# Dynamic Representation Learning for Higher-Order Interaction Forecasting in Networks

**Anonymous Author(s)**
Affiliation
Address
`email`

## Abstract

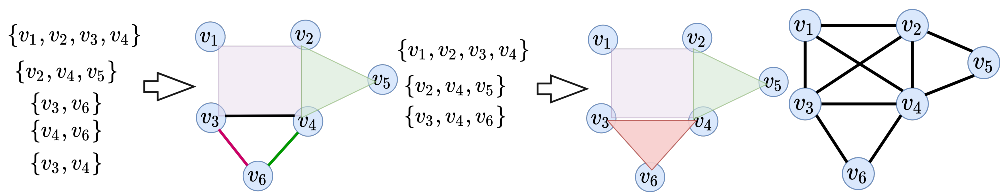
The explosion of digital information and the growing involvement of people in social networks led to enormous research activity to develop methods that can extract meaningful information from interaction data. Commonly, interactions are represented by edges in a network or a graph, which implicitly assumes that the interactions are pairwise and static. However, real-world interactions deviate from these assumptions: (i) interactions can be multi-way involving more than two nodes or individuals (e.g., family relationships, protein interactions), and (ii) interactions can change over a period of time (e.g., change of opinions and friendship status). While pairwise interactions have been studied in a dynamic network setting and multi-way interactions have been studied using hypergraphs in static networks, there exists no method that can predict multi-way interactions or hyperedges in dynamic settings. Existing related methods cannot answer temporal queries like what type of interaction will occur next and when it will occur. This paper proposes a temporal point process model for hyperedge prediction to address these problems. Our proposed model uses dynamic representation techniques for nodes in a neural point process framework to forecast hyperedges. We present several experimental results and set benchmark results. As far as our knowledge, this is the first work that uses the temporal point process to forecast hyperedges in dynamic networks.

## 1 Introduction

Learning from temporal interactions between entities to extract meaningful information and knowledge is of paramount importance. For example, learning how a person interacts on social media can provide knowledge about that person's preferences, and it can help in recommending items to that person. Similarly, an e-commerce website can better understand the users' needs if it efficiently extracts knowledge from users' consumption history. Previously these problems have been studied using representation or embedding learning in dynamic networks where interactions were modeled as instantaneous links or edges between two nodes formed at the time of interaction [17; 16]. For this, Temporal Point Processes (TPP) [6] have been introduced for modeling edge formation in dynamic networks. TPPs are stochastic processes that model localized events in time, and the events can be of multiple types. To model dynamic networks, one represent each edge as an event type, and a probability distribution is defined over the time of its formation. Here, the probability distribution is parameterized using an intensity function based on representations of nodes. These node representations are functions of time and past interaction events. The parameters of these functions are learned by minimizing the negative log-likelihood of the interactions in the training data.

However, most real-world interactions are more complex than just pairwise interactions. For example, a person can have multiple items in a single shopping order, a group of people can co-author an article, mutual funds have stocks of various companies, and so on. A common technique that is

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.




(a) Hyperedges at time $t$ (b) Hyperedges at time $t$ (c) Projected pairwise graph for Hypergraphs in 1a and 1b

Figure 1: Higher-order interactions at time $t$ are shown as hyperedges in Figures 1a and 1b. Here, hyperedges are represented by geometric shapes with their ends/corners showing the nodes and color showing their identity. We can see two different hypergraphs having the same projected graph in Figure 1c. So, we need a technique that predicts hyperedges without projecting them to a pairwise graph.

employed to deal with this problem is to approximate these multiway interactions with pairwise interactions. That amounts to approximating hypergraphs with graphs, leading to enormous information loss. This is demonstrated in Figure 1, where two different kinds of interaction between nodes $\{v_1, v_2, v_3, v_4, v_5, v_5\}$ have the same pairwise interaction graph. Further, it is impossible to infer the original interactions once they are projected into a pairwise graph. Hence, in this paper, we address the problem of forecasting higher-order interactions as hyperedge events using TPP. We define a conditional intensity function on each hyperedge that takes node representations as inputs. Since each hyperedge can have a variable number of nodes, we use a self attention-based architecture for hyperedge encoding. Further, for learning node representation, we need to consider that the nodes evolve as they interact, so one needs to have a dynamic node representation. In earlier works on dynamic networks [22; 5; 3], node embeddings are updated based on the node embedding of the other node in the interaction. For example, consider edge event $(v_a, v_b)$ occurring at time $t$. To update node embeddings of $v_a$, we use the node embeddings of $v_b$ and vice versa. However, hyperedge events have a variable number of nodes (Figure 1), so the techniques developed for pairwise interaction are not directly applicable for higher-order interaction. Hence we use a self-attention-based encoding for node update with parameters shared with the hyperlink prediction model to solve this.

**Contributions.** We propose a model called *Hypergraph Dynamic Hyperedge* (HGDHE) to model higher-order interactions as hyperedge events in a dynamic Hypergraph. Further, these interactions will not always be between a homogeneous set of nodes. So, we also created a bipartite hyperedge variant of our model called *Hypergraph Bipartite Dynamic Hyperedge* (HGBDHE). This will help in modeling interactions between two different types of nodes. Our contributions are as follows. **(1)** A temporal point process framework for hyperedge modeling that can forecast the type and time of interaction; **(2)** A model for representation learning for higher-order interaction data; **(3)** Extensive experiments on real-world datasets on both homogeneous and bipartite hyperedges; **(4)** Empirical results on performance gain obtained when we use hyperedges instead of pairwise modeling; and **(5)** Empirical results on performance gain obtained when we use dynamic models instead of static models.

