CAt-Walk: Inductive Hypergraph Learning via Set Walks

Anonymous Author(s)
Affiliation
Address
email

Abstract

Temporal hypergraphs provide a powerful paradigm for modeling time-dependent, higher-order interactions in complex systems. Representation learning for hypergraphs is essential for extracting patterns of the higher-order interactions that are critically important in real-world problems in social network analysis, neuroscience, finance, etc. However, existing methods are typically designed only for specific tasks or static hypergraphs. We present CAt-Walk, an inductive method that learns the underlying dynamic laws that govern the temporal and structural processes underlying a temporal hypergraph. CAt-Walk introduces a temporal, higher-order walk on hypergraphs, SetWalk, that extracts higher-order causal patterns. CAt-Walk uses a novel adaptive and permutation invariant pooling strategy, SetMixer, along with a set-based anonymization process that hides the identity of hyperedges. Finally, we present a simple yet effective neural network model to encode hyperedges. Our evaluation on 10 hypergraph benchmark datasets shows that CAt-Walk attains outstanding performance on temporal hyperedge prediction benchmarks in both inductive and transductive settings. It also shows competitive performance with state-of-the-art methods for node classification.

1 Introduction

Temporal networks have become increasingly popular for modeling interactions among entities in dynamic systems [1–4]. While most existing work focuses on pairwise interactions between entities, many real-world complex systems exhibit natural relationships among multiple entities [5–7]. Hypergraphs provide a natural extension to graphs by allowing an edge to connect any number of vertices, making them capable of representing higher-order structures in data. Representation learning on (temporal) hypergraphs has been recognized as an important machine learning problem and has become the cornerstone behind a wealth of high-impact applications in computer vision [8, 9], biology [10, 11], social networks [12, 13], and neuroscience [14, 15].

Many recent attempts to design representation learning methods for hypergraphs are equivalent to applying Graph Neural Networks (Gnns) to the clique-expansion (CE) of a hypergraph [16–20]. CE is a straightforward way to generalize graph algorithms to hypergraphs by replacing hyperedges with (weighted) cliques [17–19]. However, this decomposition of hyperedges limits expressiveness, leading to suboptimal performance [5, 21–23] (see Theorem 1 and Theorem 4). New methods that encode hypergraphs directly partially address this issue [10, 24–27]. However, these methods suffer from some combination of the following three limitations: they are designed for ① learning the structural properties of *static hypergraphs* and do not consider temporal properties, ② the transductive setting, limiting their performance on unseen patterns and data, and ③ a specific downstream task (e.g., node classification [24], hyperedge prediction [25], or subgraph classification [26]) and cannot easily be extended to other downstream tasks, limiting their application.

Submitted to 37th Conference on Neural Information Processing Systems (NeurIPS 2023). Do not distribute.

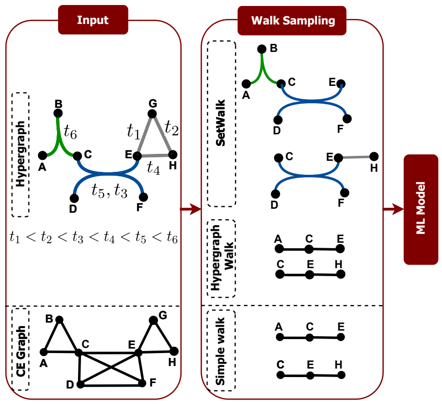

Figure 1: The advantage of SetWalks in walk-based hypergraph learning.

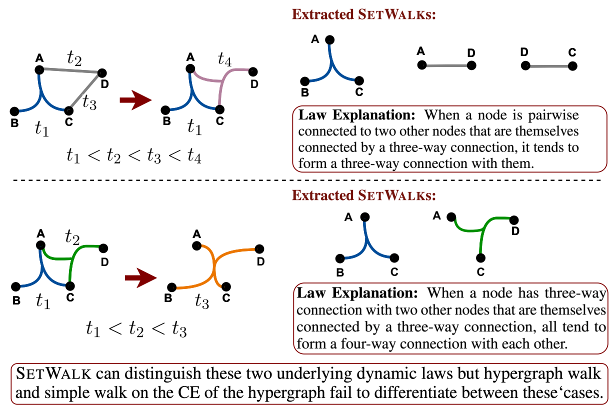

Figure 2: The advantage of SetWalks in causality extraction and capturing complex dynamic laws.

Temporal motif-aware and neighborhood-aware methods have been developed to capture complex patterns in data [28–30]. However, counting temporal motifs in large networks is time-consuming and non-parallelizable, limiting the scalability of these methods. To this end, several recent studies suggest using temporal random walks to automatically retrieve such motifs [31–35]. One possible solution to capturing underlying temporal and higher-order structure is to extend the concept of a hypergraph random walk [36–42] to its temporal counterpart by letting the walker walk over time. However, existing definitions of random walks on hypergraphs offer limited expressivity and sometimes degenerate to simple walks on the CE of the hypergraph [39] (see Appendix C). There are two reasons for this: ① Random walks are composed of a sequence of *pair-wise* interconnected vertices, even though edges in a hypergraph connect *sets* of vertices. Decomposing them into sequences of simple pair-wise interactions loses the semantic meaning of the hyperedges (see Theorem 4). ② A sampling probability of a walk on a hypergraph must be different from its sampling probability on the CE of the hypergraph [36–42]. However, Chitra and Raphael [39] shows that each definition of the random walk with edge-independent sampling probability of nodes is equivalent to random walks on a weighted CE of the hypergraph. Existing studies on random walks on hypergraphs ignore ① and focus on ② to distinguish the walks on simple graphs and hypergraphs. However, as we show in Table 2, ① is equally important, if not more so.

For example, Figure 1 shows the procedure of existing walk-based machine learning methods. The neural networks in the model take as input only sampled walks. However, the output of the hypergraph walk [36, 37] and simple walk on the CE graph are the same. This means that the neural network cannot distinguish between pair-wise and higher-order interactions.

We present **C**ausal **A**nonymous Se**t** **Walk**s (CAt-Walk), an inductive hyperedge learning method. We introduce a hyperedge-centric random walk on hypergraphs, called SetWalk, that automatically extracts temporal, higher-order motifs. The hyperedge-centric approach enables SetWalks to distinguish multi-way connections from their corresponding CEs (see Figure 1, Figure 2, and Theorem 1). We use temporal hypergraph motifs that reflect network dynamics (Figure 2) to enable CAt-Walk to work well in the inductive setting. To make the model agnostic to the hyperedge identities of these motifs, we use two-step, set-based anonymization: ① Hide node identities by assigning them new positional encodings based on the number of times that they appear at a specific position in a set of sampled SetWalks, and ② Hide hyperedge identities by combining the positional encodings of the nodes comprising the hyperedge using a novel permutation invariant pooling strategy, called SetMixer. We incorporate a neural encoding method that samples a few SetWalks starting from nodes of interest. It encodes and aggregates them via MLP-Mixer [43] and our new pooling strategy SetMixer, respectively, to predict temporal, higher-order interactions. Finally, we discuss how to extend CAt-Walk for node classification. Figure 3 shows the schematic of the CAt-Walk.

We theoretically and experimentally discuss the effectiveness of CAt-Walk and each of its components. More specifically, we prove that SetWalks are more expressive than existing random walk algorithms on hypergraphs. We demonstrate SetMixer's efficacy as a permutation invariant pooling strategy for hypergraphs and prove that using it in our anonymization process makes that process more expressive than existing anonymization processes [44, 32] when applied to the CE of the hypergraphs. To

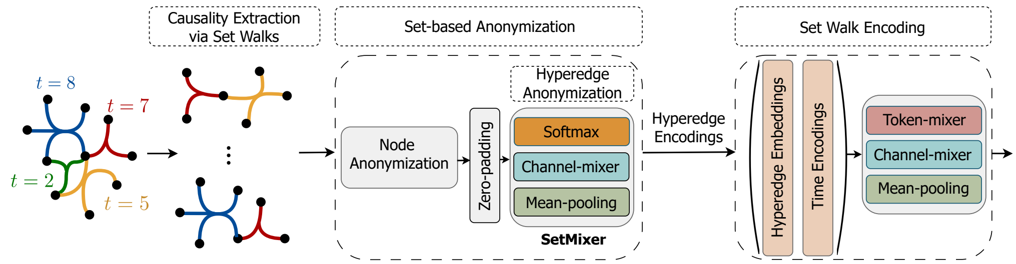

Figure 3: **Schematic of the CAt-Walk.** CAt-Walk consists of three stages: (1) Causality Extraction via Set Walks (§3.2), (2) Set-based Anonymization (§3.3), and (3) Set Walk Encoding (§3.4).

the best of our knowledge, we report the most extensive experiments in the hypergraph learning literature pertaining to unsupervised hyperedge prediction with 10 datasets and eight baselines. Results show that our method produces 9% and 17% average improvement in transductive and inductive settings, outperforming all state-of-the-art baselines in the hyperedge prediction task. Also, CAt-Walk achieves the best or on-par performance on dynamic node classification tasks. All proofs appear in the Appendix.

2 Related Work

Temporal graph learning is an active research area in recent years [3, 4]. A major group of methods uses Gnns to learn node encodings and Recurrent Neural Networks (Rnns) to update these encodings over time [45–52]. More sophisticated methods based on anonymous temporal random walks [32, 33], line graphs [53], GraphMixer [54], neighborhood representation [55], and subgraph sketching [56] are recently designed to capture complex structures in vertex neighborhoods. Although these methods show promising results in a variety of tasks, they are fundamentally limited in that they are designed for *pair-wise* interaction among vertices and not the higher-order interactions in hypergraphs.

Representation learning on hypergraphs addresses this problem [16, 57]. We group work in this area into three overlapping categories:

① **Clique and Star Expansion**: CE-based methods replace hyperedges with (weighted) cliques and apply Gnns, sometimes with sophisticated propagation rules [20, 24], degree normalization, and nonlinear hyperedge weights [16–20, 38, 58, 59]. Although these methods are simple, it is well-known that CE causes undesired losses in learning performance, specifically when relationships within an incomplete subset of a hyperedge do not exist [5, 21–23]. Star expansion (SE) methods first use hypergraph star expansion and model the hypergraph as a bipartite graph, where one set of vertices represents nodes and the other represents hyperedges [24, 27, 60, 61]. Next, they apply modified heterogeneous GNNs, with possibly dual attention mechanisms from nodes to hyperedges and vice versa [24, 26]. Although this group does not cause as large a distortion as CE, they are neither memory nor computationally efficient. ② **Message Passing**: Most existing hypergraph learning methods, use message passing over hypergraphs [16–20, 24–26, 38, 58, 59, 62]. Recently, Chien et al. [24] and Huang and Yang [27] designed universal message-passing frameworks that include propagation rules of most previous methods (e.g., [16, 18]). The main drawback of these two frameworks is that they are limited to node classification tasks and do not easily generalize to other tasks. ③ **Walk-based**: random walks are a common approach to extracting graph information for machine learning algorithms [31–33, 63, 64]. Several walk-based hypergraph learning methods are designed for a wide array of applications [42, 65–74]. However, most existing methods use simple random walks on the CE of the hypergraph (e.g., [25, 42, 74]). More complicated random walks on hypergraphs address this limitation [39–41, 75]. Although some of these studies show that their walk's transition matrix differs from the simple walk on the CE [39, 75], their extracted walks can still be the same, limiting their expressivity (see Figure 1, Figure 2, and Theorem 1).

Our method differs from all prior (temporal) (hyper)graph learning methods in five ways: CAt-Walk: (I) Captures temporal higher-order properties in a streaming manner: Contrary to existing methods in hyperedge prediction, our method captures temporal property in a streaming manner, avoiding the drawbacks of snapshot-based methods. (II) Works in the inductive setting by extracting underlying dynamic laws of the hypergraph, making it generalizable to unseen patterns and nodes. (III) Introduces a hyperedge-centric, temporal, higher-order walk with a new perspective on the walk sampling procedure. IV Presents a new two-step anonymization process: Anonymization of higher-order patterns (i.e., SetWalks), requires hiding the identity of both nodes and hyperedges. We present a new permutation-invariant pooling strategy to hide hyperedges' identity according to their vertices, making the process provably more expressive than the existing anonymization processes [32, 33, 74]. V Removes self-attention and Rnns from the walk encoding procedure, avoiding their limitations.

Appendices A and B provide a more comprehensive discussion of related work.

3 Method: CAt-Walk Network

3.1 Preliminaries

Definition 1 (Temporal Hypergraphs)**.** *A temporal hypergraph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{X})$, can be represented as a sequence of hyperedges that arrive over time, i.e., $\mathcal{E} = \{(e_1, t_1), (e_2, t_2), \dots\}$, where $\mathcal{V}$ is the set of nodes, $e_i \in 2^{\mathcal{V}}$ are hyperedges, t_i is the timestamp showing when e_i arrives, and $\mathcal{X} \in \mathbb{R}^{|\mathcal{V}|\times f}$ is a matrix that encodes node attribute information for nodes in $\mathcal{V}$. Note that we treat each hyperedge e_i as the set of all vertices connected by e_i.*

Given a hypergraph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{X})$, we represent the set of hyperedges attached to a node u before time t as $\mathcal{E}^t(u) = \{(e, t')|t' < t, u \in e\}$. We say two hyperedges e and e' are adjacent if $e \cap e' \neq \emptyset$ and use $\mathcal{E}^t(e) = \{(e', t')|t' < t, e' \cap e \neq \emptyset\}$ to represent the set of hyperedges adjacent to e before time t. Next, we focus on the hyperedge prediction task: Given a subset of vertices $\mathcal{U} = \{u_1, \dots, u_k\}$, we want to predict whether a hyperedge among all the u_is will appear in the next timestamp or not. In AppendixG.2 we discuss how this approach can be extended to node classification.

3.2 Causality Extraction via SetWalk

The collaboration network in Figure 1 shows how prior work that models hypergraph walks as sequences of vertices fails to capture complex connections in hypergraphs. Consider the two walks: $A \to C \to E$ and $H \to E \to C$. These two walks can be obtained either from a hypergraph random walk or from a simple random walk on the CE graph. Due to the symmetry of these walks with respect to (A, C, E) and (H, E, C), they cannot distinguish A and H, although the neighborhoods of these two nodes follow different patterns: A, B, and C have published a paper together (connected by a hyperedge), but each pair of E, G, and H has published a paper (connected by a pairwise link). We address this limitation by defining a temporal walk on hypergraphs as a sequence of hyperedges:

Definition 2 (SetWalk)**.** *Given a temporal hypergraph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{X})$, a SetWalk with length ℓ on temporal hypergraph $\mathcal{G}$ is a randomly generated sequence of hyperedges (sets):*

$$\mathrm{Sw}: \ (e_1, t_{e_1}) \to (e_2, t_{e_2}) \to \cdots \to (e_\ell, t_{e_\ell}),$$

where $e_i \in \mathcal{E}$, $t_{e_{i+1}} < t_{e_i}$, and the intersection of e_i and e_{i+1} is not empty, $e_i \cap e_{i+1} \neq \emptyset$. In other words, for each $1 \leq i \leq \ell - 1$: $e_{i+1} \in \mathcal{E}^{t_i}(e_i)$. We use $\mathrm{Sw}[i]$ to denote the i-th hyperedge-time pair in the SetWalk*. That is,* $\mathrm{Sw}[i][0] = e_i$ *and* $\mathrm{Sw}[i][1] = t_{e_i}$*.*

Theorem 1. *A random* SetWalk *is equivalent to neither the hypergraph random walk, the random walk on the CE graph, nor the random walk on the SE graph. Also, for a finite number of samples of each,* SetWalk *is more expressive than existing walks.*

In Figure 1, SetWalks capture higher-order interactions and distinguish the two nodes A and H, which are indistinguishable via hypergraph random walks and graph random walks in the CE graph. We present a more detailed discussion and comparison with previous definitions of random walks on hypergraphs in Appendix C.

Causality Extraction. We introduce a sampling method to allow SetWalks to extract temporal higher-order motifs that capture causal relationships by backtracking over time and sample adjacent hyperedges. As discussed in previous studies [32, 33], more recent connections are usually more informative than older connections. Inspired by Wang et al. [32], we use a hyperparameter $\alpha \geq 0$ to sample a hyperedge e with probability proportional to $\exp\left(\alpha(t - t_p)\right)$, where t and t_p are the timestamps of e and the previously sampled hyperedge in the SetWalk, respectively. Additionally, we want to bias sampling towards pairs of adjacent hyperedges that have a greater number of common nodes to capture higher-order motifs. However, as discussed in previous studies, the importance of each node for each hyperedge can be different [24, 26, 39, 62]. Accordingly, the transferring probability from hyperedge e_i to its adjacent hyperedge e_j depends on the importance of the nodes that they share. We address this via a temporal SetWalk sampling process with *hyperedge-dependent node weights*. Given a temporal hypergraph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{X})$, a hyperedge-dependent node-weight function $\Gamma : \mathcal{V} \times \mathcal{E} \to \mathbb{R}^{\geq 0}$, and a previously sampled hyperedge (e_p, t_p), we sample a hyperedge (e, t) with probability:

$$\mathbb{P}[(e,t)|(e_p,t_p)] \propto \frac{\exp\left(\alpha(t-t_p)\right)}{\sum_{(e',t')\in\mathcal{E}^{t_p}(e_p)} \exp\left(\alpha(t'-t_p)\right)} \times \frac{\exp\left(\varphi(e,e_p)\right)}{\sum_{(e',t')\in\mathcal{E}^{t_p}(e_p)} \exp\left(\varphi(e',e_p)\right)}, \tag{1}$$

where $\varphi(e, e') = \sum_{u \in e \cap e'} \Gamma(u, e)\Gamma(u, e')$, representing the assigned weight to $e \cap e'$. We refer to the first and second terms as *temporal bias* and *structural bias*, respectively.

The pseudocode of our SetWalk sampling algorithm and its complexity analysis are in Appendix D. We also discuss this hyperedge-dependent sampling procedure and how it is provably more expressive than existing hypergraph random walks in Appendix C.

Given a (potential) hyperedge $e_0 = \{u_1, u_2, \dots, u_k\}$ and a time t_0, we say a SetWalk, Sw, starts from u_i if $u_i \in \text{Sw}[1][0]$. We use the above procedure to generate M SetWalks with length m starting from each $u_i \in e_0$. We use $\mathcal{S}(u_i)$ to store SetWalks that starts from u_i.

