporal node representation learning. Additionally, we have discussed the limitations of this model in Appendix I. We have also included scalability experiments and a comparison against the state-of-the-art event forecasting model in Appendix H.1 and H.2, respectively.

Comparing directed and undirected models. To observe the advantage of directed modeling, we compare the directed model DHyperNodeTPP with the undirected model HyperNodeTPP. Here, we see an average increase of 70% in the MRR metric for event type prediction in citation network based datasets Hepth and ML-Arxiv. The proposed model gives comparable performance to the undirected model for the other datasets. This is because in these datasets, around 70% of relations have only a single node in the left hyperedge, and HyperNodeTPP is performing well on these hyperedges of size two ($k^r + k^l = 2$). Figure 3b shows for the Enron-Email dataset, HyperNodeTPP performs better than DHyperNodeTPP for hyperedges of size two, while for hyperedges of size greater than two, DhyperNodeTPP outperforms HyperNodeTPP. We have observed similar trends in Eu-Email, Twitter, and Bitcoin datasets, as shown in Figures 7b, 8b, and 11b. For Hepth and ML-Arxiv, the proposed directed model outperforms the undirected model in all hyperedge size groups, as shown in Figures 9b and 10b, respectively. For the task of event time prediction, both models performed similarly in all datasets, as observed in Figure 3c and Appendix G.1. This is because events are modeled on the nodes, not on the hyperedges. We also compared the models for the task of predicting projected adjacency vectors in the candidate hyperedge generation module. This is done by finding estimates for projected adjacency vectors by thresholding the estimated probability vectors, $\hat{\mathbf{a}}^r_{n,i} = \mathbb{I}_{\sigma(\boldsymbol{\theta}^r_{n,i}) > thres}$, $\hat{\mathbf{a}}^\ell_{n,i} = \mathbb{I}_{\sigma(\boldsymbol{\theta}^\ell_{n,i}) > thres}$, and recall is calculated. Here, $thres$ is selected so that a fixed percentage of nodes will be presented as neighbors. Figure 3a shows recall calculated for Enron-Email dataset with respect to the percentage of nodes by varying it from 5% to 50% in the step of 5%. Here, we can see DHyperNodeTPP has more recall than HyperNodeTPP, especially when the percentage of



(a) Adjacency forecasting (b) Event type prediction (c) Event time prediction (d) Hyperedge size prediction

Figure 3: Comparison of the performance of our directed and undirected model on different forecasting tasks. Figures 3a, 3b, 3c are on Enron-Email dataset. Here, we can observe that representation from DHyperNodeTPP performs better than HyperNodeTPP for adjacency forecasting. Furthermore, DhyperNodeTPP performs better than HyperNodeTPP for hyperedge sizes greater than two for the event type prediction. For event time prediction, both models perform equally, as events are modeled on nodes, and for hyperedge size prediction, directed models perform better. Hence, we can learn better representation using direction information.

Methods	HyperSAGNN	CATSETMAT	Ours
iAF1260b	60.1 ± 2.4	35.0 ± 1.4	$\mathbf{61.3 \pm 0.5}$
iJO1366	57.9 ± 1.8	36.2 ± 0.9	$\mathbf{59.0 \pm 1.8}$
USPTO	35.9 ± 1.8	37.8 ± 0.5	$\mathbf{38.1 \pm 0.6}$

Table 3: Performance of our hyperedge predictor in Section 3.1 compared to previous works on static directed hyperedge prediction. Here, the MRR metric is used to evaluate the performance. We can observe that our proposed hyperedge predictor performs better than previous models.

allowed neighbors is small and similar trends are observed in other datasets in Appendix G.3. Further, in the case of hyperedge size prediction, DHyperNode outperforms HyperNodeTPP in all the datasets, as shown in Figure 3d. The multi-label classification metric AUC-marco (Fawcett 2006) is calculated to evaluate this task. Here, we can see our model DHyperNodeTPP outperforms HyperNodeTPP considerably. There is an average improvement of 13.3% AUC macro metric.

Comparing hyperedge models with the pairwise models. The advantage of hyperedge models over pairwise edge models can be inferred by comparing models DHyperNodeTPP to TGN, which is the pairwise equivalent of DHyperNodeTPP. Our model DHyperNodeTPP has an average improvement of 31.8% in MRR metric over TGN in event type prediction.