**Related Works.** Earlier works in modeling temporal information into networks can be categorized as (i) discrete-time models, and (ii) continuous-time models. In discrete-time models, time is discretized into bins of equal size, and recurrent neural network based models are used for modeling temporal evolution [11; 10]. Since discretization results in information loss and selecting bin size is a difficult task, the recent focus has been on continuous-time models [16; 4]. Unlike these discrete-time models, TPP-based continuous-time models can predict both dynamic interaction and time of interaction. Recently neural network based TPP has been proposed to model the dynamic interaction. However, these works approximate higher-order interactions with pairwise interactions [5; 22; 3] It has been shown in Hyper-SAGNN [24] that directly modeling the higher-order interaction will result in better performance than decomposing them into pairwise interactions.

Higher-order interaction between nodes can be modeled as link prediction in a hypergraph. Earlier works use matrix completion techniques to predict hyperedge. Coordinated Matrix Minimization (CMM) [23], infer the missing hyperedges in the network by modeling adjacency matrix using a non-negative matrix factorization based latent representation. Recent works mainly concentrate on neural network-based scoring functions as they perform better than matrix completion-based techniques and are easier to train. Hyperpath [13] model's hyperedge as a tuple and uses a neural network-based scoring function to predict links. This method cannot model higher-order interactions as it expects the hyperedge size to be uniform for all edges. HyperSANN [24] uses a self-attention based architecture for predicting hyperlinks. It can learn node embeddings and predict non-uniform hyperlinks. However, it is a static model and cannot model the dynamic nature of hyperedges. The work presented in this paper is the first work that uses TPPs to forecast hyperedges when networks are evolving with time.

## 2 Dynamic Hyperedge Forecasting

Given a set of nodes $\mathcal{V} = \{v_1, v_2, \ldots, v_{|\mathcal{V}|}\}$, an interaction event between a subset of these nodes is modeled as a hyperedge $(h_i)$ with the time of interaction as its edge attribute. Here, $h_i \in \mathcal{H}$ is a subset of nodes in $\mathcal{V}$ and $\mathcal{H}$ is set of all valid combination of nodes. Given the historical events $\mathcal{E}(t_m) = \{(h_1, t_1), , \ldots, (h_m, t_m)\}$ till time $t_m$, aim is to forecast future hyperedge $h_i$ at time $t > t_m$.

### 2.1 Hyperedge Event Modeling

Given hyperedge $h = \{v_1, v_2, \ldots, v_k\}$, the probability of that hyperedge occurring at time $t$ can be modeled using TPP with conditional intensity $\lambda_h(t)$ as

$$p_h(t) = \lambda_h(t)\mathcal{S}_h(t), \tag{1}$$

where $\mathcal{S}_h(t)$ is survival function that denotes probability that no event happened during the interval $[t_v^p, t)$ for hyperedge $h$. This is defined as

$$\mathcal{S}_h(t) = \exp\left(\int_{t_h^p}^{t} -\lambda_h(\tau)d\tau\right), \tag{2}$$

where $t_h^p = \max_{v\in h} t_v^p$ and $t_v^p$ denotes the most recent interaction time of node in $h$. We provide more detailed background explanation of TPP in Appendix A. The conditional intensity function $\lambda_h(t)$ is parameterized by a defining positive function over the embeddings of nodes in $h$ as,

$$\lambda_h(t) = f(\boldsymbol{v_1}(t), \boldsymbol{v_2}(t), \ldots, \boldsymbol{v_k}(t)). \tag{3}$$

Here $f(.) \geq 0$ can be realized by neural network for hyperedge events, and $\boldsymbol{v_i}(t) \in \mathbb{R}^d$ is the node embeddings at time $t$ for node $v_i$. We follow the same architecture of hyperedge modeling technique as Hyper-SAGNN [24] with a final softplus layer as explained in Appendix B.1. We also created baseline models using piece-wise constant node embeddings with intensity defined by Rayleigh Process to create an equivalent model for DeepCoevolve [5] in hyperedge interactions, as shown below,

$$\lambda_h(t) = f(\boldsymbol{v_1}(t_h^p), \boldsymbol{v_2}(t_h^p), \ldots, \boldsymbol{v_k}(t_h^p))(t - t_h^p). \tag{4}$$

This formulation will give us a closed-form solution for survival function, thereby for the probability of the event. Otherwise, integration has to be approximated by sampling.

### 2.2 Dynamic Node Representation

For each node in the network, we learn a low dimensional embedding $\boldsymbol{v}(t) \in \mathbb{R}^d$ that changes with time. It is done through three stages, i) Temporal Drift, ii) History Aggregation, and iii) Interaction Update, as shown below,

$$\boldsymbol{v}(t) = \tanh(\underbrace{\boldsymbol{W_0}\boldsymbol{v}(t_v^{p+})}_{\textbf{Interaction Update}} + \underbrace{\boldsymbol{W_1}\boldsymbol{\Phi}(t - t_v^p)}_{\textbf{Temporal Drift}} + \underbrace{\boldsymbol{W_2}\boldsymbol{v^s}(t_{i-1})}_{\textbf{History Aggregation}} + \boldsymbol{b_0}). \tag{5}$$

Here, $\boldsymbol{W_0}, \boldsymbol{W_1}, \boldsymbol{W_2} \in \mathbb{R}^{d\times d}, \boldsymbol{b_0} \in \mathbb{R}^d$ are learnable parameters, $\boldsymbol{v}(t_v^{p+})$ is the node embedding just after previous interaction for node $v$ at time $t_v^p$ and $t_{i-1}$ is the last event time, $t_{i-1} < t$.