3.3 Set-based Anonymization of Hyperedges

In the anonymization process, we replace hyperedge identities with position encodings, capturing structural information while maintaining the inductiveness of the method. Micali and Zhu [44] studied Anonymous Walks (AWs), which replace a *node's identity* by the order of its appearance in each walk. The main limitation of AWs is that the position encoding of each node depends only on its specific walk, missing the dependency and correlation of different sampled walks [32]. To mitigate this drawback, Wang et al. [32] suggest replacing node identities with the hitting counts of the nodes based on a set of sampled walks. In addition to the fact that this method is designed for walks on simple graphs, there are two main challenges to adopting it for SetWalks: ① SetWalks are a sequence of hyperedges, so we need an encoding for the position of hyperedges. Natural attempts to replace hyperedges' identity with the hitting counts of the hyperedges based on a set of sampled SetWalks, misses the similarity of hyperedges with many same nodes. ② Each hyperedge is a set of vertices and natural attempts to encode its nodes' positions and aggregate them to obtain a position encoding of the hyperedge requires a permutation invariant pooling strategy. This pooling strategy also requires consideration of the higher-order dependencies between obtained nodes' position encodings to take advantage of higher-order interactions (see Theorem 2). To address these challenges we present a set-based anonymization process for SetWalks. Given a hyperedge $e_0 = \{u_1, \dots, u_k\}$, let w_0 be any node in e_0. For each node w that appears on at least one SetWalk in $\bigcup_{i=1}^{k} \mathcal{S}(u_i)$, we assign a relative, node-dependent node identity, $\mathcal{R}(w, \mathcal{S}(w_0)) \in \mathbb{Z}^m$, as follows:

$$\mathcal{R}(w, \mathcal{S}(w_0))[i] = |\{\text{Sw}|\text{Sw} \in \mathcal{S}(w_0), w \in \text{Sw}[i][0]\}| \quad \forall i \in \{1, 2, \dots, m\}. \tag{2}$$

For each node w we further define $\textsc{Id}(w, e_0) = \{\mathcal{R}(w, \mathcal{S}(u_i))\}_{i=1}^{k}$. Let $\Psi(.) : \mathbb{R}^{d \times d_1} \to \mathbb{R}^{d_2}$ be a pooling function that gets a set of d_1-dimensional vectors and aggregates them to a d_2-dimensional vector. Given two instances of this pooling function, $\Psi_1(.)$ and $\Psi_2(.)$, for each hyperedge $e = \{w_1, w_2, \dots, w_{k'}\}$ that appears on at least one SetWalk in $\bigcup_{i=1}^{k} \mathcal{S}(u_i)$, we assign a relative hyperedge identity as:

$$\textsc{Id}(e, e_0) = \Psi_1\left(\{\Psi_2\left(\textsc{Id}(w_i, e_0)\right)\}_{i=1}^{k'}\right). \tag{3}$$

That is, for each node $w_i \in e$ we first aggregate its relative node-dependent identities (i.e., $\mathcal{R}(w_i, \mathcal{S}(u_j))$) to obtain the relative hyperedge-dependent identity. Then we aggregate these hyperedge-dependent identities for all nodes in e. Since the size of hyperedges can vary, we zero-pad to a fixed length. Note that this zero padding is important to capture the size of the hyperedge. The hyperedge with more zero-padded dimensions has fewer nodes.

This process addresses the first challenge and encodes the position of hyperedges. Unfortunately, many simple and known pooling strategies (e.g., Sum(.), Attn-Sum(.), Mean(.), etc.) can cause missing information when applied to hypergraphs. We formalize this in the following theorem:

Theorem 2. *Given an arbitrary positive integer $k \in \mathbb{Z}^+$, let $\Psi(.)$ be a pooling function such that for any set $S = \{w_1, \dots, w_d\}$:*

$$\Psi(S) = \sum_{\substack{S' \subseteq S \\ |S'|=k}} f(S'), \tag{4}$$

where f is some function. Then the pooling function can cause missing information, limiting the expressiveness of the method to applying to the CE of the hypergraph.

While simple concatenation does not suffer from this undesirable property, it is not permutation invariant. To overcome these challenges, we design an all-MLP permutation invariant pooling function, SetMixer, that not only captures higher-order dependencies of set elements but also captures dependencies across the number of times that a node appears at a certain position in SetWalks.

SetMixer. MLP-Mixer [43] is a family of models based on multi-layer perceptrons, widely used in the computer vision community, that are simple, amenable to efficient implementation, and robust to over-squashing and long-term dependencies (unlike Rnns and attention mechanisms) [43, 54]. However, the token-mixer phase of these methods is sensitive to the order of the input (see Appendix A). To address this limitation, inspired by MLP-Mixer [43], we design SetMixer as follows: Let $S = \{\mathbf{v}_1, \dots, \mathbf{v}_d\}$, where $\mathbf{v}_i \in \mathbb{R}^{d_1}$, be the input set and $\mathbf{V} = [\mathbf{v}_1, \dots, \mathbf{v}_d] \in \mathbb{R}^{d \times d_1}$ be its matrix representation:

$$\Psi(S) = \textsc{Mean}\left(\mathbf{H}_{\text{token}} + \sigma\left(\texttt{LayerNorm}\left(\mathbf{H}_{\text{token}}\right)\mathbf{W}_s^{(1)}\right)\mathbf{W}_s^{(2)}\right), \tag{5}$$

where

$$\mathbf{H}_{\text{token}} = \mathbf{V} + \sigma\left(\texttt{Softmax}\left(\texttt{LayerNorm}\left(\mathbf{V}\right)^T\right)\right)^T. \tag{6}$$

Here $\mathbf{W}_s^{(1)}$ and $\mathbf{W}_s^{(2)}$ are learnable parameters, $\sigma(.)$ is an activation function (we use GeLU [76] in our experiments), and `LayerNorm` is layer normalization [77]. Equation 5 is the channel mixer and Equation 6 is the token mixer. The main intuition of SetMixer is to use the `Softmax`(.) function to bind token-wise information in a non-parametric manner, avoiding permutation variant operations in the token mixer. We formally prove the following theorem in Appendix E.3.

Theorem 3. SetMixer *is permutation invariant.*

Based on the above theorem, SetMixer can overcome the challenges we discussed earlier. Accordingly, in our anonymization process, we use $\Psi(.) = \textsc{SetMixer}(.)$ in Equation 3 to hide hyperedge identities. Next, we guarantee that our anonymization process does not depend on hyperedges or nodes identities, which justifies the claim of inductiveness of our model:

Proposition 1. *Given two (potential) hyperedges $e_0 = \{u_1, \dots, u_k\}$ and $e'_0 = \{u'_1, \dots, u'_k\}$, if there exists a bijective mapping π between node identities such that for each* SetWalk *like* $\textsc{Sw} \in \bigcup_{i=1}^{k} \mathcal{S}(u_i)$ *can be mapped to one* SetWalk *like* $\textsc{Sw}' \in \bigcup_{i=1}^{k} \mathcal{S}(u'_i)$*, then for each hyperedge $e = \{w_1, \dots, w_{k'}\}$ that appears in at least one* SetWalk *in $\bigcup_{i=1}^{k} \mathcal{S}(u_i)$, we have $\textsc{Id}(e, e_0) = \textsc{Id}(\pi(e), e'_0)$, where $\pi(e) = \{\pi(w_1), , \dots, \pi(w_{k'})\}$.*

Finally, we guarantee that our anonymization approach is more expressive than existing anonymization process [32, 44] when applying to the CE of the hypergraphs:

Theorem 4. *The set-based anonymization method is more expressive than any existing anonymization strategies on the CE of the hypergraph. More precisely, there exists a pair of hypergraphs $\mathcal{G}_1 = (\mathcal{V}_1, \mathcal{E}_1)$ and $\mathcal{G}_2 = (\mathcal{V}_2, \mathcal{E}_2)$ with different structures (i.e., $\mathcal{G}_1 \ncong \mathcal{G}_2$) that are distinguishable by our anonymization process and are not distinguishable by the CE-based methods.*

3.4 SetWalk Encoding

Previous walk-based methods [32, 33, 74] view a walk as a sequence of nodes. Accordingly, they plug nodes' positional encoding in a Rnn [78] or Transformer [79] to obtain the encoding of each walk. However, in addition to the computational cost of Rnn and Transformers, they suffer from over-squashing and fail to capture long-term dependencies. To this end, we design a simple and low-cost SetWalk encoding procedure which uses two steps: ① A time encoding module to distinguish different timestamps, and ② A mixer module to summarize temporal and structural information extracted by SetWalks.

Time Encoding. We follow previous studies [32, 80] and adopt random Fourier features [81, 82] to encode time. However, these features are periodic, so they capture only periodicity in the data. We add a learnable linear term to the feature representation of the time encoding. We encode a given time t as follows:

$$\mathrm{T}(t) = (\omega_l t + \mathbf{b}_l) \,||\, \cos(t\omega_p), \tag{7}$$

where $\omega_l, \mathbf{b}_l \in \mathbb{R}$ and $\omega_p \in \mathbb{R}^{d_2-1}$ are learnable parameters and || shows concatenation.

Mixer Module. To summarize the information in each SetWalk, we use a MLP-Mixer [43] on the sequence of hyperedges in a SetWalk as well as their corresponding encoded timestamps. Contrary to the anonymization process, where we need a permutation invariant procedure, here, we need a permutation variant procedure since the appearance order of hyperedges in a SetWalk is important. Given a (potential) hyperedge $e_0 = \{u_1, \dots, u_k\}$, we first assign $\textsc{Id}(e, e_0)$ to each hyperedge e that appears on at least one sampled SetWalk starting from e_0 (Equation 3). Given a SetWalk, $\mathrm{Sw} : (e_1, t_{e_1}) \to \cdots \to (e_m, t_{e_m})$, we let $\mathbf{E}$ be a matrix that $\mathbf{E}_i = \textsc{Id}(e_i, e_0) || \mathrm{T}(t_{e_i})$:

$$\textsc{Enc}(\mathrm{Sw}) = \textsc{Mean}\left(\mathbf{H}_{\text{token}} + \sigma\left(\texttt{LayerNorm}\left(\mathbf{H}_{\text{token}}\right)\mathbf{W}^{(1)}_{\text{channel}}\right)\mathbf{W}^{(2)}_{\text{channel}}\right), \tag{8}$$

where

$$\mathbf{H}_{\text{token}} = \mathbf{E} + \mathbf{W}^{(2)}_{\text{token}}\sigma\left(\mathbf{W}^{(1)}_{\text{token}}\texttt{LayerNorm}\left(\mathbf{E}\right)\right). \tag{9}$$

3.5 Training

In the training phase, for each hyperedge in the training set, we adopt the commonly used negative sample generation method [57] to generate a negative sample. Next, for each hyperedge in the training set such as $e_0 = \{u_1, u_2, \dots, u_k\}$, including both positive and negative samples, we sample M SetWalks with length m starting from each $u_i \in e_0$ to construct $\mathcal{S}(u_i)$. Next, we anonymize each hyperedge that appears in at least one SetWalk in $\bigcup_{i=1}^{k} \mathcal{S}(u_i)$ by Equation 3 and then use the Mixer module to encode each $\mathrm{Sw} \in \bigcup_{i=1}^{k} \mathcal{S}(u_i)$. To encode each node $u_i \in e_0$, we use $\textsc{Mean}(.)$ pooling over SetWalks in $\mathcal{S}(u_i)$. Finally, to encode e_0 we use SetMixer to mix obtained node encodings. For hyperedge prediction, we use a 2-layer perceptron over the hyperedge encodings to make the final prediction. We discuss node classification in Appendix G.2.

4 Experiments

We evaluate the performance of our model on two important tasks: hyperedge prediction and node classification (see Appendix G.2) in both inductive and transductive settings. We then discuss the importance of our model design and the significance of each component in CAt-Walk.

4.1 Experimental Setup

Baselines. We compare our method to eight previous state-of-the-art baselines on the hyperedge prediction task. These methods can be grouped into three categories: ① Deep hypergraph learning methods including HyperSAGCN [25], NHP [83], and CHESHIRE [10]. ② Shallow methods including HPRA [84] and HPLSF [85]. ③ CE methods: CE-CAW [32], CE-EvolveGCN [86] and CE-GCN [87]. Details on these models and hyperparameters used appear in Appendix F.2.

Datasets. We use 10 available benchmark datasets [5] from the existing hypergraph neural networks literature. These datasets' domains include drug networks (i.e., NDC [5]), contact networks (i.e., High School [88] and Primary School [89]), the US. Congress bills network [90, 91], email networks (i.e., Email Enron [5] and Email Eu [92]), and online social networks (i.e., Question Tags and Users-Threads [5]). Detailed descriptions of these datasets appear in Appendix F.1.

Evaluation Tasks. We focus on Hyperedge Prediction: In the transductive setting, we train on the temporal hyperedges with timestamps less than or equal to T_{train} and test on those with timestamps greater than T_{train}. Inspired by Wang et al. [32], we consider two inductive settings. In the **Strongly Inductive** setting, we predict hyperedges consisting of some unseen nodes. In the **Weakly Inductive** setting,we predict hyperedges with *at least* one seen and some unseen nodes. We first follow the procedure used in the transductive setting, and then we randomly select 10% of the nodes and remove

Table 1: Performance on hyperedge prediction: Mean AUC (%) ± standard deviation. Boldfaced letters shaded blue indicate the best result, while gray shaded boxes indicate results within one standard deviation of the best result. N/A: the method has numerical precision or computational issues.

	Methods	NDC Class	High School	Primary School	Congress Bill	Email Enron	Email Eu	Question Tags	Users-Threads
Inductive					Strongly Inductive				
	CE-GCN	52.31 ± 2.99	60.54 ± 2.06	52.34 ± 2.75	49.18 ± 3.61	63.04 ± 1.80	52.76 ± 2.41	56.10 ± 1.88	57.91 ± 1.56
	CE-EvolveGCN	49.78 ± 3.13	46.12 ± 3.83	58.01 ± 2.56	54.00 ± 1.84	57.31 ± 4.19	44.16 ± 1.27	64.08 ± 2.75	52.00 ± 2.32
	CE-CAW	76.45 ± 0.29	83.73 ± 1.42	80.31 ± 1.46	75.38 ± 1.25	70.81 ± 1.13	72.99 ± 0.20	70.14 ± 1.89	73.12 ± 1.06
	NHP	70.53 ± 4.95	65.29 ± 3.80	70.86 ± 3.42	69.82 ± 2.19	49.71 ± 6.09	65.35 ± 2.07	68.23 ± 3.34	71.83 ± 2.64
	HyperSAGCN	79.05 ± 2.48	88.12 ± 3.01	80.13 ± 1.38	79.51 ± 1.27	73.09 ± 2.60	78.01 ± 1.24	73.66 ± 1.95	73.94 ± 2.57
	CHESHIRE	72.24 ± 2.63	82.54 ± 0.88	77.26 ± 1.01	79.43 ± 1.58	70.03 ± 2.55	69.98 ± 2.71	N/A	76.99 ± 2.82
	CAt-Walk	**98.89 ± 1.82**	**96.03 ± 1.50**	**95.32 ± 0.89**	**93.54 ± 0.56**	**73.45 ± 2.92**	**91.68 ± 2.78**	**88.03 ± 3.38**	**89.84 ± 6.02**
					Weakly Inductive				
	CE-GCN	51.80 ± 3.29	50.33 ± 3.40	52.19 ± 2.54	52.38 ± 2.75	50.81 ± 2.87	49.60 ± 3.96	55.13 ± 2.76	57.06 ± 3.16
	CE-EvolveGCN	55.39 ± 5.16	57.85 ± 3.51	51.50 ± 4.07	55.63 ± 3.41	45.66 ± 2.10	52.44 ± 2.38	61.79 ± 1.63	55.81 ± 2.54
	CE-CAW	77.61 ± 1.05	83.77 ± 1.41	82.98 ± 1.06	79.51 ± 0.94	**80.54 ± 1.02**	73.54 ± 1.19	77.29 ± 0.86	80.79 ± 0.82
	NHP	75.17 ± 2.02	67.25 ± 5.19	71.92 ± 1.83	69.58 ± 4.07	60.38 ± 4.45	67.19 ± 4.33	70.46 ± 3.52	76.44 ± 1.90
	HyperSAGCN	79.45 ± 2.18	88.53 ± 1.26	85.08 ± 1.45	80.12 ± 2.00	78.86 ± 0.63	77.26 ± 2.09	78.15 ± 1.41	75.38 ± 1.43
	CHESHIRE	79.03 ± 1.24	88.40 ± 1.06	83.55 ± 1.27	79.67 ± 0.83	74.53 ± 0.91	77.31 ± 0.95	N/A	81.27 ± 0.85
	CAt-Walk	**99.16 ± 1.08**	**94.68 ± 2.37**	**96.53 ± 1.39**	**98.38 ± 0.21**	64.11 ± 7.96	**91.98 ± 2.41**	**90.28 ± 2.81**	**97.15 ± 1.81**
Transductive	HPRA	70.83 ± 0.01	94.91 ± 0.00	89.86 ± 0.06	79.48 ± 0.03	78.62 ± 0.00	72.51 ± 0.00	83.18 ± 0.00	70.49 ± 0.02
	HPLSF	76.19 ± 0.82	92.14 ± 0.29	88.57 ± 1.09	79.31 ± 0.52	75.73 ± 0.05	75.27 ± 0.31	83.45 ± 0.93	74.38 ± 1.11
	CE-GCN	66.83 ± 3.74	62.99 ± 3.02	59.14 ± 3.87	64.42 ± 3.11	58.06 ± 3.80	64.19 ± 2.79	55.18 ± 5.12	62.78 ± 2.69
	CE-EvolveGCN	67.08 ± 3.51	65.19 ± 2.26	63.15 ± 1.32	69.30 ± 2.27	69.98 ± 5.38	64.36 ± 4.17	72.56 ± 1.72	68.55 ± 2.26
	CE-CAW	76.30 ± 0.84	81.63 ± 0.97	86.53 ± 0.84	76.99 ± 1.02	79.57 ± 0.14	78.19 ± 1.10	81.73 ± 2.48	80.86 ± 0.45
	NHP	82.39 ± 2.81	76.85 ± 3.08	80.04 ± 3.42	80.27 ± 2.53	63.17 ± 3.79	78.90 ± 4.39	79.14 ± 3.36	82.33 ± 1.02
	HyperSAGCN	80.76 ± 2.64	94.98 ± 1.30	90.77 ± 2.05	82.84 ± 1.61	**83.59 ± 0.98**	79.61 ± 2.35	84.07 ± 2.50	79.62 ± 2.04
	CHESHIRE	84.91 ± 1.05	95.11 ± 0.94	91.62 ± 1.18	86.81 ± 1.24	82.27 ± 0.86	86.38 ± 1.23	N/A	82.75 ± 1.99
	CAt-Walk	**98.72 ± 1.38**	**95.30 ± 0.43**	**97.91 ± 3.30**	**88.15 ± 1.46**	80.45 ± 5.30	**96.74 ± 1.28**	**91.63 ± 1.41**	**93.51 ± 1.27**

Table 2: Ablation study on CAt-Walk. AUC scores on inductive hyperedge prediction.