Comparing different hyperedge prediction architectures. In Table 3, we compared our directed hyperedge predictor to previous works, HyperSAGNN (Zhang, Zou, and Ma 2019) for undirected hyperedge prediction, and CATSETMAT (Sharma et al. 2021) for bipartite hyperedge prediction. Our method performs considerably better than previous architectures. We get an average improvement of 3.7% over the undirected model and 46.6% over the bipartite model in the MRR metric. This is because our link predictor models self-connections in both right and left hyperedges and cross edges between them. The poor performance of CATSETMAT is due to its focus on modeling cross connections, and it fails to model self-connections in both right and left hyperedge due to the bipartite assumption.

5 Conclusion

In this work, we have proposed a comprehensive solution for the problem of learning representations for higher-order, directed, and temporal relations by presenting the model **DHyperNodeTPP**. Most of the previous works either reduce higher-order relations to pairwise edges (Xu et al. 2020; Trivedi et al. 2019; Kumar, Zhang, and Leskovec 2019; Gracious et al. 2021) or ignore the directionality information in the relations (Gracious and Dukkipati 2023). So, to address this, we propose a sequential generative process involving three stages. The first stage is used to forecast nodes where the events will occur, which are then consumed by the second task to forecast candidate hyperedges occurring at the event time. The second task is then used by the directed hyperedge prediction stage to filter out the true hyperedges. Since the number of nodes that observe an event at a particular time is very small, we could considerably reduce the search space for the possible hyperedge. Further, we provide a temporal node representation learning technique that can do batch training by using a Memory Module. This reduces computational complexity while training on very large datasets. We also provide an architecture for directed hyperedge prediction by combining three levels of information in the hyperedges. This work includes the creation of five datasets and demonstrate the advantage of the proposed model over existing models for undirected and bipartite hyperedge forecasting models.

Acknowledgements

The authors would like to thank the SERB, Department of Science, and Technology, Government of India, for the generous funding towards this work through the IMPRINT Project: IMP/2019/000383.

References

Bai, S.; Zhang, F.; and Torr, P. H. 2021. Hypergraph convolution and hypergraph attention. *Pattern Recognition*, 110: 107637.

Benson, A. R.; Abebe, R.; Schaub, M. T.; Jadbabaie, A.; and Kleinberg, J. 2018. Simplicial closure and higher-order link prediction. *Proceedings of the National Academy of Sciences*, 115: E11221–E11230.

Cao, J.; Lin, X.; Cong, X.; Guo, S.; Tang, H.; Liu, T.; and Wang, B. 2021. Deep structural point process for learning temporal interaction networks. In *Machine Learning and Knowledge Discovery in Databases. Research Track*, 305–320.

Cho, K.; van Merriënboer, B.; Gulcehre, C.; Bahdanau, D.; Bougares, F.; Schwenk, H.; and Bengio, Y. 2014. Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 1724–1734. Association for Computational Linguistics.

Chodrow, P.; and Mellor, A. 2019. Annotated hypergraphs: Models and applications. *Applied Network Science*, 5: 1–25.

Fawcett, T. 2006. An introduction to ROC analysis. *Pattern Recognition Letters*, 27(8): 861–874.

Gehrke, J.; Ginsparg, P.; and Kleinberg, J. 2003. Overview of the 2003 KDD Cup. *SIGKDD Explor. Newsl.*, 5(2): 149–151.

Ghoshdastidar, D.; and Dukkipati, A. 2017a. Consistency of Spectral Hypergraph Partitioning under Planted Partition Model. *The Annals of Statistics*, 45(1): 289–315.

Ghoshdastidar, D.; and Dukkipati, A. 2017b. Uniform Hypergraph Partitioning: Provable Tensor Methods and Sampling Techniques. *The Journal of Machine Learning Research*, 18(50): 1–41.

Gracious, T.; and Dukkipati, A. 2023. Dynamic representation learning with temporal point processes for higher-order interaction forecasting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, 7748–7756.

Gracious, T.; Gupta, S.; Kanthali, A.; Castro, R. M.; and Dukkipati, A. 2021. Neural Latent Space Model for Dynamic Networks and Temporal Knowledge Graphs. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, 4054–4062.

Hawkes, A. G. 1971. Spectra of some self-exciting and mutually exciting point processes. *Biometrika*, 58(1): 83–90.

Hwang, H.; Lee, S.; Park, C.; and Shin, K. 2022. AHP: Learning to Negative Sample for Hyperedge Prediction. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2237–2242.