**Temporal Drift.** This term models the inter-event evolution of a node with time. For a node $v$ with previous event time $t_v^p$, the drift in embedding at time $t$ is modelled by $\boldsymbol{W_1}\boldsymbol{\Phi}(t - t_v^p)$. Here, $\boldsymbol{\Phi}(t) \in \mathbb{R}^d$ is Fourier time features [4; 3] defined as $\boldsymbol{\Phi}(t) = [\cos(\omega_1 t + \theta_1), \ldots, \cos(\omega_d t + \theta_d)]$. Here, $\{\omega_i\}_{i=1}^d$, and $\{\theta_i\}_{i=1}^d$ are learnable parameters.

**History Aggregation.** This stage uses hypergraph convolution based feature aggregation to incorporate the effect of past events. For this, we will construct a hypergraph using the past $M$ events, $\{(h_{i-M}, t_{i-M}), \ldots, (h_{i-1}, t_{i-1})\}$, and uses its incidence matrix $\mathbf{H}(t_{i-1}) \in \mathbb{R}^{|V| \times M}$ to apply hypergraph convolution as $[\boldsymbol{v_1^s}(t_{i-1}), \ldots, \boldsymbol{v^s_{|\mathcal{V}|}}(t_{i-1})] = \mathrm{HGNN}\left(\mathbf{H}(t_{i-1}), \left[\boldsymbol{v_1}(t^p_{v_1}{}^+), \ldots, \boldsymbol{v_1}(t^p_{v_{|\mathcal{V}|}}{}^+)\right]\right)$. Here, HGNN is a hypergraph graph convolution defined as in Bai et al. [1].

**Interaction Update.** When a node $v$ is involved in an interaction $h$, it is influenced by the nodes it interacts within $h$. For extracting features of interaction, we use the dynamic embedding $\boldsymbol{d_v^h}$ calculated as a function of embeddings of nodes $h - \{v\}$ at time t. The architecture of calculating this is shared with conditional intensity function as shown in Equation 13 in Appendix B.1.The entire update equation is $\boldsymbol{v}(t^+) = \tanh(\boldsymbol{W_3}\boldsymbol{v}(t_v^{p+}) + \boldsymbol{W_4}\boldsymbol{\Phi}(t - t_v^p) + \boldsymbol{W_5}\boldsymbol{d_v^h} + \boldsymbol{b_1})$. Here, $\boldsymbol{W_3}, \boldsymbol{W_4}, \boldsymbol{W_5} \in \mathbb{R}^{d \times d}$ and $\boldsymbol{b_1} \in \mathbb{R}^d$ are learnable parameters. If node $v$ is involved in multiple hyperedge events $\{h_i, h_{i+1}, \ldots h_{i+L}\}$, then we will take the mean of dynamic embeddings from all of the hyperedges. Here, $L$ is the number of concurrent hyperedges.

# 3 Learning Procedure

## 3.1 Loss Function

Once intensity parameterization is fixed for temporal point process as in Equation 3 or 4, the likelihood for hyperedge events $\mathcal{E}(T) = \{(h_1, t_1), \ldots, (h_m, t_m)\}$ occurring in an interval $[0, T]$ can be modeled as, $p(\mathcal{E}(T)) = \prod_{i=1}^m p_{h_i}(t_i) \prod_{h \in \mathcal{H}} \mathcal{S}_h(t_h^l, T)$. Here, $p_{h_i}(t_i)$ is the probability of hyperedge event $h_i$ occurring at time $t_i$ as defined in Equation 1, $\mathcal{S}_h(t_h^l, T)$ is the probability that no event occurred for hyperedge $h$ for the interval $[t_h^l, T]$, and $t_h^l$ is the last time occurrence for $h$ ($t_h^l = 0$, when no event of $h$ is observed). The loss for learning parameters can be found by taking the negative of log-likelihood as, $\mathcal{L} = -\sum_{i=1}^m \log(\lambda_{h_i}(t_i)) + \sum_{h \in \mathcal{H}} \int_0^T \lambda_h(t)dt$. Here, the first term corresponds to the sum of the negative log intensity of occurred events. The second term corresponds to the sum of intensities of all events. The following happens when we minimize the loss, the intensity rate of occurred events increases due to minimization of the first term, and intensity rates of events not occurred decreases due to minimization of the second term. However, directly implementing this equation is computationally inefficient as $|\mathcal{H}| \leq 2^{|\mathcal{V}|}$ is very large. Further, the integration in the second term does not always have a closed-form expression. In the next section, we will give this model a computationally efficient mini-batch training procedure.

## 3.2 Mini-Batch Loss

We divide the event sequences into independent segments to make backpropagation through time feasible, as done in previous works on pair-wise dynamic networks [5]. Then the loss for each segment is calculated as follows, for each $(h_i, t_i)$ in the segment $\mathcal{E}_M = \{(h_1, t_1), \ldots, (h_M, t_M)\}$, we use Monte-Carlo integration to find the log survival term of $p_{h_i}(t_i)$. Then the negative log likelihood for that event is,

$$\mathcal{T}^s = \{t_j^s\}_{j=1}^N \leftarrow \mathrm{Uniform}(t_{i-1}, t_i, N)$$
$$\mathcal{L}_{h_i} = -\log(\lambda_{h_i}(t_i)) + \sum_{j=2}^N (t_j^s - t_{j-1}^s)\lambda_{h_i}(t_j^s). \tag{6}$$

Here, $\mathcal{T}^s$ is the set of uniformly sampled time points from the interval $[t_{i-1}, t_i]$. Then to consider the interactions events $h \in \mathcal{H}$ that were not observed during the above period, we sample some negative hyperedges for each interaction event $(h_i, t_i)$ as described below,

1. Choose the size of negative hyperedge $k$ based on a categorical distribution over hyperedge sizes observed in the training data. Here, parameters of the categorical distribution are learned from the training dataset.
2. Sample $\min(\lceil k/2 \rceil, |h_i|)$ nodes from the hyperedge $h_i$ and rest of the nodes from $\mathcal{V} - h_i$. This strategy will avoid the trivial negative samples.