Methods	High School	Primary School	Users in Threads
CAt-Walk	**96.03 ± 1.50**	**95.32 ± 0.89**	**89.84 ± 6.02**
Replace SetWalk by Random Walk	92.10 ± 2.18	51.56 ± 5.63	53.24 ± 1.73
Remove Time Encoding	95.94 ± 0.19	86.80 ± 6.33	70.58 ± 9.32
Replace SetMixer by Mean(.)	94.58 ± 1.22	95.14 ± 4.36	63.59 ± 5.26
Replace MLP-Mixer by Rnn	92.85 ± 1.53	50.29 ± 4.07	58.11 ± 1.60
fix $\alpha = 0$	74.06 ± 14.9	58.3 ± 18.62	74.41 ± 10.69

all hyperedges that include them from the training set. We then remove all hyperedges with seen nodes from the validation and testing sets. For dynamic node classification, see Appendix G.2. For all datasets, we fix $T_{\text{train}} = 0.7\,T$, where T is the last timestamp. To evaluate the models' performance we follow the literature and use Area Under the ROC curve (AUC) and Average Precision (AP).

4.2 Results and Discussion

Hyperedge Prediction. We report the results of CAt-Walk and baselines in Table 1 and Appendix G. The results show that CAt-Walk achieves the best overall performance compared to the baselines in both transductive and inductive settings. In the transductive setting, not only does our method outperform baselines in all but one dataset, but it achieves near perfect results (i.e., ≈ 98.0) on the NDC and Primary School datasets. In the Weakly Inductive setting, our model achieves high scores (i.e., > 91.5) in all but one dataset, while most baselines perform poorly as they are not designed for the inductive setting and do not generalize well to unseen nodes or patterns. In the Strongly Inductive setting, CAt-Walk still achieves high AUC (i.e., > 90.0) on most datasets and outperforms baselines on *all* datasets. There are three main reasons for CAt-Walk's superior performance: ① Our SetWalks, capture higher-order patterns. ② CAt-Walk incorporates temporal properties (both from SetWalks and our time encoding module), thus learning underlying dynamic laws of the network. The other temporal methods (CE-CAW and CE-EvolveGCN) are CE-based methods, limiting their ability to capture higher-order patterns. ③ CAt-Walk 's a set-based anonymization process that avoid using node and hyperedge identities allows it to generalize to unseen patterns and nodes.

Ablation Studies. We next conduct ablation studies on the High School, Primary School, and Users-Threads datasets to validate the effectiveness of CAt-Walk's critical components. Table 2 shows AUC results for inductive hyperedge prediction. The first row reports the performance of the

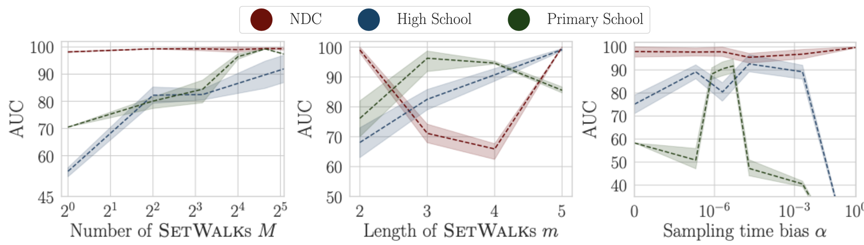

Figure 4: Hyperparameter sensitivity in CAt-Walk. AUC on inductive test hyperedges are reported.

complete CAt-Walk implementation. Each subsequent row shows results for CAt-Walk with one module modification: row 2 replace SetWalk by hypergraph walk, row 3 removes the time encoding module, row 4 replaces SetMixer with Mean(.) pooling, 5 replaces the MLP-Mixer module with a Rnn instead (see Appendix G for more experiments on the significance of using MLP-Mixer in walk encoding), and row 6 replaces the hyperparameter α with uniform sampling of hyperedges over all time periods. These results show that each component is critical for achieving CAt-Walk's superior performance. The greatest contribution comes from SetWalk, MLP-Mixer in walk encoding, α in temporal hyperedge sampling, and SetMixer pooling, respectively.

Hyperparameter Sensitivity. We analyze the effect of hyperparameters used in CAt-Walk, including temporal bias coefficient α, SetWalk length m, and sampling number M. The mean AUC performance on all inductive test hyperedges is reported in Figure 4. As expected, the left figure shows that increasing the number of sampled SetWalks produces better performance. The main reason is that the model has more extracted structural and temporal information from the network. Also, notably, we observe that only a small number of sampled SetWalks are needed to achieve competitive performance: in the best case 1 and in the worst case 16 sampled SetWalks per each hyperedge are needed. The middle figure reports the effect of the length of SetWalks on performance. These results show that performance peaks at certain SetWalk lengths and the exact value varies with the dataset. That is, longer SetWalks are required for the networks that evolve according to more complicated laws encoded in temporal higher-order motifs. The right figure shows the effect of the temporal bias coefficient α. Results suggest that α has a dataset-dependent optimal interval. That is, a small α suggests an almost uniform sampling of interaction history, which results in poor performance when the short-term dependencies (interactions) are more important in the dataset. Also, very large α might damage performance as it makes the model focus on the most recent few interactions, missing long-term patterns.

More experimental results, including time complexity and scalability, can be found in Appendix G.

5 Conclusion, Limitation, and Future Work

We present CAt-Walk, an inductive hypergraph representation learning that learns both higher-order patterns and the underlying dynamic laws of temporal hypergraphs. CAt-Walk uses SetWalks, a new temporal, higher-order random walk on hypergraphs that are provably more expressive than existing walks on hypergraphs, to extract temporal higher-order motifs from hypergraphs. CAt-Walk then uses a two-step, set-based anonymization process to establish the correlation between the extracted motifs. We further design a permutation invariant pooling strategy, SetMixer, for aggregating nodes' positional encodings in a hyperedge to obtain hyperedge level positional encodings. Consequently, the experimental results show that CAt-Walk i produces superior performance compared to the state-of-the-art in temporal hyperedge prediction tasks, and ii competitive performance in temporal node classification tasks. These results suggest many interesting directions for future studies: Using CAt-Walk as a positional encoder in existing anomaly detection frameworks to design an inductive anomaly detection method on hypergraphs. There are, however, a few limitations: Currently, CAt-Walk uses *fixed-length* SetWalks, which might cause suboptimal performance. Developing a procedure to learn from SetWalks of varying lengths might produce better results.

References

[1] Serina Chang, Emma Pierson, Pang Wei Koh, Jaline Gerardin, Beth Redbird, David Grusky, and Jure Leskovec. Mobility network models of covid-19 explain inequities and inform

reopening. *Nature*, 589(7840):82–87, 2021.

[2] Martino Ciaperoni, Edoardo Galimberti, Francesco Bonchi, Ciro Cattuto, Francesco Gullo, and Alain Barrat. Relevance of temporal cores for epidemic spread in temporal networks. *Scientific reports*, 10(1):1–15, 2020.

[3] Joakim Skarding, Bogdan Gabrys, and Katarzyna Musial. Foundations and modeling of dynamic networks using dynamic graph neural networks: A survey. *IEEE Access*, 9:79143–79168, 2021.

[4] Antonio Longa, Veronica Lachi, Gabriele Santin, Monica Bianchini, Bruno Lepri, Pietro Lio, Franco Scarselli, and Andrea Passerini. Graph neural networks for temporal graphs: State of the art, open challenges, and opportunities. *arXiv preprint arXiv:2302.01018*, 2023.

[5] 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. ISSN 0027-8424. doi: 10.1073/pnas.1800683115.

[6] Federico Battiston, Enrico Amico, Alain Barrat, Ginestra Bianconi, Guilherme Ferraz de Arruda, Benedetta Franceschiello, Iacopo Iacopini, Sonia Kéfi, Vito Latora, Yamir Moreno, Micah M. Murray, Tiago P. Peixoto, Francesco Vaccarino, and Giovanni Petri. The physics of higher-order interactions in complex systems. *Nature Physics*, 17(10):1093–1098, Oct 2021. ISSN 1745-2481. doi: 10.1038/s41567-021-01371-4. URL https://doi.org/10.1038/s41567-021-01371-4.

[7] Yuanzhao Zhang, Maxime Lucas, and Federico Battiston. Higher-order interactions shape collective dynamics differently in hypergraphs and simplicial complexes. *Nature Communications*, 14(1):1605, Mar 2023. ISSN 2041-1723. doi: 10.1038/s41467-023-37190-9. URL https://doi.org/10.1038/s41467-023-37190-9.

[8] Eun-Sol Kim, Woo Young Kang, Kyoung-Woon On, Yu-Jung Heo, and Byoung-Tak Zhang. Hypergraph attention networks for multimodal learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 14581–14590, 2020.

[9] Yichao Yan, Jie Qin, Jiaxin Chen, Li Liu, Fan Zhu, Ying Tai, and Ling Shao. Learning multi-granular hypergraphs for video-based person re-identification. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2899–2908, 2020.

[10] Can Chen, Chen Liao, and Yang-Yu Liu. Teasing out missing reactions in genome-scale metabolic networks through hypergraph learning. *Nature Communications*, 14(1):2375, 2023.

[11] Guobo Xie, Yinting Zhu, Zhiyi Lin, Yuping Sun, Guosheng Gu, Jianming Li, and Weiming Wang. Hbrwrlda: predicting potential lncrna–disease associations based on hypergraph bi-random walk with restart. *Molecular Genetics and Genomics*, 297(5):1215–1228, 2022.

[12] Junliang Yu, Hongzhi Yin, Jundong Li, Qinyong Wang, Nguyen Quoc Viet Hung, and Xiangliang Zhang. Self-supervised multi-channel hypergraph convolutional network for social recommendation. In *Proceedings of the web conference 2021*, pages 413–424, 2021.

[13] Dingqi Yang, Bingqing Qu, Jie Yang, and Philippe Cudre-Mauroux. Revisiting user mobility and social relationships in lbsns: a hypergraph embedding approach. In *The world wide web conference*, pages 2147–2157, 2019.

[14] Junren Pan, Baiying Lei, Yanyan Shen, Yong Liu, Zhiguang Feng, and Shuqiang Wang. Characterization multimodal connectivity of brain network by hypergraph gan for alzheimer's disease analysis. In *Pattern Recognition and Computer Vision: 4th Chinese Conference, PRCV 2021, Beijing, China, October 29–November 1, 2021, Proceedings, Part III 4*, pages 467–478. Springer, 2021.

[15] Li Xiao, Junqi Wang, Peyman H Kassani, Yipu Zhang, Yuntong Bai, Julia M Stephen, Tony W Wilson, Vince D Calhoun, and Yu-Ping Wang. Multi-hypergraph learning-based brain functional connectivity analysis in fmri data. *IEEE transactions on medical imaging*, 39(5): 1746–1758, 2019.

[16] Yifan Feng, Haoxuan You, Zizhao Zhang, Rongrong Ji, and Yue Gao. Hypergraph neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 3558–3565, 2019.

[17] Dengyong Zhou, Jiayuan Huang, and Bernhard Schölkopf. Learning with hypergraphs: Clustering, classification, and embedding. In B. Schölkopf, J. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems*, volume 19. MIT Press, 2006. URL https://proceedings.neurips.cc/paper_files/paper/2006/file/dff8e9c2ac33381546d96deea9922999-Paper.pdf.

[18] Naganand Yadati, Madhav Nimishakavi, Prateek Yadav, Vikram Nitin, Anand Louis, and Partha Talukdar. Hypergcn: A new method for training graph convolutional networks on hypergraphs. *Advances in neural information processing systems*, 32, 2019.

[19] Sameer Agarwal, Jongwoo Lim, Lihi Zelnik-Manor, Pietro Perona, David Kriegman, and Serge Belongie. Beyond pairwise clustering. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 2, pages 838–845. IEEE, 2005.

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

[21] Matthias Hein, Simon Setzer, Leonardo Jost, and Syama Sundar Rangapuram. The total variation on hypergraphs - learning on hypergraphs revisited. In C.J. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc., 2013. URL https://proceedings.neurips.cc/paper_files/paper/2013/file/8a3363abe792db2d8761d6403605aeb7-Paper.pdf.

[22] Pan Li and Olgica Milenkovic. Submodular hypergraphs: p-laplacians, Cheeger inequalities and spectral clustering. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 3014–3023. PMLR, 10–15 Jul 2018. URL https://proceedings.mlr.press/v80/li18e.html.

[23] I (Eli) Chien, Huozhi Zhou, and Pan Li. hs^2: Active learning over hypergraphs with pointwise and pairwise queries. In Kamalika Chaudhuri and Masashi Sugiyama, editors, *Proceedings of the Twenty-Second International Conference on Artificial Intelligence and Statistics*, volume 89 of *Proceedings of Machine Learning Research*, pages 2466–2475. PMLR, 16–18 Apr 2019. URL https://proceedings.mlr.press/v89/chien19a.html.

[24] Eli Chien, Chao Pan, Jianhao Peng, and Olgica Milenkovic. You are allset: A multiset function framework for hypergraph neural networks. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=hpBTIv2uy_E.

[25] Ruochi Zhang, Yuesong Zou, and Jian Ma. Hyper-sagnn: a self-attention based graph neural network for hypergraphs. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=ryeHuJBtPH.

[26] Yuan Luo. SHINE: Subhypergraph inductive neural network. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=IsHRUzXPqhI.

[27] Jing Huang and Jie Yang. Unignn: a unified framework for graph and hypergraph neural networks. In Zhi-Hua Zhou, editor, *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, pages 2563–2569. International Joint Conferences on Artificial Intelligence Organization, 8 2021. doi: 10.24963/ijcai.2021/353. URL https://doi.org/10.24963/ijcai.2021/353. Main Track.

[28] Ghadeer AbuOda, Gianmarco De Francisci Morales, and Ashraf Aboulnaga. Link prediction via higher-order motif features. In *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2019, Würzburg, Germany, September 16–20, 2019, Proceedings, Part I*, pages 412–429. Springer, 2020.

[29] Mayank Lahiri and Tanya Y Berger-Wolf. Structure prediction in temporal networks using frequent subgraphs. In *2007 IEEE Symposium on computational intelligence and data mining*, pages 35–42. IEEE, 2007.

[30] Mahmudur Rahman and Mohammad Al Hasan. Link prediction in dynamic networks using graphlet. In *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2016, Riva del Garda, Italy, September 19-23, 2016, Proceedings, Part I 16*, pages 394–409. Springer, 2016.

[31] Giang H Nguyen, John Boaz Lee, Ryan A Rossi, Nesreen K Ahmed, Eunyee Koh, and Sungchul Kim. Dynamic network embeddings: From random walks to temporal random walks. In *2018 IEEE International Conference on Big Data (Big Data)*, pages 1085–1092. IEEE, 2018.

[32] Yanbang Wang, Yen-Yu Chang, Yunyu Liu, Jure Leskovec, and Pan Li. Inductive representation learning in temporal networks via causal anonymous walks. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=KYPz4YsCPj.

[33] Ming Jin, Yuan-Fang Li, and Shirui Pan. Neural temporal walks: Motif-aware representation learning on continuous-time dynamic graphs. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=NqbktPUkZf7.

[34] Zhining Liu, Dawei Zhou, Yada Zhu, Jinjie Gu, and Jingrui He. Towards fine-grained temporal network representation via time-reinforced random walk. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 4973–4980, 2020.

[35] 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.

[36] Timoteo Carletti, Federico Battiston, Giulia Cencetti, and Duccio Fanelli. Random walks on hypergraphs. *Physical review E*, 101(2):022308, 2020.

[37] Koby Hayashi, Sinan G Aksoy, Cheong Hee Park, and Haesun Park. Hypergraph random walks, laplacians, and clustering. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, pages 495–504, 2020.

[38] Josh Payne. Deep hyperedges: a framework for transductive and inductive learning on hypergraphs, 2019.

[39] Uthsav Chitra and Benjamin Raphael. Random walks on hypergraphs with edge-dependent vertex weights. In *International conference on machine learning*, pages 1172–1181. PMLR, 2019.

[40] Sinan G. Aksoy, Cliff Joslyn, Carlos Ortiz Marrero, Brenda Praggastis, and Emilie Purvine. Hypernetwork science via high-order hypergraph walks. *EPJ Data Science*, 9(1):16, Jun 2020. ISSN 2193-1127. doi: 10.1140/epjds/s13688-020-00231-0. URL https://doi.org/10.1140/epjds/s13688-020-00231-0.

[41] Austin R. Benson, David F. Gleich, and Lek-Heng Lim. The spacey random walk: A stochastic process for higher-order data. *SIAM Review*, 59(2):321–345, 2017. doi: 10.1137/16M1074023. URL https://doi.org/10.1137/16M1074023.

[42] Ankit Sharma, Shafiq Joty, Himanshu Kharkwal, and Jaideep Srivastava. Hyperedge2vec: Distributed representations for hyperedges, 2018. URL https://openreview.net/forum?id=rJ5C67-C-.

[43] Ilya Tolstikhin, Neil Houlsby, Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Thomas Unterthiner, Jessica Yung, Andreas Peter Steiner, Daniel Keysers, Jakob Uszkoreit, Mario Lucic, and Alexey Dosovitskiy. MLP-mixer: An all-MLP architecture for vision. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, 2021. URL https://openreview.net/forum?id=EI2KOXKdnP.

[44] Silvio Micali and Zeyuan Allen Zhu. Reconstructing markov processes from independent and anonymous experiments. *Discrete Applied Mathematics*, 200:108–122, 2016. ISSN 0166-218X. doi: https://doi.org/10.1016/j.dam.2015.06.035. URL https://www.sciencedirect.com/science/article/pii/S0166218X15003212.

[45] Youngjoo Seo, Michaël Defferrard, Pierre Vandergheynst, and Xavier Bresson. Structured sequence modeling with graph convolutional recurrent networks. In *International conference on neural information processing*, pages 362–373. Springer, 2018.

[46] Ling Zhao, Yujiao Song, Chao Zhang, Yu Liu, Pu Wang, Tao Lin, Min Deng, and Haifeng Li. T-gcn: A temporal graph convolutional network for traffic prediction. *IEEE Transactions on Intelligent Transportation Systems*, 21(9):3848–3858, 2019.

[47] Hao Peng, Hongfei Wang, Bowen Du, Md Zakirul Alam Bhuiyan, Hongyuan Ma, Jianwei Liu, Lihong Wang, Zeyu Yang, Linfeng Du, Senzhang Wang, et al. Spatial temporal incidence dynamic graph neural networks for traffic flow forecasting. *Information Sciences*, 521:277–290, 2020.

[48] Xiaoyang Wang, Yao Ma, Yiqi Wang, Wei Jin, Xin Wang, Jiliang Tang, Caiyan Jia, and Jian Yu. Traffic flow prediction via spatial temporal graph neural network. In *Proceedings of The Web Conference 2020*, pages 1082–1092, 2020.