Jin, W.; Coley, C.; Barzilay, R.; and Jaakkola, T. 2017. Predicting Organic Reaction Outcomes with Weisfeiler-Lehman Network. In *Advances in Neural Information Processing Systems*, volume 30.

Kazemi, S. M.; Goel, R.; Jain, K.; Kobyzev, I.; Sethi, A.; Forsyth, P.; and Poupart, P. 2020. Representation learning for dynamic graphs: A survey. *The Journal of Machine Learning Research*, 21(1): 2648–2720.

Kim, S.; Choe, M.; Yoo, J.; and Shin, K. 2022. Reciprocity in Directed Hypergraphs: Measures, Findings, and Generators. In *2022 IEEE International Conference on Data Mining (ICDM)*, 1005–1010.

Kumar, S.; Zhang, X.; and Leskovec, J. 2019. Predicting dynamic embedding trajectory in temporal interaction networks. In *Proceedings of the 25th ACM SIGKDD international conference on Knowledge discovery and data mining*.

Lee, D.; and Shin, K. 2023. I'm Me, We're Us, and I'm Us: Tri-directional Contrastive Learning on Hypergraphs. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Liu, Y.; Ma, J.; and Li, P. 2022. Neural Predicting Higher-Order Patterns in Temporal Networks. In *Proceedings of the ACM Web Conference 2022*, WWW '22, 1340–1351.

Mei, H.; and Eisner, J. M. 2017. The neural Hawkes process: A neurally self-modulating multivariate point process. In *Advances in Neural Information Processing Systems*, volume 30.

Rossi, E.; Chamberlain, B.; Frasca, F.; Eynard, D.; Monti, F.; and Bronstein, M. 2020. Temporal graph networks for deep learning on dynamic graphs. arXiv:2006.10637.

S. Gupta, G. S.; and Dukkipati, A. 2019. A generative model for dynamic networks with applications. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Sankar, A.; Wu, Y.; Gou, L.; Zhang, W.; and Yang, H. 2020. DySAT: Deep Neural Representation Learning on Dynamic Graphs via Self-Attention Networks. In *Proceedings of the 13th International Conference on Web Search and Data Mining*, 519–527.

Sharma, G.; Singh, S. P.; Devi, V. S.; and Murty, M. N. 2021. The CAT SET on the MAT: Cross Attention for Set Matching in Bipartite Hypergraphs. arXiv:2111.00243.

Trivedi, R.; Farajtabar, M.; Biswal, P.; and Zha, H. 2019. DyRep: Learning Representations over Dynamic Graphs. In *International Conference on Learning Representations*.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, L.; and Polosukhin, I. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30.

Veličković, P.; Cucurull, G.; Casanova, A.; Romero, A.; Liò, P.; and Bengio, Y. 2018. Graph Attention Networks. In *International Conference on Learning Representations*.

Wei, T.; You, Y.; Chen, T.; Shen, Y.; He, J.; and Wang, Z. 2022. Augmentations in Hypergraph Contrastive Learning: Fabricated and Generative. In *Advances in Neural Information Processing Systems*.

Wu, J.; Liu, J.; Chen, W.; Huang, H.; Zheng, Z.; and Zhang, Y. 2022. Detecting Mixing Services via Mining Bitcoin Transaction Network With Hybrid Motifs. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 52(4): 2237–2249.

Xu, D.; Ruan, C.; Korpeoglu, E.; Kumar, S.; and Achan, K. 2020. Inductive representation learning on temporal graphs. In *International Conference on Learning Representations*.

Yin, H.; Benson, A. R.; Leskovec, J.; and Gleich, D. F. 2017. Local Higher-Order Graph Clustering. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

Zhang, R.; Zou, Y.; and Ma, J. 2019. Hyper-SAGNN: a self-attention based graph neural network for hypergraphs. In *International Conference on Learning Representations*.

Zhou, H.; Zheng, D.; Nisa, I.; Ioannidis, V.; Song, X.; and Karypis, G. 2022. TGL: A general framework for temporal gnn training on billion-scale graphs. *Proc. VLDB Endow.*, 15: 1572–1580.

Zuo, S.; Jiang, H.; Li, Z.; Zhao, T.; and Zha, H. 2020. Transformer Hawkes process. In *Proceedings of the 37th International Conference on Machine Learning*, 11692–11702.