Following the above steps, we sample $\mathcal{H}_i^n = \{h_1^n, \dots, h_{\mathcal{B}}^n\}$ negative hyperedges, and for each of them, we calculate the negative log-likelihood for events not happening using Monte-Carlo integration. Then Equation 6 becomes, $\mathcal{L}_{h_i} = -\log(\lambda_{h_i}(t_i)) + \sum_{h \in \mathcal{H}_i^n \cup \{h_i\}} \sum_{j=2}^{N} (t_j^s - t_{j-1}^s) \lambda_h(t_j^s)$. The final mini-batch loss is calculated by summing all $\mathcal{L}_{h_i}$ for the events $(h_i, t_i)$ in $\mathcal{E}_M$, $\sum_{i=1}^{M} \mathcal{L}_{h_i}$. Then the gradients are backpropagated for this loss, and the above training procedure is repeated in the next segment.

# 4 Extending to Bipartite Hyperedges

A higher-order interaction between nodes of two different type can be represented as a bipartite hyperedge $h = (\{v_1, \dots, v_k\}, \{v_{1'}, \dots, v_{k'}\})$. Here, $\{v_1, \dots, v_k\} \in \mathcal{H}$ is the left hyperedge with nodes from set $\mathcal{V}$ , $\{v_{1'}, \dots, v_{k'}\} \in \mathcal{H}^{'}$ is the right hyperedge with nodes from set $\mathcal{V}^{'}$, and $\mathcal{V} \cap \mathcal{V}^{'} = \emptyset$. More details on these type of higher-order heterogeneous graphs can be found in the recent work [20]. For defining conditional intensity function, similar to homogenous hyperedges, we define $\lambda_h(t) = f(\{v_1, \dots, v_k\}, \{v_{1'}, \dots, v_{k'}\})$. Here $f(.) \geq 0$ is defined by a neural network, and $\boldsymbol{v_i}(t), \boldsymbol{v_{i'}}(t)$ are node embeddings of the nodes in $h$. We follow the same architecture of CATSETMAT [20] for defining $f(.)$ as explained in Appendix B.2.

Table 1: Datasets used for Homogeneous and Bipartite Hyperedges along with their vital statistics.

| **Datasets** | $\lvert\mathcal{V}\rvert$ | $\lvert\mathcal{V}^{'}\rvert$ | $\lvert\mathcal{E}(T)\rvert$ | $\lvert\mathcal{H}\rvert$ | $\lvert\mathcal{H}^{'}\rvert$ |
|---|---|---|---|---|---|
| **email-Enron** | 143 | N/A | 10,883 | 1,542 | N/A |
| **email-Eu** | 998 | N/A | 234,760 | 25,791 | N/A |
| **congress-bills** | 1,718 | N/A | 260,851 | 85,082 | N/A |
| **NDC-classes** | 1,161 | N/A | 49,724 | 1,222 | N/A |
| **NDC-sub** | 5,311 | N/A | 112,405 | 10,025 | N/A |
| **CastGenre** | 5,763 | 20 | 12,295 | 11,665 | 2,065 |
| **CastKeyword** | 12,180 | 3,997 | 3,360 | 3,360 | 3,345 |
| **CastCrew** | 12,634 | 12,704 | 3,517 | 3,507 | 3,517 |

Now for learning good dynamic representation for nodes, we follow the same architecture explained in Section 2.2. But, the left and right hyperedges have their own set of parameters for the `Temporal Drift`, `History Aggregation`, and `Interaction Update` stages. Further, for dynamic embeddings in the `Interaction Update` stage, nodes in the left hyperedge have it as a function of embeddings of nodes in the right hyperedge, and vice versa. This function shares its parameters with the conditional intensity function, as shown in Equation 16 in Appendix B.2.

For learning parameters, we follow the same procedure as that of homogenous hyperedges as explained in Section 3, except the negative sampling is done differently. We keep the left, or right hyperedge fixed for generating negative samples and add a corrupted hyperedge on the other side. For example, for generating left hyperedge negative samples, we replace the right hyperedge by selecting a random subset of nodes from the right node-set $\mathcal{V}^{'}$. The size of the corrupted hyperedge is selected based on the categorical distribution of sizes of the right hyperedge in the training set.

# 5 Experimental Settings

## 5.1 Datasets

All the datasets for homogeneous hyperedge interactions in Table 1 are taken from work [2][1]. The bipartite hyperedges interactions are prepared from Kaggle's The Movies Dataset [2]. A detailed explanation of all the datasets is given in Appendix C. For homogeneous hypergraphs, $|\mathcal{V}^{'}|$ is not applicable (N/A) as there is only one type of node, and $|\mathcal{H}^{'}|$ is N/A as it is not directed.

[1] https://www.cs.cornell.edu/ arb/data/

[2] https://www.kaggle.com/datasets/tmdb/tmdb-movie-metadata

### 5.2 Baselines

In Table 2, we have compared the properties of the baseline models we created against the proposed models HGDHE for the homogenous hyperedge interactions and HGBHDE for the bipartite hyperedge interactions. Here, models *Dynamic Edge* (DE) and *Dynamic Edge-drift* (DE-drift) use pairwise edge models instead of hyperedge interaction. Similarly, model *Bipartite Dynamic Edge* (BDE) uses bipartite pairwise edges to model bipartite hyperedge interaction. Further in models *Rayleigh Hyperedge* (RHE) and *Rayleigh Dynamic Hyperedge* (RDHE), conditional intensity is modeled as Rayleigh process as in Equation 4, and duration predictions are made using the closed-form expression in Equation 9. *Dynamic Hyperedge* (DHE) is the model that uses hyperedges for predicting higher-order interactions and has the same dynamic node presentation as of DE model. So, in our studies, we will compare these two models to claim and establish the advantage of hyperedge modeling over pairwise modeling. Similarly, *Dynamic Hyperedge-drift* (DHE-drift) is the hyperedge model version of the DE-drift model. In the case of bipartite hyperedge interactions, we will compare models *Bipartite Dynamic Hyperedge* (BDHE) and BDE. A more detailed description of baselines can be found in Appendix D.