[49] Jiaxuan You, Tianyu Du, and Jure Leskovec. Roland: Graph learning framework for dynamic graphs. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '22, page 2358–2366, New York, NY, USA, 2022. Association for Computing Machinery. ISBN 9781450393850. doi: 10.1145/3534678.3539300. URL https://doi.org/10.1145/3534678.3539300.

[50] Farnoosh Hashemi, Ali Behrouz, and Milad Rezaei Hajidehi. Cs-tgn: Community search via temporal graph neural networks. In *Companion Proceedings of the Web Conference 2023*, WWW '23, New York, NY, USA, 2023. Association for Computing Machinery. doi: 10.1145/3543873.3587654. URL https://doi.org/10.1145/3543873.3587654.

[51] 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 & Data Mining*, KDD '19, page 1269–1278, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450362016. doi: 10.1145/3292500.3330895. URL https://doi.org/10.1145/3292500.3330895.

[52] Ali Behrouz and Margo Seltzer. Anomaly detection in multiplex dynamic networks: from blockchain security to brain disease prediction. In *NeurIPS 2022 Temporal Graph Learning Workshop*, 2022. URL https://openreview.net/forum?id=UDGZDfwmay.

[53] Sudhanshu Chanpuriya, Ryan A. Rossi, Sungchul Kim, Tong Yu, Jane Hoffswell, Nedim Lipka, Shunan Guo, and Cameron N Musco. Direct embedding of temporal network edges via time-decayed line graphs. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=Qamz7Q_Ta1k.

[54] Weilin Cong, Si Zhang, Jian Kang, Baichuan Yuan, Hao Wu, Xin Zhou, Hanghang Tong, and Mehrdad Mahdavi. Do we really need complicated model architectures for temporal networks? In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=ayPPc0SyLv1.

[55] Yuhong Luo and Pan Li. Neighborhood-aware scalable temporal network representation learning. In *The First Learning on Graphs Conference*, 2022. URL https://openreview.net/forum?id=EPUtNe7a9ta.

[56] Benjamin Paul Chamberlain, Sergey Shirobokov, Emanuele Rossi, Fabrizio Frasca, Thomas Markovich, Nils Yannick Hammerla, Michael M. Bronstein, and Max Hansmire. Graph neural networks for link prediction with subgraph sketching. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=m1oqEOAozQU.

[57] Can Chen and Yang-Yu Liu. A survey on hyperlink prediction. *arXiv preprint arXiv:2207.02911*, 2022.

[58] Devanshu Arya, Deepak Gupta, Stevan Rudinac, and Marcel Worring. Hyper{sage}: Generalizing inductive representation learning on hypergraphs, 2021. URL https://openreview.net/forum?id=cKnKJcTPRcV.

[59] Devanshu Arya, Deepak K Gupta, Stevan Rudinac, and Marcel Worring. Adaptive neural message passing for inductive learning on hypergraphs. *arXiv preprint arXiv:2109.10683*, 2021.

[60] Sameer Agarwal, Kristin Branson, and Serge Belongie. Higher order learning with graphs. In *Proceedings of the 23rd international conference on Machine learning*, pages 17–24, 2006.

[61] Chaoqi Yang, Ruijie Wang, Shuochao Yao, and Tarek Abdelzaher. Hypergraph learning with line expansion. *arXiv preprint arXiv:2005.04843*, 2020.

[62] Kaize Ding, Jianling Wang, Jundong Li, Dingcheng Li, and Huan Liu. Be more with less: Hypergraph attention networks for inductive text classification. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4927–4936, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.399. URL https://aclanthology.org/2020.emnlp-main.399.

[63] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 855–864, 2016.

[64] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 701–710, 2014.

[65] Xin-Jian Xu, Chong Deng, and Li-Jie Zhang. Hyperlink prediction via local random walks and jensen–shannon divergence. *Journal of Statistical Mechanics: Theory and Experiment*, 2023(3):033402, mar 2023. doi: 10.1088/1742-5468/acc31e. URL https://dx.doi.org/10.1088/1742-5468/acc31e.

[66] Valerio La Gatta, Vincenzo Moscato, Mirko Pennone, Marco Postiglione, and Giancarlo Sperlí. Music recommendation via hypergraph embedding. *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–13, 2022. doi: 10.1109/TNNLS.2022.3146968.

[67] Dingqi Yang, Bingqing Qu, Jie Yang, and Philippe Cudre-Mauroux. Revisiting user mobility and social relationships in lbsns: A hypergraph embedding approach. In *The World Wide Web Conference*, WWW '19, page 2147–2157, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450366748. doi: 10.1145/3308558.3313635. URL https://doi.org/10.1145/3308558.3313635.

[68] Aurélien Ducournau and Alain Bretto. Random walks in directed hypergraphs and application to semi-supervised image segmentation. *Computer Vision and Image Understanding*, 120:91–102, 2014. ISSN 1077-3142. doi: https://doi.org/10.1016/j.cviu.2013.10.012. URL https://www.sciencedirect.com/science/article/pii/S1077314213002038.

[69] Lei Ding and Alper Yilmaz. Interactive image segmentation using probabilistic hypergraphs. *Pattern Recognition*, 43(5):1863–1873, 2010. ISSN 0031-3203. doi: https://doi.org/10.1016/j.patcog.2009.11.025. URL https://www.sciencedirect.com/science/article/pii/S0031320309004440.

[70] Sai Nageswar Satchidanand, Harini Ananthapadmanaban, and Balaraman Ravindran. Extended discriminative random walk: A hypergraph approach to multi-view multi-relational transductive learning. In *Proceedings of the 24th International Conference on Artificial Intelligence*, IJCAI'15, page 3791–3797. AAAI Press, 2015. ISBN 9781577357384.

[71] Jie Huang, Chuan Chen, Fanghua Ye, Jiajing Wu, Zibin Zheng, and Guohui Ling. Hyper2vec: Biased random walk for hyper-network embedding. In Guoliang Li, Jun Yang, Joao Gama, Juggapong Natwichai, and Yongxin Tong, editors, *Database Systems for Advanced Applications*, pages 273–277, Cham, 2019. Springer International Publishing. ISBN 978-3-030-18590-9.

[72] Jie Huang, Xin Liu, and Yangqiu Song. Hyper-path-based representation learning for hyper-networks. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, CIKM '19, page 449–458, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450369763. doi: 10.1145/3357384.3357871. URL https://doi.org/10.1145/3357384.3357871.

[73] Jie Huang, Chuan Chen, Fanghua Ye, Weibo Hu, and Zibin Zheng. Nonuniform hyper-network embedding with dual mechanism. *ACM Trans. Inf. Syst.*, 38(3), may 2020. ISSN 1046-8188. doi: 10.1145/3388924. URL https://doi.org/10.1145/3388924.

[74] Yunyu Liu, Jianzhu Ma, and Pan Li. Neural predicting higher-order patterns in temporal networks. In *Proceedings of the ACM Web Conference 2022*, WWW '22, page 1340–1351, New York, NY, USA, 2022. Association for Computing Machinery. ISBN 9781450390965. doi: 10.1145/3485447.3512181. URL https://doi.org/10.1145/3485447.3512181.

[75] Jiying Zhang, Fuyang Li, Xi Xiao, Tingyang Xu, Yu Rong, Junzhou Huang, and Yatao Bian. Hypergraph convolutional networks via equivalency between hypergraphs and undirected graphs, 2022. URL https://openreview.net/forum?id=zFyCvjXof60.

[76] Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus), 2020.

[77] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[78] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.

[79] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[80] da Xu, chuanwei ruan, evren korpeoglu, sushant kumar, and kannan achan. Inductive representation learning on temporal graphs. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=rJeW1yHYwH.

[81] Da Xu, Chuanwei Ruan, Evren Korpeoglu, Sushant Kumar, and Kannan Achan. Self-attention with functional time representation learning. *Advances in neural information processing systems*, 32, 2019.

[82] Seyed Mehran Kazemi, Rishab Goel, Sepehr Eghbali, Janahan Ramanan, Jaspreet Sahota, Sanjay Thakur, Stella Wu, Cathal Smyth, Pascal Poupart, and Marcus Brubaker. Time2vec: Learning a vector representation of time. *arXiv preprint arXiv:1907.05321*, 2019.

[83] Naganand Yadati, Vikram Nitin, Madhav Nimishakavi, Prateek Yadav, Anand Louis, and Partha Talukdar. Nhp: Neural hypergraph link prediction. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, pages 1705–1714, 2020.

[84] Tarun Kumar, K Darwin, Srinivasan Parthasarathy, and Balaraman Ravindran. Hpra: Hyperedge prediction using resource allocation. In *12th ACM conference on web science*, pages 135–143, 2020.

[85] Ye Xu, Dan Rockmore, and Adam M Kleinbaum. Hyperlink prediction in hypernetworks using latent social features. In *Discovery Science: 16th International Conference, DS 2013, Singapore, October 6-9, 2013. Proceedings 16*, pages 324–339. Springer, 2013.

[86] Aldo Pareja, Giacomo Domeniconi, Jie Chen, Tengfei Ma, Toyotaro Suzumura, Hiroki Kanezashi, Tim Kaler, Tao Schardl, and Charles Leiserson. Evolvegcn: Evolving graph convolutional networks for dynamic graphs. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 5363–5370, 2020.

[87] Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. Simplifying graph convolutional networks. In *International conference on machine learning*, pages 6861–6871. PMLR, 2019.

[88] Rossana Mastrandrea, Julie Fournet, and Alain Barrat. Contact patterns in a high school: A comparison between data collected using wearable sensors, contact diaries and friendship surveys. *PLOS ONE*, 10(9):e0136497, 2015. doi: 10.1371/journal.pone.0136497. URL https://doi.org/10.1371/journal.pone.0136497.

[89] Juliette Stehlé, Nicolas Voirin, Alain Barrat, Ciro Cattuto, Lorenzo Isella, Jean-François Pinton, Marco Quaggiotto, Wouter Van den Broeck, Corinne Régis, Bruno Lina, and Philippe Vanhems. High-resolution measurements of face-to-face contact patterns in a primary school. *PLoS ONE*, 6(8):e23176, 2011. doi: 10.1371/journal.pone.0023176. URL https://doi.org/10.1371/journal.pone.0023176.

[90] James H. Fowler. Connecting the congress: A study of cosponsorship networks. *Political Analysis*, 14(04):456–487, 2006. doi: 10.1093/pan/mpl002. URL https://doi.org/10.1093/pan/mpl002.

[91] James H. Fowler. Legislative cosponsorship networks in the US house and senate. *Social Networks*, 28(4):454–465, oct 2006. doi: 10.1016/j.socnet.2005.11.003. URL https://doi.org/10.1016/j.socnet.2005.11.003.

[92] Hao Yin, Austin R. Benson, Jure Leskovec, and David F. Gleich. Local higher-order graph clustering. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM Press, 2017. doi: 10.1145/3097983.3098069. URL https://doi.org/10.1145/3097983.3098069.

[93] Fan RK Chung. The laplacian of a hypergraph. In "", 1993.

[94] Timoteo Carletti, Duccio Fanelli, and Renaud Lambiotte. Random walks and community detection in hypergraphs. *Journal of Physics: Complexity*, 2(1):015011, 2021.

[95] Linyuan Lu and Xing Peng. High-order random walks and generalized laplacians on hypergraphs. *Internet Mathematics*, 9(1):3–32, 2013.

[96] Chenzi Zhang, Shuguang Hu, Zhihao Gavin Tang, and TH Hubert Chan. Re-revisiting learning on hypergraphs: confidence interval and subgradient method. In *International Conference on Machine Learning*, pages 4026–4034. PMLR, 2017.

[97] T-H Hubert Chan, Zhihao Gavin Tang, Xiaowei Wu, and Chenzi Zhang. Diffusion operator and spectral analysis for directed hypergraph laplacian. *Theoretical Computer Science*, 784: 46–64, 2019.

[98] Pan Li and Olgica Milenkovic. Inhomogeneous hypergraph clustering with applications. *Advances in neural information processing systems*, 30, 2017.

[99] Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. *Advances in neural information processing systems*, 30, 2017.

[100] Marta Garnelo, Dan Rosenbaum, Christopher Maddison, Tiago Ramalho, David Saxton, Murray Shanahan, Yee Whye Teh, Danilo Rezende, and SM Ali Eslami. Conditional neural processes. In *International conference on machine learning*, pages 1704–1713. PMLR, 2018.

[101] David Lopez-Paz, Robert Nishihara, Soumith Chintala, Bernhard Scholkopf, and Léon Bottou. Discovering causal signals in images. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6979–6987, 2017.

[102] Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Russ R Salakhutdinov, and Alexander J Smola. Deep sets. *Advances in neural information processing systems*, 30, 2017.

[103] Juho Lee, Yoonho Lee, Jungtaek Kim, Adam Kosiorek, Seungjin Choi, and Yee Whye Teh. Set transformer: A framework for attention-based permutation-invariant neural networks. In *International conference on machine learning*, pages 3744–3753. PMLR, 2019.

[104] Jinheon Baek, Minki Kang, and Sung Ju Hwang. Accurate learning of graph representations with graph multiset pooling. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=JHcqXGaqiGn.

[105] Ronald Harry Atkin. *Mathematical structure in human affairs*. Heinemann Educational London, 1974.

[106] David I Spivak. Higher-dimensional models of networks. *arXiv preprint arXiv:0909.4314*, 2009.

[107] Jacob Charles Wright Billings, Mirko Hu, Giulia Lerda, Alexey N Medvedev, Francesco Mottes, Adrian Onicas, Andrea Santoro, and Giovanni Petri. Simplex2vec embeddings for community detection in simplicial complexes. *arXiv preprint arXiv:1906.09068*, 2019.

[108] Mustafa Hajij, Kyle Istvan, and Ghada Zamzmi. Cell complex neural networks. In *TDA & Beyond*, 2020. URL https://openreview.net/forum?id=6Tq18ySFpGU.

[109] Celia Hacker. k-simplex2vec: a simplicial extension of node2vec. In *TDA & Beyond*, 2020. URL https://openreview.net/forum?id=Aw9DUXPjq55.

[110] Stefania Ebli, Michaël Defferrard, and Gard Spreemann. Simplicial neural networks. In *TDA & Beyond*, 2020. URL https://openreview.net/forum?id=nPCt39DVIfk.

[111] Maosheng Yang, Elvin Isufi, and Geert Leus. Simplicial convolutional neural networks. In *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 8847–8851. IEEE, 2022.

[112] Eric Bunch, Qian You, Glenn Fung, and Vikas Singh. Simplicial 2-complex convolutional neural networks. In *TDA & Beyond*, 2020. URL https://openreview.net/forum?id=TLbnsKrt6J-.

[113] Christopher Wei Jin Goh, Cristian Bodnar, and Pietro Lio. Simplicial attention networks. In *ICLR 2022 Workshop on Geometrical and Topological Representation Learning*, 2022. URL https://openreview.net/forum?id=ScfRNWkpec.

[114] Mustafa Hajij, Karthikeyan Natesan Ramamurthy, Aldo Guzmán-Sáenz, and Ghada Za. High skip networks: A higher order generalization of skip connections. In *ICLR 2022 Workshop on Geometrical and Topological Representation Learning*, 2022.

[115] Matthias Hein, Simon Setzer, Leonardo Jost, and Syama Sundar Rangapuram. The total variation on hypergraphs - learning on hypergraphs revisited. In C.J. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc., 2013. URL https://proceedings.neurips.cc/paper_files/paper/2013/file/8a3363abe792db2d8761d6403605aeb7-Paper.pdf.

[116] Sameer Agarwal, Kristin Branson, and Serge Belongie. Higher order learning with graphs. In *Proceedings of the 23rd International Conference on Machine Learning*, ICML '06, page 17–24, New York, NY, USA, 2006. Association for Computing Machinery. ISBN 1595933832. doi: 10.1145/1143844.1143847. URL https://doi.org/10.1145/1143844.1143847.

[117] Edmund Ihler, Dorothea Wagner, and Frank Wagner. Modeling hypergraphs by graphs with the same mincut properties. *Inf. Process. Lett.*, 45(4):171–175, mar 1993. ISSN 0020-0190. doi: 10.1016/0020-0190(93)90115-P. URL https://doi.org/10.1016/0020-0190(93)90115-P.

[118] Patrick Kidger, James Morrill, James Foster, and Terry Lyons. Neural controlled differential equations for irregular time series. *Advances in Neural Information Processing Systems*, 33: 6696–6707, 2020.

[119] 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*, WWW '18, page 969–976, Republic and Canton of Geneva, CHE, 2018. International World Wide Web Conferences Steering Committee. ISBN 9781450356404. doi: 10.1145/3184558.3191526. URL https://doi.org/10.1145/3184558.3191526.

[120] Weilin Cong, Si Zhang, Jian Kang, Baichuan Yuan, Hao Wu, Xin Zhou, Hanghang Tong, and Mehrdad Mahdavi. Do we really need complicated model architectures for temporal networks? In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=ayPPc0SyLv1.

Appendix

Table of Contents

A Preliminaries, Backgrounds, and Motivations

We begin by reviewing the preliminaries and background concepts that we refer to in the main paper. Next, we discuss the fundamental differences between our method and techniques from prior work.

A.1 Anonymous Random Walks

Micali and Zhu [44] studied Anonymous Walks (AWs), which replace a *node's identity* by the order of its appearance in each walk. Given a simple network, an AW starts from a node, performs random walks over the graph to collect a sequence of nodes $W : (u_1, u_2, \dots, u_k)$ and then replaces the node identities by their order of appearance in each walk. That is:

$$\textsc{Id}_{\text{AW}}(w_0, W) = |\{u_1, u_2, \dots, u_{k^*}\}| \text{ where } k^* \text{ is the smallest index such that } u_{k^*} = w_0. \tag{10}$$

While this method is a simple anonymization process, it misses the correlation between different walks and assigns new node identities based on only one single walk. The correlation between different walks is more important in temporal networks to assign new node identities, as a single walk cannot capture the frequency of a pattern over time [32]. To this end, Wang et al. [32] design a set-based anonymization process that assigns new node identities based on a set of sampled walks. Given a vertex u, they sample M walks with length m starting from u and store them in S_u. Next, for each node w_0 that appears on at least one walk in S_u, they assign a vector to each node as its hidden identity [32]:

$$g(w_0, S_u)[i] = |\{W | W \in S_u, W[i] = w_0\}| \quad \forall\, i \in \{0, \dots, m\}, \tag{11}$$

where $W[i]$ shows the i-th node in the walk W. This anonymization process not only hides the identity of vertices but it also can establish such hidden identity based on different sampled walks, capturing the correlation between several walks starting from a vertex.

Both of these anonymization processes are designed for graphs with pair-wise interactions, and there are three main challenges in adopting them for hypergraphs: ① To capture higher-order patterns, we use SetWalks, which are a sequence of hyperedges. Accordingly, we need an encoding for the position of hyperedges. A natural attempt to encode the position of hyperedges is to count the position of hyperedges across sampled SetWalks, as CAW [32] does for nodes. However, this approach misses the similarity of hyperedges with many same nodes. That is, given two hyperedges $e_1 = \{u_1, u_2, \dots, u_k\}$ and $e_2 = \{u_1, u_2, \dots, u_k, u_{k+1}\}$. Although we want to encode the position of these two hyperedges, we also want these two hyperedges to have almost the same encoding as they share many vertices. Accordingly, we suggest seeing a hyperedge as a set of vertices. Next, we first encode the position of vertices, and then we aggregate the position encodings of nodes that are connected by a hyperedge to compute the positional encoding of the hyperedge. ② However, since we focus on undirected hypergraphs, the order of hyperedge's vertices in the aggregation process should not affect the hyperedge positional encodings. Therefore, we need a permutation invariant pooling strategy. ③ While several existing studies used simple pooling functions like Mean(.) or Sum(.) [25], these pooling functions do not capture the higher-order dependencies between obtained nodes' position encodings, missing the advantage of higher-order interactions. That is, a pooling function like Mean(.) is a non-parametric method that sees the positional encoding of each node in a hyperedge separately. Therefore, it is unable to aggregate them in a non-linear manner, which depending on the data can miss information. To address challenges ② and ③, we design SetMixer, a permutation invariant pooling strategy that uses MLPs to learn how to aggregate positional encodings of vertices in a hyperedge to compute the hyperedge positional encoding.

A.2 Random Walk on Hypergraphs

Chung [93] starts one of the first research on hypergraph Laplacian and defines the Laplacian of k-uniform hypergraph. Following this direction, Zhou et al. [17] defined a two-step CE-based random walk-based Laplacian for general hypergraphs. Given a node u, in the first step, we uniformly sample a hyperedge e including node u and in the second step, we uniformly sample a node in e. Following this idea, several studies developed more sophisticated (weighted) CE-based random walks on hypergraphs [19]. However, Chitra and Raphael [39] shows that random walks on hypergraphs with edge-independent node weights are limited to capturing pair-wise interactions, making them limited to capturing higher-order information. To this end, they designed an edge-dependent sampling procedure of random walks on hypergraphs. Carletti et al. [36] and Carletti et al. [94] argued that to sample more informative walks from a hypergraph, we are required to consider the degree of hyperedges in measuring the importance of vertices in the first step. Concurrently, some studies discuss the dependencies among hyperedges and define s-th Laplacian based on simple walks on the dual hypergraphs [40, 95]. Finally, more sophisticated random walks with non-linear Laplacian have been designed [22, 96–98].

There are three main drawbacks for existing methods that are addressed by SetWalks: ① None of these methods are designed for temporal hypergraphs and cannot capture the temporal properties of the network. Also, natural attempts to extend them to temporal hypergraphs and let walker uniformly walk over time misses the fact that recent temporal hyperedges are more informative than old ones (see Table 2). To address this issue, SetWalk uses a temporal bias factor in its sampling procedure (Equation 1). ② Existing hypergraph random walks are unable to capture either higher-order interactions of vertices or higher-order dependencies of hyperedges. That is, random walks with edge-independent weights [36] are not able to capture higher-order interactions and are equivalent to simple random walks on the CE of the hypergraph [39]. The expressivity of random walks on hypergraphs with edge-dependent walks is also limited when we have a limited number of sampled walks (see Theorem 1). Finally, defining hypergraph random walk as a random walk on the dual hypergraph also cannot capture the higher-order dependencies of hyperedges (see Appendix C and Appendix D). SetWalk by its nature is able to walk over hyperedges (instead of vertices) and time and can capture higher-order interactions. Also, with a structural bias factor in its sampling procedure, which is based on hyperedge-dependent node weights, it is more informative than a simple random walk on the dual hypergraph, capturing higher-order dependencies of hyperedges. See Appendix C for more discussions.

A.3 MLP-Mixer

MLP-Mixer [43] is a family of models, based on multi-layer perceptions (MLPs), that are simple, amenable to efficient implementation, and robust to over-squashing and long-term dependencies (unlike Rnns, attention mechanisms, and Transformers [79]). The original architecture is designed for image data, where it takes image tokens as inputs. It then encodes them with a linear layer, which is equivalent to a convolutional layer over the image tokens, and updates their representations with a sequence of feed-forward layers applied to image tokens and features. Accordingly, we can divide the architecture of MLP-Mixer into two main parts: ① Token Mixer: The main intuition of the token mixer is to clearly separate the cross-location operations and learn the cross-feature (cross-location) dependencies.② Channel Mixer: The intuition behind the channel mixer is to clearly separate the per-location operations and provide positional invariance, a prominent feature of convolutions. In both Mixer and SetMixer we use the channel mixer as designed in MLP-Mixer. Next, we discuss the token mixer and its limitation in mixing features in a permutation variant manner:

Token Mixer. Let $\mathbf{E}$ be the input of the MLP-Mixer, then the token mixer phase is defined as:

$$\mathbf{H}_{\text{token}} = \mathbf{E} + \mathbf{W}^{(2)}_{\text{token}}\sigma\left(\mathbf{W}^{(1)}_{\text{token}}\texttt{LayerNorm}(\mathbf{E})^T\right)^T, \tag{12}$$

where $\sigma(.)$ is nonlinear activation function (usually GeLU [76]). Since it feeds the input's columns to an MLP, it mixes the cross-feature information, which results in the MLP-Mixer being a permutation variant method. Natural attempts to remove the token mixer or its linear layer, although can result in a permutation invariant method, it misses the cross-feature dependencies, which is the main motivation for using MLP-Mixer architecture. To address this issue, SetMixer uses the $\texttt{Softmax}(.)$ function over features. Using $\texttt{Softmax}$ over features can be seen as a cross-feature normalization, which can capture their dependencies. While $\texttt{Softmax}(.)$ is a non-parametric method that can bind token-wise information, it is also permutation equivariant and as we prove in Appendix E.3, makes the SetMixer permutation invariant.

B Additional Related Work

B.1 Learning (Multi)Set Functions

(Multi)set functions are pooling architectures for (multi)sets with a wide array of applications in many real-world problems including few-shot image classification [99], conditional regression [100], and causality discovery [101]. Zaheer et al. [102] develop DeepSets, a universal approach to parameterize the (multi)set functions. Following this direction, some works design attention mechanisms to learn multiset functions [103], which also inspired Baek et al. [104] to adopt attention mechanisms designed for (multi)set functions in graph representation learning. Finally, Chien et al. [24] build the connection between learning (multi)set functions with propagations on hypergraphs. To the best of our knowledge, SetMixer is the first adaptive permutation invariant pooling strategy for hypergraphs, which sees each hyperedge as a set of vertices and aggregate node encodings by considering their higher-order dependencies.

B.2 Simplicial Complexes Representation Learning

Simplicial complexes can be considered a special case of hypergraphs and are defined as a collection of polytopes such as triangles and tetrahedra, which are called simplices [105]. While these frameworks can be used to represent higher-order relations, simplicial complexes require the downward closure property [106]. That is, every substructure or face of a simplex contained in a complex $\mathcal{K}$ is also in $\mathcal{K}$. Recently, to encode higher-order interactions, representation learning on simplicial complexes has attracted much attention [5, 107–113]. The first group of methods extend node2vec [63] to simplicial complexes with random walks on interactions through Hasse diagrams and simplex connections inside p-chains [107, 109]. With the recent advances in message-passing-based methods, several studies focus on designing neural networks on simplicial complexes [110–113]. Ebli et al. [110] introduced Simplicial neural networks (SNN), generalization of spectral graph convolution to simplicial complexes with higher-order Laplacian matrices. Following this direction, some works propose simplicial convolutional neural networks with different simplicial filters to exploit the relationships in upper- and lower-neighborhoods [111, 112]. Finally, the last group of studies use encoder-decoder architecture as well as message-passing to learn the representation of simplicial complexes [108, 114].

CAt-Walk is different from all these methods in three main aspects: ① Contrary to these methods, CAt-Walk is designed for temporal hypergraphs and is capable of capturing higher-order temporal properties in a streaming manner, avoiding the drawbacks of snapshot-based methods. ② CAt-Walk works in the inductive setting by extracting underlying dynamic laws of the hypergraph, making it generalizable to unseen patterns and nodes. ③ All these methods are designed for simplicial complexes, which are special cases of hypergraphs, while CAt-Walk is designed for general hypergraphs and does not require any assumption of the downward closure property.

B.3 How Does CAt-Walk Differ from Existing Works? (Contributions)

As we discussed in Appendix A.2, existing random walks on hypergraphs are unable to capture either ① higher-order interactions between nodes, or ② higher-order dependencies of hyperedges. Moreover, all these walks are for static hypergraphs and are not able to capture temporal properties. To this end, we design SetWalk a higher-order temporal walk on hypergraphs. Naturally, SetWalks are capable of capturing higher-order patterns as a SetWalkis defined as a sequence of hyperedges. We further design a new sampling procedure with temporal and structural biases, making SetWalks capable of capturing higher-order dependencies of hyperedges. To take advantage of complex information provided by SetWalks as well as training the model in an inductive manner, we design a two-step anonymization process with a novel pooling strategy, called SetMixer. The anonymization process starts with encoding the position of vertices with respect to a set of sampled SetWalks and then aggregates node positional encodings via a non-linear permutation invariant pooling function, SetMixer, to compute their corresponding hyperedge positional encodings. This two-step process lets us capture structural properties while we also care about the similarity of hyperedges. Finally, to take advantage of continuous-time dynamics in data and avoid the limitations of sequential encoding, we design a neural network for temporal walk encoding that leverages a time encoding module to encode time as well as a Mixer module to encode the structure of the walk.

C SetWalk and Random Walk on Hypergraphs

We reviewed existing random walks in Appendix A.2. Here, we discuss how these concepts are different from SetWalks and investigate whether SetWalks are more expressive than these methods.

As we discussed in Sections 1 and 3.2, there are two main challenges for designing random walks on hypergraphs: ① Random walks are a sequence of *pair-wise* interconnected vertices, even though edges in a hypergraph connect *sets* of vertices. ② A sampling probability of a walk on a hypergraph must be different from its sampling probability on the CE of the hypergraph [36–42]. To address these challenges, most existing works on random walks on hypergraphs ignore ① and focus on ② to distinguish the walks on simple graphs and hypergraphs, and ① is relatively unexplored. To this end, we answer the following questions:

Q1: Can ② alone be sufficient to take advantage of higher-order interactions? First, semantically, decomposing hyperedges into sequences of simple pair-wise interactions (CE) loses the semantic meaning of the hyperedges. Consider the collaboration network in Figure 1. When decomposing the hyperedges into pair-wise interactions, both (A, B, C) and (H, G, E) have the same structure (a triangle), while the semantics of these two structures in the data are completely different. That is, (A, B, C) have *all* published a paper together, while each pair of (H, G, E) separately have published a paper. One might argue that although the output of hypergraph random walks and simple random walks on the CE might be the same, the sampling probability of each walk is different and with a large number of samples, our model can distinguish these two structures. In Theorem 1 (proof in Appendix E) we theoretically show that when we have a finite number of hypergraph walk samples, M, there is a hypergraph $\mathcal{G}$ such that with M hypergraph walks, the $\mathcal{G}$ and its CE are not distinguishable. Note that in reality, the bottleneck for the number of sampled walks in machine learning-based methods is memory. Accordingly, even with tuning the number of samples for each dataset, the size of samples is bounded by a small number. This theorem shows that with a limited budget for walk sampling, ② alone is not enough to capture higher-order patterns.

Q2: Can addressing ① alone be sufficient to take advantage of higher-order interactions? To answer this question, we use the extended version of the edge-to-vertex dual graph concept for hypergraphs:

Definition 3 (Dual Hypergraph)**.** *Given a hypergraph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, the dual hypergraph of $\mathcal{G}$ is defined as $\tilde{\mathcal{G}} = (\tilde{\mathcal{V}}, \tilde{\mathcal{E}})$, where $\tilde{\mathcal{V}} = \mathcal{E}$ and a hyperedge $\tilde{e} = \{e_1, e_2, \dots, e_k\} \in \tilde{\mathcal{E}}$ shows that $\bigcap_{i=1}^{k} e_i \neq \emptyset$.*

To address ①, we need to see walks on hypergraphs as a sequence of hyperedges (instead of a sequence of pair-wise connected nodes). One can interpret this as a hypergraph walk on the dual hypergraph. That is, each hypergraph walk on the dual graph is a sequence of $\mathcal{G}$'s hyperedges: $e_1 \to e_2 \to \dots \to e_k$. However, as shown by Chitra and Raphael [39], each walk on hypergraphs with edge-independent weights for sampling vertices is equivalent to a simple walk on the (weighted) CE graph. To this end, addressing ② alone can be equivalent to sample walks on the CE of the dual hypergraph, which misses the higher-order interdependencies of hyperedges and their intersections.

Based on the above discussion, both ① and ② are required to capture higher-order interaction between nodes as well as higher-order interdependencies of hyperedges. The definition of SetWalks (Definition 2) with *structural bias*, introduced in Equation 1, satisfies both ① and ②. In the next section, we discuss how a simple extension of SetWalks can not only be more expressive than all existing walks on hypergraphs and their CEs, but its definition also is universal and all these methods are special cases of extended SetWalk.

C.1 Extension of SetWalks

Random walks on hypergraphs are simple but less expressive methods for extracting network motifs while SetWalks are more complex patterns that provide more expressive motif extraction approaches. One can model the trade-off of simplicity and expressivity to connect all these concepts in a single notion of walks. To establish a connection between SetWalks and existing walks on hypergraphs, as well as a universal random walk model on hypergraphs, we extend SetWalks to r-SetWalks, where parameter r controls the size of hyperedges appear in the walk:

Definition 4 (r-SetWalk)**.** *Given a temporal hypergraph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{X})$, and a threshold $r \in \mathbb{Z}^+$, a r-SetWalk with length ℓ on temporal hypergraph $\mathcal{G}$ is a randomly generated sequence of hyperedges (sets):*

$$\mathrm{Sw}: (e_1, t_{e_1}) \to (e_2, t_{e_2}) \to \dots \to (e_\ell, t_{e_\ell}),$$

where $e_i \in \mathcal{E}$, $|e_i| \le r$, $t_{e_{i+1}} < t_{e_i}$, and the intersection of e_i and e_{i+1} is not empty, $e_i \cap e_{i+1} \neq \emptyset$. In other words, for each $1 \le i \le \ell - 1$: $e_{i+1} \in \mathcal{E}^{t_i}(e_i)$. We use $\mathrm{Sw}[i]$ to denote the i-th hyperedge-time pair in the SetWalk. That is, $\mathrm{Sw}[i][0] = e_i$ and $\mathrm{Sw}[i][1] = t_{e_i}$.

The only difference between this definition and Definition 2 is that r-SetWalk limits hyperedges in the walk to hyperedges with size at most r. The sampling process of r-SetWalks is the same as the SetWalk's (introduced in Section 3.2 and Appendix D) while we only sample hyperedges with size at most r. Now to establish the connection of r-SetWalks and existing walks on hypergraphs, we define the extended version of the clique expansion technique:

Definition 5 (r-Projected Hypergraph)**.** *Given a hypergraph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ and an integer $r \geq 2$, we construct the (weighted) r-projected hypergraph of $\mathcal{G}$ as a hypergraph $\hat{\mathcal{G}}_r = (\mathcal{V}, \hat{\mathcal{E}}_r)$, where for each $e = \{u_1, u_2, \dots, u_k\} \in \mathcal{E}$:*

1. *if $k \leq r$: add e to the $\hat{\mathcal{E}}_r$,*
2. *if $k \geq r+1$: add $e_i = \{u_{i_1}, u_{i_2}, \dots, u_{i_r}\}$ to $\hat{\mathcal{E}}_r$, for every possible $\{i_1, i_2, \dots, i_r\} \subseteq \{1, 2, \dots, k\}$.*

Each of steps 1 or 2 can be done in a weighted manner. In other words, we approximate each hyperedge with a size of more than r with $\binom{k}{r}$ (weighted) hyperedges with size r. For example, when $r = 2$, the 2-projected graph of $\mathcal{G}$ is equivalent to its clique expansion, and $r = \infty$ is the hypergraph itself. Furthermore, we define Union Projected Hyperaph (UP hypergraph) as the union of all r-projected hypergraphs, i.e., $\mathcal{G}^* = (\mathcal{V}, \bigcup_{r=2}^{\infty} \hat{\mathcal{E}}_r)$. Note that the UP hypergraph has downward closure property and is equivalent to the simplicial complex representation of the hypergraph $\mathcal{G}$. The next proposition establishes the universality of the r-SetWalk concept.

Proposition 2. *Edge-independent random walks on hypergraphs [36], edge-dependent random walks on hypergraphs [39], and simple random walks on the CE of hypergraphs are all special cases of r-SetWalk, when applied to the 2-projected graph, UP hypergraph, and 2-projected graph, respectively. Furthermore, all the above methods are less expressive than r-SetWalks.*

The proof of this proposition is in Appendix E.5.

D Efficient Hyperedge Sampling

For sampling SetWalks, inspired by Wang et al. [32], we use two steps: ① Online score computation: we assign a set of scores to each incoming hyperedge. ② Iterative sampling: we use assigned scores in the previous step to sample hyperedges in a SetWalk.

Algorithm 1 Online Score Computation

```
Input: Given a hypergraph G = (V, E) and α ∈ [0, 1]
Output: A probability score for each vertex
P ← ∅;
for (e, t) ∈ E with an increasing order of t do
    P_{e,t}[0] ← exp(αt);
    P_{e,t}[1] ← 0, P_{e,t}[2] ← 0, P_{e,t}[3] ← ∅;
    for u ∈ e do
        for e_n ∈ E^t(u) do
            if e_n is not visited then
                P_{e,t}[2] ← P_{e,t}[2] + exp(φ(e_n, e));
                P_{e,t}[3] ← P_{e,t}[3] ∪ {exp(φ(e_n, e))};
                P_{e,t}[1] ← P_{e,t}[1] + exp(α × t_n);
return P;
```

Online Score Computation. The first part essentially works in an online manner and assigns each new incoming hyperedge e a four-tuple of scores:

$$P_{e,t}[0] = \exp(\alpha \times t), \qquad P_{e,t}[1] = \sum_{(e',t') \in \mathcal{E}^t(e)} \exp(\alpha \times t')$$

$$P_{e,t}[2] = \sum_{(e',t') \in \mathcal{E}^t(e)} \exp(\varphi(e, e')), \qquad P_{e,t}[3] = \{\exp(\varphi(e, e'))\}_{(e',t') \in \mathcal{E}^t(e)}$$

Iterative Sampling. In the iterative sampling algorithm, we use pre-computed scores by Algorithm 1 and sample a hyperedge (e, t) given a previously sampled hyperedge (e_p, t_p). In the next proposition,

Algorithm 2 Iterative SetWalk Sampling

Input: Given a hypergraph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, $\alpha \in [0, 1]$, and previously sampled hyperedge (e_p, t_p)
Output: Next sampled hyperedge (e, t)

for $(e, t) \in \mathcal{E}^{t_p}(e_p)$ with an decreasing order of t **do**
  Sample $b \sim$ Uniform$(0, 1)$;
  Get $P_{e,t}[0], P_{e_p,t_p}[1], P_{e_p,t_p}[2]$ and $\varphi(e, e_p)$ from the output of Algorithm 1;
  $\mathcal{P} \leftarrow$ Normalize $\frac{P_{e,t}[0]}{P_{e_p,t_p}[1]} \times \frac{\exp(\varphi(e,e_p))}{P_{e_p,t_p}[2]}$;
  if $b < \mathcal{P}$ **then return** (e, t);
return (e_X, t_X); ▷ (e_X, t_X) is a dummy empty hyperedge signaling the end of algorithm.