Table 2: Models and their properties. Here, ✓ indicates the usage of that property, and ✗ indicates the absence of that property.

| **Methods** | **Temporal Drift** | **History Aggregation** | **Interaction Update** | **Hyperedge** | **Bipartite** |
|---|---|---|---|---|---|
| RHE | ✗ | ✗ | ✗ | ✓ | ✗ |
| RDHE | ✓ | ✗ | ✓ | ✓ | ✗ |
| DE-drift | ✓ | ✗ | ✗ | ✗ | ✗ |
| DE | ✓ | ✗ | ✓ | ✗ | ✗ |
| DHE-drift | ✓ | ✗ | ✗ | ✓ | ✗ |
| DHE | ✓ | ✗ | ✓ | ✓ | ✗ |
| HGDHE-hist | ✓ | ✓ | ✗ | ✓ | ✗ |
| HGDHE | ✓ | ✓ | ✓ | ✓ | ✗ |
| BDE | ✓ | ✗ | ✓ | ✗ | ✓ |
| BDHE | ✓ | ✗ | ✓ | ✓ | ✓ |
| HGBDHE | ✓ | ✓ | ✓ | ✓ | ✓ |

### 5.3 Prediction Tasks

Using temporal point process models, we can predict both the next event type and the time of the event. The following are the equivalent tasks in our settings.

**Interaction Type Prediction.** The type of interaction that occurs at time $t$ can be predicted by finding the $h_i$ with the maximum intensity value at that time, as shown below,

$$\hat{h} = \arg\max_{h_i} \lambda_{h_i}(t). \tag{7}$$

**Interaction Duration Prediction.** For interaction $h$ occurred at time $t_h^p$, to predict the duration for future interaction, we have to calculate the expected time $t$ with respect the conditional distribution $p_h(t)$ in Equation 1,

$$\hat{t} = \int_{t_h^p}^{\infty} (t - t_h^p) p_h(t). \tag{8}$$

If $\lambda(t)$ is modeled using a Rayleigh process as in Equation 4, we calculate the $\hat{t}$ in close form as,

$$\hat{t} = \sqrt{\frac{\pi}{2 \exp f(\boldsymbol{v_1}(t_h^p), \boldsymbol{v_2}(t_h^p), \ldots, \boldsymbol{v_k}(t_h^p))}}. \tag{9}$$

Otherwise, we have to compute the integration by sampling.

### 5.4 Metrics of Evaluation

**Mean Average Reciprocal Rank (MRR) .** We use this for evaluating the performance of interaction prediction at time $t$. For finding this, we find the reciprocal of rank ($r_i$) of the true hyperedge against

Table 3: Performance of dynamic homogeneous hyperedge forecasting in tasks of interaction type and interaction duration prediction. Here, interaction type prediction is evaluated using MRR in $\%$, and interaction duration prediction is evaluated using MAE. The proposed model HGDHE beats baseline models in almost in all the settings.

| Methods | email-Enron | | email-Eu | | congress-bills | |
|---|---|---|---|---|---|---|
| | **MRR** | **MAE** | **MRR** | **MAE** | **MRR** | **MAE** |
| RHE | $34.45 \pm 0.81$ | $127.19 \pm 12.17$ | $52.37 \pm 0.47$ | $26.43 \pm 0.47$ | $32.14 \pm 0.38$ | $54.13 \pm 9.60$ |
| RDHE | $26.73 \pm 2.76$ | $34.22 \pm 0.49$ | $27.68 \pm 5.02$ | $17.54 \pm 0.68$ | $52.90 \pm 3.24$ | $2.44 \pm 0.22$ |
| DE-drift | $30.84 \pm 0.29$ | $48.50 \pm 0.65$ | $43.47 \pm 1.69$ | $21.21 \pm 0.06$ | $43.27 \pm 0.23$ | $4.07 \pm 0.32$ |
| DE | $52.89 \pm 0.38$ | $16.52 \pm 3.14$ | $44.27 \pm 0.74$ | $19.37 \pm 1.27$ | $56.27 \pm 2.89$ | $2.65 \pm 0.42$ |
| DHE-drift | $50.25 \pm 1.65$ | $88.14 \pm 4.59$ | $58.38 \pm 0.20$ | $33.56 \pm 0.82$ | $84.50 \pm 0.13$ | $3.84 \pm 0.26$ |
| DHE | $60.57 \pm 1.97$ | $25.48 \pm 4.37$ | $64.03 \pm 2.22$ | $19.72 \pm 2.00$ | $\mathbf{92.21 \pm 0.19}$ | $1.87 \pm 0.22$ |
| HGDHE-hist | $\mathbf{65.26 \pm 1.24}$ | $18.25 \pm 0.43$ | $60.69 \pm 0.09$ | $24.66 \pm 0.77$ | $85.31 \pm 0.10$ | $3.44 \pm 0.34$ |
| HGDHE | $62.21 \pm 2.85$ | $\mathbf{16.12 \pm 1.45}$ | $\mathbf{66.12 \pm 2.90}$ | $\mathbf{15.18 \pm 2.14}$ | $92.09 \pm 0.03$ | $\mathbf{1.65 \pm 0.06}$ |

Table 4: Performance of dynamic homogeneous hyperedge forecasting in tasks of interaction type and interaction duration prediction. The proposed model HGDHE beats baseline models in almost in all the settings.

| Methods | NDC-classes | | NDC-sub | |
|---|---|---|---|---|
| | **MRR** | **MAE** | **MRR** | **MAE** |
| RHE | $87.64 \pm 1.63$ | $8.98 \pm 1.09$ | $74.33 \pm 0.30$ | $4.66 \pm 0.15$ |
| RDHE | $81.17 \pm 2.56$ | $6.29 \pm 0.91$ | $66.46 \pm 1.50$ | $2.52 \pm 0.07$ |
| DE-drift | $60.64 \pm 0.19$ | $5.02 \pm 0.15$ | $64.90 \pm 0.29$ | $12.38 \pm 0.84$ |
| DE | $64.58 \pm 0.78$ | $3.60 \pm 0.38$ | $65.83 \pm 1.41$ | $13.82 \pm 1.93$ |
| DHE-drift | $88.68 \pm 0.71$ | $1.92 \pm 0.12$ | $79.31 \pm 0.41$ | $3.03 \pm 0.01$ |
| DHE | $88.93 \pm 0.16$ | $1.84 \pm 0.21$ | $86.52 \pm 0.18$ | $3.49 \pm 0.14$ |
| HGDHE-hist | $\mathbf{91.24 \pm 0.74}$ | $1.42 \pm 0.08$ | $80.73 \pm 0.15$ | $1.75 \pm 0.19$ |
| HGDHE | $91.01 \pm 0.35$ | $\mathbf{1.21 \pm 0.04}$ | $\mathbf{86.92 \pm 0.51}$ | $\mathbf{1.65 \pm 0.18}$ |

candidate negative hyperedge in descending order of $\lambda_h(t)$ and then average them for all samples in the test set, $MRR = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{r_i+1}$. Here, better-performing models have higher MRR values.

**Mean Average Error (MAE) .** We use this for evaluating the performance of interaction duration prediction, $MAE = \frac{1}{N}\sum_{i=1}^{N}|\hat{t}_i - t_i^{true}|$. Here, better performing models have lower MAE values.

### 5.5 Parameter Settings

For all experiments, we use the learning rate of $0.001$, the embedding size $d$ is fixed at $64$ for homogeneous hyperedges and bipartite hyperedges, the batch size $M$ is fixed as $128$, and the negative sampling is fixed as $\mathcal{B} = 20$, and the training is done for $100$ epochs. For the Monte Carlo estimate of $\log$ of survival probability in Section 3.2, we use $N = 20$ for datasets email-Enron, email-EU, and NDC-classes, $N = 5$ for NDC-sub and congress-bills datasets. The choice of $N$ is made by considering memory constraints. All models are implemented PyTorch [19], and all training is done using its Adam [14] optimizer. For all datasets, we use the first $50\%$ of interactions for training, the next $25\%$ for validation, and the rest for testing. The details of the computational infrastructure used is provided in the appendix E. All the reported scores are the average of ten randomized runs along with their standard deviation.

## 6 Results

In Tables 3 and 4, one can see that the proposed model *Hypergraph Dynamic Hyperedge* (HGDHE) performs better than baselines in almost all the settings. Further, it significantly outperforms RHE,

Table 5: Performance of dynamic bipartite hyperedge forecasting in tasks of interaction type and interaction duration prediction. Proposed model HGBDHE beats baseline models in almost in all the settings.

| Methods | CastGenre | | CastKeyword | | CastCrew | |
|---|---|---|---|---|---|---|
| | **MRR** | **MAE** | **MRR** | **MAE** | **MRR** | **MAE** |
| BDE | $23.55 \pm 0.75$ | $10.34 \pm 0.34$ | $13.61 \pm 0.14$ | $21.98 \pm 0.60$ | $13.61 \pm 0.24$ | $22.23 \pm 1.03$ |
| DHE | $27.58 \pm 0.88$ | $\mathbf{2.88 \pm 0.43}$ | $36.18 \pm 1.34$ | $15.32 \pm 1.75$ | $26.03 \pm 1.75$ | $9.29 \pm 2.01$ |
| BDHE | $33.22 \pm 0.52$ | $2.91 \pm 0.26$ | $38.77 \pm 1.69$ | $9.61 \pm 2.33$ | $37.29 \pm 2.65$ | $9.18 \pm 1.37$ |
| HGDHE | $27.59 \pm 1.60$ | $19.39 \pm 1.62$ | $35.32 \pm 1.96$ | $22.64 \pm 1.50$ | $25.19 \pm 2.83$ | $8.85 \pm 2.07$ |
| HGBDHE | $\mathbf{33.65 \pm 1.58}$ | $4.16 \pm 0.84$ | $\mathbf{41.32 \pm 1.74}$ | $\mathbf{9.27 \pm 1.67}$ | $\mathbf{42.77 \pm 2.00}$ | $\mathbf{8.77 \pm 1.68}$ |

which uses static, and RDHE, which uses piece-wise constant node embeddings. Even though these models have a closed form expression for the event probability and duration estimation $\hat{t}$, their performance is poor compared to HGDHE, which has conditional intensity as a function of its dynamic node representation. We can also see that by comparing baselines, DHE to RDHE and RHE to DHE-drift, the models DHE and DHE-drift perform better as they use dynamic node representations as input to the conditional intensity function. This is because those models have more expressiveness and do not assume a parametric form for $\lambda_h(t)$.