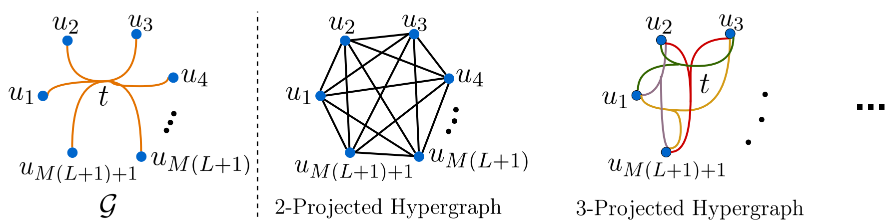

Figure 5: The example of a hypergraph $\mathcal{G}$ and its 2- and 3-projected hypergraphs.

we show that this sampling algorithm samples each hyperedge with the probability mentioned in Section 3.2.

Proposition 3. *Algorithm 2 sample a hyperedge (e, t) after (e_p, t_p) with a probability proportional to $\mathbb{P}[(e, t)|(e_p, t_p)]$ (Equation 1).*

How can this sampling procedure capture higher-order patterns? As discussed in Appendix C, SetWalks on $\mathcal{G}$ can be interpreted as a random walk on the dual hypergraph of $\mathcal{G}$, $\tilde{\mathcal{G}}$. However, a simple (or hyperedge-independent) random walk on the dual hypergraph is equivalent to the walk on the CE of the dual hypergraph [39, 40], missing the higher-order dependencies of hyperedges. Inspired by Chitra and Raphael [39], we use hyperedge-dependent weights $\Gamma : \mathcal{V} \times \mathcal{E} \rightarrow \mathbb{R}^{\geq 0}$ and sample hyperedges with a probability proportional to $\exp\left(\sum_{u \in e \cap e_p} \Gamma(u, e)\Gamma(u, e')\right)$, where e_p is the previously sampled hyperedge. In the dual hypergraph $\tilde{\mathcal{G}} = (\mathcal{E}, \mathcal{V})$, we assign a score $\tilde{\Gamma} : \mathcal{E} \times \mathcal{V} \rightarrow \mathbb{R}^{\geq 0}$ to each pair of (e, u) as $\tilde{\Gamma}(e, u) = \Gamma(u, e)$. Now, a SetWalk with this sampling procedure is equivalent to the edge-dependent hypergraph walk on the dual hypergraph of $\mathcal{G}$ with edge-dependent weight $\tilde{\Gamma}(.)$. Chitra and Raphael [39] show that an edge-dependent hypergraph random walk can capture some information about higher-order interactions and is not equivalent to a simple walk on the weighted CE of the hypergraph. Accordingly, even on the dual hypergraph, SetWalk with this sampling procedure can capture higher-order dependencies of hyperedges and is not equivalent to a simple walk on the CE of the dual hypergraph $\tilde{\mathcal{G}}$. We conclude that, unlike existing random walks on hypergraphs [36, 37, 40, 75], SetWalk can capture both higher-order interactions of nodes, and, based on its sampling procedure, higher-order dependencies of hyperedges.

E Theoretical Results

E.1 Proof of Theorem 1

Theorem 1. *A random SetWalk is equivalent to neither the hypergraph random walk, the random walk on the CE graph, nor the random walk on the SE graph. Also, for a finite number of samples of each, SetWalk is more expressive than existing walks.*

Proof. In this proof, we focus on the hypergraph random walk and simple random walk on the CE. The proof for the SE graph is the same and also it has been proven that the SE graph and the CE of a hypergraph have close (or equal in uniform hypergraphs) Laplacian and have the same expressiveness power in the representation of hypergraphs [115–117].

First, note that each SetWalk can be approximately decomposed to a set of either hypergraph walks, simple random walks, or walk on the SE. Moreover, each of these walks can be mapped to a corresponding SetWalk(but not a bijective mapping), by sampling hyperpedges corresponding to each consecutive pair of nodes in these walks. Accordingly, SetWalks includes the information provided by these walks and so its expressiveness is not less than these methods. To this end, next, we discuss two examples in two different tasks that SetWalks are successful while other walks fail.

① In the first task, we want to see if there is any pair of hypergraphs with different semantics that SetWalks can distinguish them while other walks fail to do so. We construct such hypergraphs. Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a hypergraph with $\mathcal{V} = \{u_1, u_2, \dots, u_N\}$ and $\mathcal{E} = \{(e, t_i)\}_{i=1}^{T}$, where $e = \{u_1, u_2, \dots, u_N\}$ and $t_1 < t_2 < \cdots < t_T$. Also, let $\mathcal{A}$ be an edge-independent hypergraph random walk (or random walk on the CE) sampling algorithm. Chitra and Raphael [39] show that each of these walks is equivalent to a random walk on the weighted CE. Assume that $\xi(.)$ is a function that assigns weights to edges in $\mathcal{G}^* = (\mathcal{V}, \mathcal{E}^*)$, the weighted CE of the $\mathcal{G}$, such that a hypergraph random walk on $\mathcal{G}$ is equivalent to a walk on this weighted CE graph. Next, we construct a weighted hypergraph $\mathcal{G}' = (\mathcal{V}, \mathcal{E}')$ with the same set of vertices but with $\mathcal{E}' = \bigcup_{k=1}^{T} \{((u_i, u_j), t_k)\}_{u_i, u_j \in \mathcal{V}}$, such that each edge $e_{i,j} = (u_i, u_j)$ is associated with a weight $\xi(e_{i,j})$. Clearly, sampling procedure $\mathcal{A}$ on $\mathcal{G}$ and $\mathcal{G}'$ are the same, while they have different semantics. For example, assume that both are collaboration networks. While in $\mathcal{G}$ all vertices have published a single paper together, in the $\mathcal{G}'$ each pair of vertices have published a separate paper together. The proof for the hypergraph random walk with hyperedge-dependent weights is the same, while we construct weights of the hypergraph $\mathcal{G}'$ based on the sampling probability of hyperedges in the hypergraph random walk procedure.

② Next, in the second task, we investigate the expressiveness of these walks for reconstructing hyperedges. That is, we want to see given a perfect classifier, whether these walks can provide enough information to detect higher-order patterns in the network. To this end, we show that for a finite number of samples of each walk, SetWalk is more expressive than all these walks in detecting higher-order patterns. To this end, let M be the maximum number of samples and L be the maximum length of walks, we show that for any $M \geq 2$ and $L \geq 2$ there exists a pair of hypergraphs $\mathcal{G}$, with higher-order interactions, and $\mathcal{G}'$, with pairwise interactions, such that SetWalks can distinguish them, while they are indistinguishable by any of these walks. We construct a temporal hypergraph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ as a hypergraph with $\mathcal{V} = \{u_1, u_2, \dots, u_{M(L+1)+1}\}$ and $\mathcal{E} = \{(e, t_i)\}_{i=1}^{L}$, where $e = \{u_1, u_2, \dots, u_{M(L+1)+1}\}$ and $t_1 < t_2 < \cdots < t_L$. We further construct $\mathcal{G}' = (\mathcal{V}, \mathcal{E}')$ with the same set of vertices but with $\mathcal{E}' = \bigcup_{k=1}^{L} \{((u_i, u_j), t_k)\}_{u_i, u_j \in \mathcal{V}}$. Figure 5 illustrates $\mathcal{G}$ and its projected graphs at a given timestamp $t \in \{t_1, t_2, \dots, t_L\}$.

SetWalk with only one sample, Sw, can distinguish interactions in these two hypergraphs. That is, let $\text{Sw} : (e, t_L) \to (e, t_{L-1}) \to \cdots \to (e, t_1)$ be the sample SetWalk from $\mathcal{G}$ (note that masking the time, this is the only SetWalk on $\mathcal{G}$ so in any case the sampled SetWalk is Sw). Since all interactions in $\mathcal{G}'$ are pairwise, *any* sampled SetWalk on $\mathcal{G}'$, Sw′, only includes pairwise interactions and so $\text{Sw} \neq \text{Sw}'$, in any case. Accordingly, in any case, SetWalk can distinguish interactions in these two hypergraphs.

Since the output of hypergraph random walks, simple walks on the CE, and walks on the SE include pairwise interaction, it seems that they are unable to detect higher-order patterns, so are unable to distinguish these two hypergraphs. However, one might argue that by having a large number of sampled walks and using a perfect classifier, which learned the distribution of sampled random walks and can detect whether a set of sampled walks is from a higher-order interaction, we might be able to detect higher-order interactions. To this end, we next assume that we have a perfect classifier $C(.)$ that can detect whether a set of sampled hypergraph walks, simple walks on the CE, or walks on the SE are sampled from a higher-order structure or pair-wise patterns. Next, we show that hypergraph random walks cannot provide enough information about every vertex for $C(.)$ to detect whether all vertices in $\mathcal{V}$ shape a hyperedge. To this end, assume that we sample $S = \{W_1, W_2, \dots, W_M\}$ walks from hypergraph $\mathcal{G}$ and $S' = \{W'_1, W'_2, \dots, W'_M\}$ walks from hypergraph $\mathcal{G}'$. In the best case scenario, since $C(.)$ is a perfect classifier, it can detect $\mathcal{G}'$ only includes pair-wise interactions based on sampled walk S'. To distinguish these two hypergraphs, we need $C(.)$ to detect sampled walks from $\mathcal{G}$ (i.e., S) that come from a higher-order pattern. For any M sampled walks with length L from $\mathcal{G}$, we observe at most $M \times (L+1)$ vertices, so we have information about at most $M \times (L+1)$ vertices, unable to capture any information about the neighborhood of at least one vertex. Due to the symmetry of vertices, without loss of generality, we can assume that this vertex is u_1. This means that with these M sampled hypergraph random walks with length L, we are not able to provide any information about node u_1 at any timestamp for $C(.)$. Therefore, even a perfect classifier $C(.)$ cannot verify whether u_1

is a part of higher-order interaction or pair-wise interaction, which completes the proof. Note that the proof for the simple random walk is completely the same.

□

Remark 1. *Note that while the first task investigates the expressiveness of these methods with respect to their sampling procedure, the second tasks discuss the limitation and difference in their outputs.*

Remark 2. *Note that in reality, we can have neither an unlimited number of samples nor an unlimited walk length. Also, the upper bound for the number of samples or walk length depends on the RAM of the machine on which the model is being trained. In our experiments, we observe that usually, we cannot sample more than 125 walks with a batch size of 32.*

E.2 Proof of Theorem 2

Theorem 2. *Given an arbitrary positive integer $k \in \mathbb{Z}^+$, let $\Psi(.)$ be a pooling function such that for any set $S = \{w_1, \dots, w_d\}$:*

$$\Psi(S) = \sum_{\substack{S' \subseteq S \\ |S'|=k}} f(S'), \tag{13}$$

where f is some function. Then the pooling function can cause missing information, limiting the expressiveness of the method to applying to the CE of the hypergraph.

Proof. The main intuition of this theorem is that a pooling function needs to capture higher-order dependencies of its input's elements and if it can be decomposed to a summation of functions that capture lower-order dependencies, it misses information. We show that, in the general case for a given $k \in \mathbb{Z}^+$, the pooling function $\Psi(.)$ when applied to a hypergraph $\mathcal{G}$ is at most as expressive as $\Psi(.)$ when applied to the k-projected hypergraph of $\mathcal{G}$ (Definition 5). Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a hypergraph with $\mathcal{V} = \{u_1, u_2, \dots, u_{k+1}\}$ and $\mathcal{E} = \{(\mathcal{V}, t)\} = \{(\{u_1, u_2, \dots, u_{k+1}\}, t)\}$ for a given time t, and $\hat{\mathcal{G}} = (\mathcal{V}, \hat{\mathcal{E}})$ be its k-projected graph, i.e., $\hat{\mathcal{E}} = \{(e_1, t), \dots, (e_{\binom{k+1}{k}}, t)\}$, where $e_i \subset \{u_1, u_2, \dots, u_{k+1}\}$ such that $|e_i| = k$. Applying pooling function $\Psi(.)$ on the hypergraph $\mathcal{G}$ is equivalent to applying $\Psi(.)$ to the hyperedge $(\mathcal{V}, t) \in \mathcal{E}$, which provides $\Psi(\mathcal{V}) = \sum_{i=1}^{k+1} f(e_i)$. On the other hand, applying $\Psi(.)$ on projected graph $\hat{\mathcal{G}}$ means applying it on each hyperedge $e_i \in \hat{\mathcal{E}}$. Accordingly, since for each hyperedge $e_i \in \hat{\mathcal{E}}$ we have $\Psi(e_i) = f(e_i)$, all captured information by pooling function $\Psi(.)$ on $\hat{\mathcal{G}}$ is the set of $S = \{f(e_i)\}_{i=1}^{k+1}$. It is clear that $\Psi(\mathcal{V}) = \sum_{i=1}^{k+1} f(e_i)$ is less informative than $S = \{f(e_i)\}_{i=1}^{k+1}$ as it is the summation of elements in S (in fact, $\Psi(\mathcal{V})$ cannot capture the non-linear combinations of positional encodings of vertices, while S can). Accordingly, the provided information by applying $\Psi(.)$ on $\mathcal{G}$ cannot be more informative than applying $\Psi(.)$ on the $\mathcal{G}$'s k-projected hypergraph. □

Remark 3. *Note that the pooling function $\Psi(.)$ is defined on a (hyper)graph and gets only (hyper)edges as input.*

Remark 4. *Although $\Psi(.) = \textsc{Mean}(.)$ cannot be written as Equation 13, we can simply see that the above proof works for this pooling function as well.*

E.3 Proof of Theorem 3

Theorem 3. SetMixer *is permutation invariant.*

Proof. Let $\pi(S)$ be a given permutation of set S, we aim to show that $\Psi(S) = \Psi(\pi(S))$. We first recall the SetMixer and its two phases: Let $S = \{\mathbf{v}_1, \dots, \mathbf{v}_d\}$, where $\mathbf{v}_i \in \mathbb{R}^{d_1}$, be the input set and $\mathbf{V} = [\mathbf{v}_1, \dots, \mathbf{v}_d]^T \in \mathbb{R}^{d \times d_1}$ be its matrix representation:

$$\Psi(\mathbf{V}) = \textsc{Mean}\left(\mathbf{H}_{\text{token}} + \sigma\left(\texttt{LayerNorm}\left(\mathbf{H}_{\text{token}}\right)\mathbf{W}_s^{(1)}\right)\mathbf{W}_s^{(2)}\right), \quad (\textit{Channel Mixer})$$

where

$$\mathbf{H}_{\text{token}} = \mathbf{V} + \sigma\left(\texttt{Softmax}\left(\texttt{LayerNorm}\left(\mathbf{V}\right)^T\right)\right)^T. \quad (\textit{Token Mixer})$$

Let $\pi(\mathbf{V}) = [\mathbf{v}_{\pi(1)}, \dots, \mathbf{v}_{\pi(d)}]^T$ be a permutation of the input matrix $\mathbf{V}$. In the token mixer phase, None of `LayerNorm`, `Softmax`, and activation function $\sigma(.)$ can affect the order of elements (note that

`Softmax` is applied row-wise). Accordingly, we can see the output of the token mixer is permuted by $\pi(.)$:

$$\begin{aligned}
\mathbf{H}_{\text{token}}(\pi(\mathbf{V})) &= \pi(\mathbf{V}) + \sigma\left(\texttt{Softmax}\left(\texttt{LayerNorm}\left(\pi(\mathbf{V})\right)^{T}\right)\right)^{T} \\
&= \pi(\mathbf{V}) + \pi\left(\sigma\left(\texttt{Softmax}\left(\texttt{LayerNorm}\left(\mathbf{V}\right)^{T}\right)\right)^{T}\right) \\
&= \pi\left(\mathbf{V} + \sigma\left(\texttt{Softmax}\left(\texttt{LayerNorm}\left(\mathbf{V}\right)^{T}\right)\right)^{T}\right) \\
&= \pi\left(\mathbf{H}_{\text{token}}(\mathbf{V})\right). \qquad (14)
\end{aligned}$$

Next, in the channel mixer, by using Equation 14 we have:

$$\begin{aligned}
\Psi(\pi(\mathbf{V})) &= \textsc{Mean}\left(\pi(\mathbf{H}_{\text{token}}) + \sigma\left(\texttt{LayerNorm}\left(\pi(\mathbf{H}_{\text{token}})\right)\mathbf{W}_s^{(1)}\right)\mathbf{W}_s^{(2)}\right) \\
&= \textsc{Mean}\left(\pi(\mathbf{H}_{\text{token}}) + \pi\left(\sigma\left(\texttt{LayerNorm}\left(\mathbf{H}_{\text{token}}\right)\mathbf{W}_s^{(1)}\right)\right)\mathbf{W}_s^{(2)}\right) \\
&= \textsc{Mean}\left(\pi(\mathbf{H}_{\text{token}}) + \pi\left(\sigma\left(\texttt{LayerNorm}\left(\mathbf{H}_{\text{token}}\right)\mathbf{W}_s^{(1)}\right)\mathbf{W}_s^{(2)}\right)\right) \\
&= \textsc{Mean}\left(\pi\left(\mathbf{H}_{\text{token}} + \mathbf{W}_s^{(2)}\sigma\left(\texttt{LayerNorm}\left(\mathbf{H}_{\text{token}}\right)\mathbf{W}_s^{(1)}\right)\right)\right) \\
&= \Psi(\mathbf{V}). \qquad (15)
\end{aligned}$$

In the last step, we use the fact that $\textsc{Mean}(.)$ is permutation invariant. Based on Equation 15 we can see that SetMixer is permutation invariant. □

E.4 Proof of Theorem 4

Theorem 4. *The set-based anonymization method is more expressive than any existing anonymization strategies on the CE of the hypergraph. More precisely, there exists a pair of hypergraphs $\mathcal{G}_1 = (\mathcal{V}_1, \mathcal{E}_1)$ and $\mathcal{G}_2 = (\mathcal{V}_2, \mathcal{E}_2)$ with different structures (i.e., $\mathcal{G}_1 \not\cong \mathcal{G}_2$) that are distinguishable by our anonymization process and are not distinguishable by the CE-based methods.*

Proof. To the best of our knowledge, there exist two anonymization processes for random walks by Wang et al. [32] and Micali and Zhu [44]. Both of these methods are designed for graphs and to adapt them to hypergraphs we need to apply them to the (weighted) CE. Here, we focus on the process designed by Wang et al. [32], which is more informative than the other. The proof for the Micali and Zhu [44] process is the same. Note that the goal of this theorem is to investigate whether a method can distinguish a hypergraph from its CE. Accordingly, this theorem does not provide any information about the expressivity of these methods in terms of the isomorphism test.