Further, we can observe the advantage of using hyperedge for modeling higher-order interactions when comparing models DHE to DE and DHE-drift to DE-drift. This comparison is important because those models use the same dynamic node presentation, but DHE uses hyperedge modeling, and DE uses pairwise edge modeling. The same applies to the comparison of DHE-drift to DE-drift. Between DHE and DE, there is an improvement in the MRR metric in interaction type prediction tasks for all datasets. There is a $38.4\%$ percentage gain in MRR for DHE compared to DE and a $40.3\%$ gain for DHE-drift compared to DE-drift. In the interaction duration prediction task, we can observe a significant reduction in MAE for all the datasets except for the email-Enron and email-Eu datasets. A similar observation can be made between DHE-drift and DE-drift models. This is because more than $70\%$ of interactions are pairwise in both of those datasets. Even though pairwise edges can achieve a reasonable performance, we cannot identify the hyperedge among them if there are concurrent hyperedges with common nodes, as explained in Figure 1. Hence, hyperedge models perform better than pairwise models for interaction type prediction.

**Performance on Bipartite Interactions.** In Table 5, one can observe that the proposed model *Hypergraph Bipartite Dynamic Hyperedge* (HGBDHE) performs better than all the other baselines in almost all settings. We can see that the models that use the bipartite property of the interaction perform better than their homogeneous counterparts. This can be inferred by the better performance of BDHE compared to DHE and HGBDHE compared to HGDHE. There is a gain of $23.6\%$ in MRR and a reduction of $12.4\%$ MAE for BDHE compared to DHE. Similarly, there is a gain of $36.2\%$ in MRR and a reduction of $46.1\%$ MAE for HGBDHE compared to HGDHE. This is because these models are more expressive and consider the bipartite nature of the interaction, as explained in Section 4. Similar to the case of homogeneous interactions, the pairwise model performs considerably poorer than the bipartite hyperedge models. It can be inferred from the poor performance of BDE model, which uses bipartite pairwise edge, compared to BDHE, which uses bipartite hyperedge for interaction modeling. Hence, one can conclude that the bipartite hyperedge modeling represents data more accurately than pairwise modeling.

### 6.1 Ablation Studies

**Effect of Interaction Update on Performance.** For this, we will compare models that have this stage in their update equation to models that do not. Firstly, we can observe a reduction of MAE by $24.7\%$ and a gain of $3.9\%$ in MRR for HGDHE when compared to HGDHE-hist. Similarly, we can see DHE outperforms DHE-drift considerably in the interaction type prediction task for all datasets. We can also observe a significant reduction in MAE for email-Enron and email-Eu datasets. Between RHE and RDHE, one can see that RDHE has a performance gain in interaction duration prediction in all datasets. Even though RHE performs better in MRR in most datasets, one can see MRR for

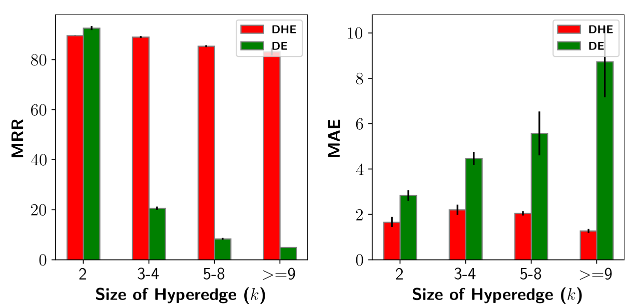
RDHE is much better in congress-bills datasets. Hence one can conclude that using `Interaction Update` stage resulted in performance improvement.

**Effect of Temporal Drift on Performance.** Similar to the earlier study, the advantage of this stage can be observed by comparing the performance of RHE to DHE-drift and RDHE to DHE. There is an average of $44\%$ gain in MRR and $42.03\%$ reduction in MAE for DHE-drift compared to RHE. Further, DHE uniformly outperforms RDHE for interaction type prediction in all datasets. There is an average of $90\%$ improvement in the MRR for the interaction type prediction task and a $13.7\%$ decrease in the MAE error for the duration prediction task. Hence, `Temporal Drift` stage helps in performance improvement.

**Effect of History Aggregation.** The advantage of this stage can be observed by comparing HGDHE to DHE and HGDHE-hist to DHE-drift. We can see HGDHE outperforms DHE in all the tasks except in Congress-bills. There is an average gain of $1.7\%$ in MRR and a $31.75\%$ reduction in MAE for HGDHE compared to DHE. We can see HGDHE-hist outperforms DHE-drift in all settings. A similar observation can be made in bipartite datasets when comparing models HGBDHE and BDHE. Both models give comparable performance for interaction duration prediction except for CastGenre dataset. For interaction type prediction HGBDHE achieves a gain of $7\%$ in MRR when compared to BDHE.

### 6.2 Visualizations

**Effect of hyperedge size ($k$) on performance.** Here, we compare models DHE and DE as the former uses a hyperedge based conditional intensity function, and the latter uses a pairwise edge based one, but both models use the same dynamic representation. To compare them, we divide hyperedges into different groups based on their sizes, and the mean value of the evaluation metric is calculated for each group. These groups are defined so that each group has enough samples to make the comparisons statistically significant. In Figures 2a and 2b, we have shown the effect of hyperedge size on our model DHE, which uses hyperedge modeling, and compared it against model DE, which uses pairwise edges. Here, each hyperedge is grouped into groups with $k = 2, 3 \leq k \leq 4, 5 \leq k \leq 8$ and $k \geq 9$. From Figure 2a we can see that performance of DE is poor for hyperedges that have a size of more than 2, but our model DHE has almost similar performance for hyperedges of different sizes. The reason for this is that DHE is suited more for hyperedges of varying sizes compared to DE. We observed a similar trend in other datasets, as shown in Appendix F. For the interaction duration prediction task, error for the DE model increases with hyperedge size while the DHE model performs similarly for all hyperedge sizes.




(a) (b)

Figure 2: Figure 2a shows the effect of hyperedge size ($k$) on interaction type prediction, and Figure 2b shows the effect on interaction duration prediction for NDC-classes dataset

## 7 Conclusion and Future Work

Forecasting higher-order interactions in time-evolving hypergraphs are very challenging and this has not been studied before. In this work, we fill this gap by proposing a model for forecasting higher-order interaction between nodes in a network as temporal hyperedge formation events. For this, we develop a mechanism that uses the temporal point process to learn the dynamic representation of nodes. Our future work includes extending the proposed model for modeling multi-relational higher orders interactions [7] and incorporating multi-hop information into node representation using hypergraph neural network-based techniques for better predictive performance. We also like to reduce the training time by using t-batch [16] based approach that can use parallel training techniques.

## References


[1] Song Bai, Feihu Zhang, and Philip HS Torr. Hypergraph convolution and hypergraph attention. *Pattern Recognition*, 110:107637, 2021.

[2] Austin R. Benson, Rediet Abebe, Michael T. Schaub, Ali Jadbabaie, and Jon Kleinberg. Simplicial closure and higher-order link prediction. *Proceedings of the National Academy of Sciences*, 2018.

[3] Jiangxia Cao, Xixun Lin, Xin Cong, Shu Guo, Hengzhu Tang, Tingwen Liu, and Bin Wang. Deep structural point process for learning temporal interaction networks. In Nuria Oliver, Fernando Pérez-Cruz, Stefan Kramer, Jesse Read, and Jose A. Lozano, editors, *Machine Learning and Knowledge Discovery in Databases. Research Track*, pages 305–320, 2021.

[4] da Xu, chuanwei ruan, evren korpeoglu, sushant kumar, and kannan achan. Inductive representation learning on temporal graphs. In *International Conference on Learning Representations (ICLR)*, 2020.

[5] Hanjun Dai, Yichen Wang, Rakshit Trivedi, and Le Song. Deep coevolutionary network: Embedding user and item features for recommendation. *arXiv*, 2016.

[6] D. J. Daley and D. Vere-Jones. *An introduction to the theory of point processes. Vol. I.* Springer-Verlag, 2003.

[7] Bahare Fatemi, Perouz Taslakian, David Vazquez, and David Poole. Knowledge hypergraphs: Prediction beyond binary relations. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, pages 2191–2197, 7 2020.

[8] James H Fowler. Legislative cosponsorship networks in the us house and senate. *Social Networks*, 28(4):454–465, 2006.

[9] Jayanta K. Ghosh. Survival and event history analysis: A process point of view by odd o. aalen, Ørnulf borgan, håkon k. gjessing. *International Statistical Review*, 77(3):463–464, 2009.

[10] Tony Gracious, Shubham Gupta, Arun Kanthali, Rui M Castro, and Ambedkar Dukkipati. Neural latent space model for dynamic networks and temporal knowledge graphs. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 4054–4062, 2021.

[11] Shubham Gupta, Gaurav Sharma, and Ambedkar Dukkipati. A generative model for dynamic networks with applications. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 7842–7849, 2019.

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

[13] Jie Huang, Xin Liu, and Yangqiu Song. Hyper-path-based representation learning for hypernetworks. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, CIKM '19, page 449–458, 2019.

[14] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR (Poster)*, 2015.

[15] Bryan Klimt and Yiming Yang. The enron corpus: A new dataset for email classification research. In *European Conference on Machine Learning*, pages 217–226, 2004.

[16] Srijan Kumar, Xikun Zhang, and Jure Leskovec. Predicting dynamic embedding trajectory in temporal interaction networks. In *Proceedings of the 25th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2019.

[17] Giang Hoang Nguyen, John Boaz Lee, Ryan A Rossi, Nesreen K Ahmed, Eunyee Koh, and Sungchul Kim. Continuous-time dynamic network embeddings. In *Companion Proceedings of the The Web Conference 2018*, pages 969–976, 2018.

[18] Ashwin Paranjape, Austin R Benson, and Jure Leskovec. Motifs in temporal networks. In *Proceedings of the tenth ACM international conference on web search and data mining*, pages 601–610, 2017.

[19] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

[20] Govind Sharma, Swyam Prakash Singh, V Susheela Devi, and M Narasimha Murty. The cat set on the mat: Cross attention for set matching in bipartite hypergraphs. *arXiv*, 2021.

[21] Oleksandr Shchur, Ali Caner Türkmen, Tim Januschowski, and Stephan Günnemann. Neural temporal point processes: A review. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, pages 4585–4593. International Joint Conferences on Artificial Intelligence Organization, 8 2021.

[22] Rakshit Trivedi, Mehrdad Farajtabar, Prasenjeet Biswal, and Hongyuan Zha. Dyrep: Learning representations over dynamic graphs. In *International Conference on Learning Representations*, 2019.

[23] Muhan Zhang, Zhicheng Cui, Shali Jiang, and Yixin Chen. Beyond link prediction: Predicting hyperlinks in adjacency space. In *AAAI*, 2018.

[24] Ruochi Zhang, Yuesong Zou, and Jian Ma. Hyper-sagnn: a self-attention based graph neural network for hypergraphs. In *International Conference on Learning Representations*, 2019.

# Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes] We have developed a framework to model higher-order interactions as hyperedge forecasting in dynamic networks using temporal point process. Please refer to Section 2.
   (b) Did you describe the limitations of your work? [Yes] See Section 7 where we mention the need to use multi-hop information in future works.We also note the need to decrease the training time by parallelizing the implementation.
   (c) Did you discuss any potential negative societal impacts of your work? [N/A]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes] We do not use any human derived data. All the datasets we used are available as open source
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] All codes and datasets we used are provided along with the supplementary material
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] Please refer to Section 5.5
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] Please refer to Tables 3, 4, and 5, and Figures 6b, 6a
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] Please refer to Appendix E

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] We have added the appropriate references. No code from outside sources was used
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes] We have preprocessed three new datasets for bipartite hyperedge forecasting and they are included in the supplementary materials
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]