The proposed 2-step anonymization process can be seen as a positional encoding for both vertices and hyperedges. Accordingly, it is expected to assign different positional encodings to vertices and hyperedges of two non-isomorphism hypergraphs. To this end, we construct the same hypergraphs as in the proof of Theorem 1. Let M be the number of sampled SetWalks with length L. We construct a temporal hypergraph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ as a hypergraph with $\mathcal{V} = \{u_1, u_2, \dots, u_{M(L+1)+1}\}$ and $\mathcal{E} = \{(e, t_i)\}_{i=1}^{L}$, where $e = \{u_1, u_2, \dots, u_{M(L+1)+1}\}$ and $t_1 < t_2 < \dots < t_L$. We further construct $\mathcal{G}' = (\mathcal{V}, \mathcal{E}')$ with the same set of vertices but with $\mathcal{E}' = \bigcup_{k=1}^{L} \{((u_i, u_j), t_k)\}_{u_i, u_j \in \mathcal{V}}$. As we have seen in Theorem 1, random walks on the CE of the hypergraph cannot distinguish these two hypergraphs. Since CAW [32] also uses simple random walks, it cannot distinguish these two hypergraphs. Accordingly, after its anonymization process, it again cannot distinguish these two hypergraphs.

The main part of the proof is to show that in our method, the assigned positional encodings are different in these hypergraphs. The first step is to assign each node a positional encoding. Masking the timestamps, there is only one SetWalk in the $\mathcal{G}$. Accordingly, the positional encodings of nodes in $\mathcal{G}$ are the same and non-zero. Given a SetWalk with length L we might see at most $L \times (d_{\max} - 1) + 1$ nodes, where $d_{\max}$ is the maximum size of hyperedges in the hypergraph. Accordingly, with M samples on $\mathcal{G}'$, which $d_{\max} = 2$, we can see at most $M \times (L+1)$ vertices. Therefore, in any case, we assign a zero vector to at least one vertex. This proves that the positional encodings by SetWalks are different in these two hypergraphs, and if the assigned hidden identities to counterpart nodes are different, clearly, feeding them to the SetMixer results in different hyperedge encodings.

Note that each SetWalk can be decomposed into a set of causal anonymous walks [32]. Accordingly, it includes the information provided by these walks, so its expressiveness is not less than the CAW method on hypergraphs, which completes the proof of the theorem. □

Although the above statement completes the proof, next we discuss that even given the same positional encodings for vertices in these two hypergraphs, SetMixer can capture higher-order interactions by capturing the size of the hyperedge. Recall token mixer phase in SetMixer:

$$\mathbf{H}_{\text{token}} = \mathbf{V} + \sigma\left(\texttt{Softmax}\left(\texttt{LayerNorm}(\mathbf{V})^T\right)\right)^T,$$

where $\mathbf{V} = [\mathbf{v}_1, \dots, \mathbf{v}_{M(L+1)+1}]^T \in \mathbb{R}^{(M(L+1)+1)\times d_1}$ and $\mathbf{v}_i \neq 0_{1\times d_1}$ represents the positional encoding of u_i in $\mathcal{G}$. We assumed that the positional encoding of u_i in $\mathcal{G}'$ is the same. The input of the token mixer phase on $\mathcal{G}$ is $\mathcal{V}$ as all of them are connected by a hyperedge. Then we have:

$$(\mathbf{H}_{\text{token}})_{i,j} = \mathbf{v}_{i,j} + \sigma\left(\frac{\exp(\mathbf{v}_{i,j})}{\sum_{k=1}^{M(L+1)+1}\exp(\mathbf{v}_{k,j})}\right). \tag{16}$$

On the other hand, when applied to hypergraph $\mathcal{G}'$ and (u_{k_1}, u_{k_2}). We have:

$$(\mathbf{H}'_{\text{token}})_{i,j} = \mathbf{v}_{i,j} + \sigma\left(\frac{\exp(\mathbf{v}_{i,j})}{\exp(\mathbf{v}_{k_1,j}) + \exp(\mathbf{v}_{k_2,j})}\right), \quad i \in \{k_1, k_2\}. \tag{17}$$

Since we use zero padding, for any $i \geq 3$, $(\mathbf{H}_{\text{token}})_{i,j} \neq 0$ and $(\mathbf{H}'_{\text{token}})_{i,j} = 0$. These zero rows, which capture the size of the hyperedge, result in different encodings for each connection.

Remark 5. *To the best of our knowledge, the only anonymization process that is used on hypergraphs is by Liu et al. [74], which uses simple walks on the CE and is the same as Wang et al. [32]. Accordingly, it also suffers from the above limitation. Also, note that this theorem shows the limitation of these anonymization procedures when simply adopted to hypergraphs.*

E.5 Proof of Proposition 2

Proposition 2. *Edge-independent random walks on hypergraphs [36], edge-dependent random walks on hypergraphs [39], and simple random walks on the CE of hypergraphs are all special cases of r-SetWalk, when applied to the 2-projected graph, UP graph, and 2-projected graph, respectively. Furthermore, all the above methods are less expressive than r-SetWalks.*

Proof. For the first part, we discuss each walk separately:

① Simple random walks on the CE of the hypergraphs: We perform 2-SetWalks on the (weighted) 2-projected hypergraph with $\Gamma(.) = 1$. Accordingly, for every two adjacent edges in the 2-Projected graph like e and e', we have $\varphi(e, e') = 1$. Therefore, it is equivalent to a simple random walk on the CE (2-projected graph).

② Edge-independent random walks on hypergraphs: As is shown by Chitra and Raphael [39], each edge-independent random walk on hypergraphs is equivalent to a simple random walk on the (weighted) CE of the hypergraph. Therefore, as discussed in ①, these walks are a special case of r-SetWalks, when $r = 2$ and applied to (weighted) 2-Projected hypergraph.

③ Edge-dependent random walks on hypergraphs: Let $\Gamma'(e, u)$ be an edge-dependent weight function used in the hypergraph random walk sampling. For each node u in the UP hypergraph, we store the set of $\Gamma'(e, u)$ that e is a maximal hyperedge that u belongs to. Note that there might be several maximal hyperedges that u belongs to. Now, we perform 2-SetWalk sampling on the UP hypergraph with these weights and in each step, we sample each hyperedge with weight $\Gamma(u, e) = \Gamma'(e, u)$. It is straightforward to show that given this procedure, the sampling probability of a hyperedge is the same in both cases. Therefore, edge-dependent random walks on hypergraphs are equivalent to 2-SetWalks when applied to the UP hypergraph.

As discussed above, all these walks are special cases of r-SetWalks and cannot be more expressive than r-SetWalks. Also, as discussed in Theorem 1, all these walks are less expressive than SetWalks, which are also special cases of r-SetWalks, when $r = \infty$. Accordingly, all these methods are less expressive than r-SetWalks. □

F Experimental Setup Details

F.1 Datasets

We use 10 publicly available[1] benchmark datasets, whose descriptions are as follows:

- **NDC Class** [5]: The NDC Class dataset is a temporal higher-order network, in which each hyperedge corresponds to an individual drug, and the nodes contained within the hyperedges represent class labels assigned to these drugs. The timestamps, measured in days, indicate the initial market entry of each drug. Here, hyperedge prediction aims to predict future drugs.
- **NDC Substances** [5]: The NDC Substances is a temporal higher-order network, where each hyperedge represents an NDC code associated with a specific drug, while the nodes represent the constituent substances of the drug. The timestamps, measured in days, indicate the initial market entry of each drug. The hyperedge prediction task is the same as NDC Classes dataset.
- **High School** [5, 88]: The High School is a temporal higher-order network dataset constructed from interactions recorded by wearable sensors in a high school setting. The dataset captures a high school contact network, where each student/teacher is represented as a node and each hyperedge shows Face-to-face contact among individuals. Interactions were recorded at a resolution of 20 seconds, capturing all interactions that occurred within the previous 20 seconds. Node labels in this data are the class of students, and we focus on the node class "PSI" in our classification tasks.
- **Primary School** [5, 89]: The primary school dataset resembles the high school dataset, differing only in terms of the school level from which the data is collected. Node labels in this data are the class of students, and we focus on the node class "Teachers" in our classification tasks.
- **Congress Bill** [5, 90, 91]: Each node in this dataset represents a US Congressperson. Each hyperedge is a legislative bill in both the House of Representatives and the Senate, connecting the sponsors and co-sponsors of each respective bill. The timestamps, measured in days, indicate the date when each bill was introduced.
- **Email Enron** [5]: In this dataset nodes are email addresses at Enron and hyperedges are formed by emails, connecting the sender and recipients of each email. The timestamps have a resolution of milliseconds.
- **Email Eu** [5, 92]: In this dataset, the nodes represent email addresses associated with a European research institution. Each hyperedge consists of the sender and all recipients of the email. The timestamps in this dataset are measured with a resolution of 1 second.
- **Question Tags M (Math sx)** [5]: This dataset consists of nodes representing tags and hyperedges representing sets of tags applied to questions on math.stackexchange.com. The timestamps in the dataset are recorded at millisecond resolution and have been normalized to start at 0.
- **Question Tags U (Ask Ubuntu)** [5]: In this dataset, the nodes represent tags, and the hyperedges represent sets of tags applied to questions on askubuntu.com. The timestamps in the dataset are recorded with millisecond resolution and have been normalized to start at 0.
- **Users-Threads** [5]: In this dataset, the nodes represent users on askubuntu.com, and a hyperedge is formed by users participating in a thread that lasts for a maximum duration of 24 hours. The timestamps in the dataset denote the time of each post, measured in milliseconds but normalized such that the earliest post begins at 0.

The statistics of these datasets can be found in Table 3.

F.2 Baselines

We compare our method to eight previous state-of-the-art methods and baselines on the hyperedge prediction task:

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

Table 3: Dataset statistics. HeP: **H**yp**e**redge **P**rediction, NC: **N**ode **C**lassification

Dataset	NDC Class	High School	Primary School	Congress Bill	Email Enron	Email Eu	Question Tags M	Users-Threads	NDC Substances	Question Tags U
$\lvert\mathcal{V}\rvert$	1,161	327	242	1,718	143	998	1,629	125,602	5,311	3,029
$\lvert\mathcal{E}\rvert$	49,724	172,035	106,879	260,851	10,883	234,760	822,059	192,947	112,405	271,233
#Timestamps	5,891	7,375	3,100	5,936	10,788	232,816	822,054	189,917	7,734	271,233
Task	HeP	HeP& NC	HeP & NC	HeP	HeP	HeP	HeP	HeP	HeP	HeP

- CHESHIRE [10]: Chebyshev spectral hyperlink predictor (CHESHIRE), is a hyperedge prediction methods that initializes node embeddings by directly passing the incidence matrix through a one-layer neural network. CHESHIRE treats a hyperedge as a fully connected graph (clique) and uses a Chebyshev spectral GCN to refine the embeddings of the nodes within the hyperedge. The Chebyshev spectral GCN leverages Chebyshev polynomial expansion and spectral graph theory to learn localized spectral filters. These filters enable the extraction of local and composite features from graphs that capture complex geometric structures. The model with code provided is here.
- HyperSAGCN [25]: Self-attention-based graph convolutional network for hypergraphs (HyperSAGCN) utilizes a Spectral Aggregated Graph Convolutional Network (SAGCN) to refine the embeddings of nodes within each hyperedge. HyperSAGCN generates initial node embeddings by hypergraph random walks and combines node embeddings by Mean (.) pooling to compute the embedding of hyperedge. The model with code provided is here.
- NHP [83]: Neural Hyperlink Predictor (NHP), is an enhanced version of HyperSAGCN. NHP initializes node embeddings using Node2Vec on the CE graph and then uses a novel maximum minimum-based pooling function that enables adaptive weight learning in a task-specific manner, incorporating additional prior knowledge about the nodes. The model with code provided is here.
- HPLSF [85]: Hyperlink Prediction using Latent Social Features (HPLSF) is a probabilistic method. It leverages the homophily property of the networks and introduces a latent feature learning approach, incorporating the use of entropy in computing hyperedge embedding. The model with code provided is here.
- HPRA [84]: Hyperlink Prediction Using Resource Allocation (HPRA) is a hyperedge prediction method based on the resource allocation process. HPRA calculates a hypergraph resource allocation (HRA) index between two nodes, taking into account direct connections and shared neighbors. The HRA index of a candidate hyperedge is determined by averaging all pairwise HRA indices between the nodes within the hyperedge. The model with code provided is here.
- CE-CAW: This model is a baseline that we apply CAW [32] on the CE of the hypergraph. CAW is a temporal edge prediction method that uses causal anonymous random walks to capture the dynamic laws of the network in an inductive manner. The model with code provided is here.
- CE-EvolveGCN: This is a snapshot-based temporal graph learning method that we apply EvolveGCN [86], which uses Rnns to estimate the GCN parameters for the future snapshots, on the CE of the hypergraph. The model with code provided is here.
- CE-GCN: We apply Graph Convolutional Networks [87] to the CE of the hypergraph to obtain node embeddings. Next, we use MLP to predict edges. The implementation is provided in the Pytorch Geometric library.

For node classification, we use four state-of-the-art deep hypergraph learning methods and a CE-based baseline:

- HyperGCN [18]: This is a generalization of GCNs to hypergraphs, where it uses hypergraph Laplacian to define convolution.
- AllDeepSets and AllSetTransformer [24]: These two methods are two variants of the general message passing framework, Allset, on hypergraphs, which are based on the aggregation of messages from nodes to hyperedges and from hyperedges to nodes.
- UniGCNII [27]: Is an advanced variant of UniGnn, a general framework for message passing on hypergraphs.

Table 4: Hyperparameters used in the grid search.

Datasets	Sampling Number M	Sampling Time Bias α	SetWalk Length m	Hidden dimensions
NDC Class	4, 8, 16, 32, 64, 128	$\{0.5, 2.0, 20, 200\} \times 10^{-7}$	2, 3, 4, 5	32, 64, 128
High School	4, 8, 16, 32, 64, 128	$\{0.5, 2.0, 20, 200\} \times 10^{-7}$	2, 3, 4, 5	32, 64, 128
Primary School	4, 8, 16, 32, 64, 128	$\{0.5, 2.0, 20, 200\} \times 10^{-7}$	2, 3, 4, 5	32, 64, 128
Congress Bill	8, 16, 32, 64	$\{0.5, 2.0, 20, 200\} \times 10^{-7}$	2, 3, 4, 5	32, 64, 128
Email Enron	8, 16, 32, 64	$\{0.5, 2.0, 20, 200\} \times 10^{-7}$	2, 3, 4, 5	32, 64, 128
Email Eu	8, 16, 32, 64	$\{0.5, 2.0, 20, 200\} \times 10^{-7}$	2, 3, 4	32, 64, 128
Question Tags M	8, 16, 32, 64	$\{0.5, 2.0, 20, 200\} \times 10^{-7}$	2, 3, 4	32, 64, 128
Users-Threads	8, 16, 32, 64	$\{0.5, 2.0, 20, 200\} \times 10^{-7}$	2, 3, 4	32, 64, 128
NDC Substances	8, 16, 32, 64	$\{0.5, 2.0, 20, 200\} \times 10^{-7}$	2, 3, 4	32, 64, 128
Question Tags U	8, 16, 32, 64	$\{0.5, 2.0, 20, 200\} \times 10^{-7}$	2, 3, 4	32, 64, 128

- CE-GCN: We apply Graph Convolutional Networks [87] to the CE of the hypergraph to obtain node embeddings. Next, we use MLP to predict the labels of nodes. The implementation is provided in the Pytorch Geometric library. [87]

For all the baselines, we set all sensitive hyperparameters (e.g., learning rate, dropout rate, batch size, etc.) to the values given in the paper that describes the technique. Following [57], for deep learning methods, we tune their hidden dimensions via grid search to be consistent with what we did for CAt-Walk. We exclude HPLSF [85] and HPRA [84] from inductive hyperedge prediction as it does not apply to them.

F.3 Implementation and Training Details

In addition to hyperparameters and modules (activation functions) mentioned in the main paper, here, we report the training hyperparameters of CAt-Walk: On all datasets, we use a batch size of 64 and set learning rate = 10^{-4}. We also use an early stopping strategy to stop training if the validation performance does not increase for more than 5 epochs. We use the maximum training epoch number of 30 and dropout layers with rate = 0.1. Other hyperparameters used in the implementation can be found in the `README` file in the supplement.

Also, for tuning the model's hyperparameters, we systematically tune them using grid search. The search domains of each hyperparameter are reported in Table 4. Note that, the last column in Table 4 reports the search domain for hidden dimensions of modules in CAt-Walk, including SetMixer, MLP-Mixer, and MLPs. Also, we tune the last layer pooling strategy with two options: SetMixer or Mean(.) whichever leads to a better performance.

We implemented our method in Python 3.7 with *PyTorch* and run the experiments on a Linux machine with *nvidia RTX A4000* GPU with 16GB of RAM.

G Additional Experimental Results

G.1 Results on More Datasets

Due to the space limit, we report the AUC results on only eight datasets in Section 4. Table 6 reports both AUC and average precision (AP) results on all 10 datasets in both inductive and transductive hyperedge prediction tasks.

G.2 Node Classification

In the main text, we focus on the hyperedge prediction task. Here we describe how CAt-Walk can be used for node classification tasks.

For each node u_0 in the training set, we sample $\max\{\deg(u_0), 10\}$ hyperedges such as $e_0 = \{u_0, u_1, \dots, u_k\}$. Next, for each sampled hyperedge we sample M SetWalks with length m starting from each $u_i \in e_0$ to construct $\mathcal{S}(u_i)$. Next, we anonymize each hyperedge that appears in at least one SetWalk in $\bigcup_{i=0}^{k} \mathcal{S}(u_i)$ by Equation 3 and then use the MLP-Mixer module to encode each $\text{Sw} \in \bigcup_{i=0}^{k} \mathcal{S}(u_i)$. To encode each node $u_i \in e_0$, we use Mean(.) pooling over SetWalks in $\mathcal{S}(u_i)$.

Table 5: Performance on node classification: Mean ACC (%) ± standard deviation. Boldfaced letters shaded blue indicate the best result, while gray shaded boxes indicate results within one standard deviation of the best result.

	Methods	High School	Primary School	Average Performance
Inductive	CE-GCN	76.24 ± 2.99	79.03 ± 3.16	77.63 ± 3.07
	HyperGCN	83.91 ± 3.05	86.17 ± 3.40	85.04 ± 3.23
	AllDeepSets	85.67 ± 4.17	81.43 ± 6.77	83.55 ± 5.47
	UniGCNII	88.36 ± 3.78	88.27 ± 3.52	88.31 ± 3.63
	AllSetTransformer	**91.19 ± 2.85**	90.00 ± 4.35	90.59 ± 3.6
	CAt-Walk	88.99 ± 4.76	**93.28 ± 2.41**	**91.13 ± 3.58**
Transductive	CE-GCN	78.93 ± 3.11	77.46 ± 2.97	78.20 ± 3.04
	HyperGCN	84.90 ± 3.59	85.23 ± 3.06	85.07 ± 3.33
	AllDeepSets	85.97 ± 4.05	80.20 ± 10.18	83.09 ± 7.12
	UniGCNII	89.16 ± 4.37	90.29 ± 4.01	89.73 ± 4.19
	AllSetTransformer	**90.75 ± 3.13**	89.80 ± 2.55	90.27 ± 2.84
	CAt-Walk	90.66 ± 4.96	**93.20 ± 2.45**	**91.93 ± 3.71**

Finally, for node classification task, we use a 2-layer perceptron over the node encodings to make the final prediction.

Table 5 reports the results of dynamic node classification tasks on High School and Primary School datasets. CAt-Walk achieves the best or on-par performance on dynamic node classification tasks. While all baselines are specifically designed for node classification tasks, CAt-Walk achieves superior results due to ① its ability to incorporate temporal properties (both from SetWalks and our time encoding module), which helps to learn underlying dynamic laws of the network, and ② its two-step set-based anonymization process that hides node identities from the model. Accordingly, CAt-Walk can learn underlying patterns needed for the node classification task, instead of using node identities, which might cause memorizing vertices.

G.3 Performance in Average Precision

In addition to the AUC, we also compare our model with baselines with respect to Average Precision (AP). Table 6 reports both AUC and AP results on all 10 datasets in inductive and transductive hyperedge prediction tasks. As discussed in Section 4, CAt-Walk due to its ability to capture both temporal and higher-order properties of the hypergraphs, achieves superior performance and outperforms all baselines in both transductive and inductive settings with a significant margin.

G.4 Scalability Analysis

In this part, we investigate the scalability of CAt-Walk. To this end, we use different versions of the High School dataset with different numbers of hyperedges from 10^4 to 1.6×10^5. Figure 6 (left) reports the runtimes of SetWalk sampling and Figure 6 (right) reports the runtimes of CAt-Walk for training one epoch using $M = 8$, $m = 3$ with batch-size = 64. Interestingly, our method scales linearly with the number of hyperedges, which enables it to be used on long hyperedge streams and large hypergraphs.

G.5 More Results on Rnn v.s. MLP-Mixer in Walk Encoding

Most existing methods on (temporal) random walk encoding see a walk as a sequence of vertices and uses sequence encoders like Rnns or Transformers to encode each walk. The main drawback of these methods is that they fail to directly process temporal walks with irregular gaps between timestamps. That is, sequential encoders can be seen as discrete approximations of dynamic systems; however, this discretization often fails if we have irregularly observed data [118]. This is the main motivation of recent studies to develop methods on continuous-time temporal networks [33, 119]. Most of these methods are too complicated and sometimes fail to generalize [120]. In CAt-Walk, we suggest a simple architecture to encode temporal walks by a time-encoding module along with a Mixer module (see Section 3.4 for the details). In this part, we evaluate the power of our Mixer module

Table 6: Performance on hyperedge prediction: AUC and Average Precision (%) ± standard deviation. Boldfaced letters shaded blue indicate the best result, while gray shaded boxes indicate results within one standard deviation of the best result. N/A: the method has computational issues.

	Datasets	NDC Class		High School		Primary School		Congress Bill		Email Enron	
	Metric	AUC	AP	AUC	AP	AUC	AP	AUC	AP	AUC	AP
Inductive	Strongly Inductive										
	CE-GCN	52.31 ± 2.99	54.33 ± 2.48	60.54 ± 2.06	59.92 ± 2.25	52.34 ± 2.75	56.41 ± 2.06	49.18 ± 3.61	53.85 ± 3.92	63.04 ± 1.80	57.70 ± 2.27
	CE-EvolveGCN	49.78 ± 3.13	55.24 ± 3.56	46.12 ± 3.83	52.87 ± 3.48	58.01 ± 2.56	55.68 ± 2.41	54.00 ± 1.84	50.27 ± 1.76	57.31 ± 4.19	54.52 ± 3.79
	CE-CAW	76.45 ± 0.29	78.58 ± 1.32	83.73 ± 1.42	82.96 ± 1.04	80.31 ± 1.46	82.84 ± 1.71	75.38 ± 1.25	77.19 ± 1.38	70.81 ± 1.13	72.07 ± 1.52
	NHP	70.53 ± 4.95	68.18 ± 4.31	65.29 ± 3.80	62.86 ± 3.74	70.86 ± 3.42	71.31 ± 3.51	69.82 ± 2.19	64.09 ± 2.87	49.71 ± 6.09	50.01 ± 4.87
	HyperSAGCN	79.05 ± 2.48	77.24 ± 2.05	88.12 ± 3.01	82.72 ± 2.93	80.13 ± 1.38	76.32 ± 2.96	79.51 ± 1.27	80.58 ± 2.61	73.09 ± 2.60	72.29 ± 3.69
	CHESHIRE	72.24 ± 2.63	70.31 ± 2.26	82.54 ± 0.88	80.34 ± 1.19	77.26 ± 1.01	77.72 ± 0.76	79.43 ± 1.58	78.63 ± 1.25	70.03 ± 2.55	72.97 ± 1.81
	CAt-Walk	**98.89 ± 1.82**	**98.97 ± 1.69**	**96.03 ± 1.50**	**96.41 ± 0.70**	**95.32 ± 0.89**	**96.03 ± 0.84**	**93.54 ± 0.56**	**93.93 ± 0.36**	**73.45 ± 2.92**	**74.66 ± 3.87**
	Weakly Inductive										
	CE-GCN	51.80 ± 3.29	50.94 ± 3.77	50.33 ± 3.40	48.54 ± 3.92	52.19 ± 2.54	53.21 ± 3.59	52.38 ± 2.75	50.81 ± 2.68	50.81 ± 2.87	55.38 ± 2.79
	CE-EvolveGCN	55.39 ± 5.16	57.24 ± 4.98	57.85 ± 3.51	63.26 ± 4.01	51.50 ± 4.07	52.59 ± 4.53	55.63 ± 3.41	5.19 ± 3.56	45.66 ± 2.10	50.93 ± 2.57
	CE-CAW	77.61 ± 1.05	80.03 ± 1.65	83.77 ± 1.41	83.41 ± 1.19	82.98 ± 1.06	80.84 ± 1.57	79.51 ± 0.94	80.39 ± 1.07	**80.54 ± 1.02**	77.41 ± 1.28
	NHP	75.17 ± 2.02	77.23 ± 3.11	67.25 ± 5.19	66.73 ± 4.94	71.92 ± 1.83	72.30 ± 1.89	69.58 ± 4.07	72.48 ± 4.83	60.38 ± 4.45	55.62 ± 4.67
	HyperSAGCN	79.45 ± 2.18	80.32 ± 2.23	88.53 ± 1.26	87.26 ± 1.49	85.08 ± 1.45	86.84 ± 1.60	80.12 ± 2.00	73.48 ± 2.77	78.86 ± 0.63	**79.14 ± 1.51**
	CHESHIRE	79.03 ± 1.24	78.98 ± 1.17	88.40 ± 1.06	86.53 ± 1.82	83.55 ± 1.27	79.42 ± 2.03	79.67 ± 0.83	80.03 ± 1.38	74.53 ± 0.91	75.88 ± 1.14
	CAt-Walk	**99.16 ± 1.08**	**99.33 ± 0.89**	**94.68 ± 2.37**	**96.54 ± 0.82**	**96.53 ± 1.39**	**96.83 ± 1.16**	**98.38 ± 0.21**	**98.48 ± 0.18**	**64.11 ± 7.96**	**67.68 ± 6.93**
Transductive	HPRA	70.83 ± 0.01	67.40 ± 0.00	94.91 ± 0.00	89.17 ± 0.00	89.86 ± 0.06	88.11 ± 0.02	79.48 ± 0.03	77.16 ± 0.03	78.62 ± 0.00	76.74 ± 0.00
	HPLSF	76.19 ± 0.82	77.62 ± 1.42	92.14 ± 0.29	92.79 ± 0.15	88.57 ± 1.09	87.69 ± 1.61	79.31 ± 0.52	75.88 ± 0.43	75.73 ± 0.05	75.32 ± 0.08
	CE-GCN	66.83 ± 3.74	65.83 ± 3.61	62.99 ± 3.02	59.76 ± 3.78	59.14 ± 3.87	55.59 ± 3.46	64.42 ± 3.11	63.19 ± 3.34	58.06 ± 3.80	55.27 ± 3.12
	CE-EvolveGCN	67.08 ± 3.51	66.51 ± 3.80	65.19 ± 2.26	59.27 ± 2.19	63.15 ± 1.32	65.18 ± 1.89	69.30 ± 2.27	64.38 ± 2.66	69.98 ± 5.38	67.76 ± 5.16
	CE-CAW	76.30 ± 0.84	77.73 ± 1.42	81.63 ± 0.97	79.37 ± 0.53	86.53 ± 084	87.03 ± 1.15	76.99 ± 1.02	77.05 ± 1.14	79.57 ± 0.14	78.37 ± 1.15
	NHP	82.39 ± 2.81	80.72 ± 2.04	76.85 ± 3.08	75.37 ± 3.12	80.04 ± 3.42	80.24 ± 3.49	80.27 ± 2.53	77.82 ± 1.91	63.17 ± 3.79	66.87 ± 3.19
	HyperSAGCN	80.76 ± 2.64	80.50 ± 2.73	94.98 ± 1.30	89.73 ± 1.21	90.77 ± 2.05	88.64 ± 2.09	82.84 ± 1.61	81.12 ± 1.79	**83.59 ± 0.98**	80.54 ± 1.66
	CHESHIRE	84.91 ± 1.05	82.24 ± 1.49	95.11 ± 0.94	94.29 ± 1.23	91.62 ± 1.18	92.72 ± 1.07	86.81 ± 1.24	83.66 ± 1.90	82.27 ± 0.86	81.39 ± 0.81
	CAt-Walk	**98.72 ± 1.38**	**98.71 ± 1.36**	**95.30 ± 0.43**	**95.90 ± 0.44**	**97.91 ± 3.30**	**97.92 ± 2.95**	**88.15 ± 1.46**	**88.66 ± 1.57**	**80.47 ± 5.30**	**82.87 ± 3.50**

	Datasets	Email Eu		Question Tags M		Users-Threads		NDC Substances		Question Tags U	
	Metric	AUC	AP	AUC	AP	AUC	AP	AUC	AP	AUC	AP
Inductive	Strongly Inductive										
	CE-GCN	52.76 ± 2.41	50.37 ± 2.59	56.10 ± 1.88	54.15 ± 1.94	57.91 ± 1.56	59.45 ± 1.21	55.70 ± 2.91	54.29 ± 2.78	51.97 ± 2.91	55.03 ± 2.72
	CE-EvolveGCN	44.16 ± 1.27	49.15 ± 1.23	64.08 ± 2.75	60.64 ± 2.78	52.00 ± 2.32	52.69 ± 2.15	58.17 ± 2.24	57.35 ± 2.13	54.57 ± 2.25	57.16 ± 2.55
	CE-CAW	72.99 ± 0.20	73.45 ± 0.68	70.14 ± 1.89	70.26 ± 1.77	73.12 ± 1.06	72.64 ± 1.18	75.87 ± 0.77	73.19 ± 0.86	74.21 ± 2.04	76.52 ± 2.06
	NHP	65.35 ± 2.07	64.24 ± 1.61	68.23 ± 3.34	69.82 ± 3.41	71.83 ± 2.64	71.09 ± 2.83	70.43 ± 3.64	73.22 ± 3.03	72.52 ± 2.90	71.56 ± 2.26
	HyperSAGCN	78.01 ± 1.24	80.04 ± 1.87	73.66 ± 1.95	73.98 ± 1.35	73.94 ± 2.57	72.97 ± 2.45	75.85 ± 2.21	73.24 ± 2.75	78.88 ± 2.69	77.53 ± 2.28
	CHESHIRE	69.98 ± 2.71	70.10 ± 3.05	*N/A*	*N/A*	76.99 ± 2.82	74.03 ± 2.78	76.60 ± 2.19	74.91 ± 2.71	75.04 ± 3.39	75.46 ± 2.90
	CAt-Walk	**91.68 ± 2.78**	**91.75 ± 2.82**	**88.03 ± 3.38**	**88.46 ± 3.09**	**89.84 ± 6.02**	**91.58 ± 4.37**	**93.29 ± 1.55**	**94.26 ± 1.21**	**97.59 ± 2.21**	**97.71 ± 2.07**
	Weakly Inductive										
	CE-GCN	49.60 ± 3.96	55.01 ± 3.25	55.13 ± 2.76	51.48 ± 2.66	57.06 ± 3.16	58.37 ± 2.86	60.92 ± 2.81	55.93 ± 2.03	56.85 ± 2.73	57.19 ± 2.52
	CE-EvolveGCN	52.44 ± 2.38	50.61 ± 2.32	61.79 ± 1.63	59.61 ± 1.12	55.81 ± 2.54	50.63 ± 2.46	58.48 ± 2.49	55.90 ± 2.51	54.10 ± 1.21	56.13 ± 2.32
	CE-CAW	73.54 ± 1.19	74.10 ± 1.41	77.29 ± 0.86	77.67 ± 1.94	80.79 ± 0.82	81.88 ± 0.63	77.28 ± 1.30	79.24 ± 1.19	76.51 ± 1.26	77.17 ± 1.39
	NHP	67.19 ± 4.33	66.53 ± 4.21	70.46 ± 3.52	65.66 ± 3.94	76.44 ± 1.90	75.23 ± 3.96	73.37 ± 3.51	70.62 ± 3.71	78.15 ± 4.41	79.64 ± 4.32
	HyperSAGCN	77.26 ± 2.09	74.05 ± 2.12	78.15 ± 1.41	76.19 ± 1.53	75.38 ± 1.43	70.35 ± 1.63	80.82 ± 2.18	76.67 ± 2.06	74.22 ± 1.91	70.57 ± 1.02
	CHESHIRE	77.31 ± 0.95	76.01 ± 0.98	*N/A*	*N/A*	81.27 ± 0.85	82.96 ± 1.41	80.68 ± 1.31	80.78 ± 1.13	77.60 ± 1.57	79.48 ± 1.79
	CAt-Walk	**91.98 ± 2.41**	**92.22 ± 2.40**	**90.28 ± 2.81**	**90.56 ± 2.62**	**97.15 ± 1.81**	**97.55 ± 1.49**	**95.65 ± 1.82**	**96.18 ± 1.52**	**98.11 ± 1.31**	**98.25 ± 1.13**
Transductive	HPRA	72.51 ± 0.00	71.08 ± 0.00	83.18 ± 0.00	80.12 ± 0.00	70.49 ± 0.02	72.83 ± 0.00	77.94 ± 0.01	75.78 ± 0.01	81.05 ± 0.00	81.71 ± 0.00
	HPLSF	75.27 ± 0.31	77.95 ± 0.14	83.45 ± 0.93	82.29 ± 1.06	74.38 ± 1.11	73.81 ± 1.45	82.12 ± 0.71	84.51 ± 0.62	80.89 ± 1.51	75.62 ± 1.38
	CE-GCN	64.19 ± 2.79	65.93 ± 2.52	55.18 ± 5.12	55.84 ± 4.53	62.78 ± 2.69	59.71 ± 2.25	63.08 ± 2.19	65.37 ± 2.48	66.79 ± 2.88	60.51 ± 2.26
	CE-EvolveGCN	64.36 ± 4.17	66.98 ± 3.72	72.56 ± 1.72	69.38 ± 1.51	68.55 ± 2.26	67.86 ± 2.61	70.09 ± 3.42	66.37 ± 3.17	71.31 ± 2.92	70.36 ± 2.72
	CE-CAW	78.19 ± 1.10	77.95 ± 0.98	81.73 ± 2.48	83.27 ± 2.34	80.86 ± 0.45	80.57 ± 1.08	84.72 ± 1.65	84.93 ± 1.26	80.37 ± 1.77	83.14 ± 0.97
	NHP	78.90 ± 4.39	76.95 ± 5.08	79.14 ± 3.36	78.79 ± 3.15	82.33 ± 1.02	81.44 ± 1.53	81.38 ± 1.42	82.17 ± 1.38	78.99 ± 4.16	80.06 ± 4.33
	HyperSAGCN	79.61 ± 2.35	75.99 ± 2.23	84.07 ± 2.50	84.22 ± 2.43	79.62 ± 2.04	79.38 ± 2.55	85.07 ± 2.46	85.32 ± 2.20	85.18 ± 2.64	80.99 ± 3.04
	CHESHIRE	86.38 ± 1.23	87.39 ± 1.07	*N/A*	*N/A*	82.75 ± 1.99	81.96 ± 1.75	86.30 ± 1.57	83.18 ± 1.92	87.83 ± 2.15	88.62 ± 1.76
	CAt-Walk	**96.74 ± 1.28**	**97.08 ± 1.20**	**91.63 ± 1.41**	**92.28 ± 1.26**	**93.51 ± 1.27**	**94.98 ± 0.98**	**90.64 ± 0.44**	**91.96 ± 0.41**	**96.59 ± 4.39**	**97.06 ± 3.72**

and compare its performance when we replace it with Rnns [78]. Figure 7 reports the results on all datasets. We observe that using MLP-Mixer with the time-encoding module in CAt-Walk can always outperform CAt-Walk when we replace MLP-Mixer with a Rnn, and mostly this improvement is more on datasets with high variance in their timestamps. We relate this superiority to the importance of using continuous-time encoding instead of sequential encoders.

H Broader Impacts

Temporal hypergraph learning methods, such as CAt-Walk, benefit a wide array of real-world applications, including but not limited to social network analysis, recommender systems, brain network analysis, drug discovery, stock price prediction, and anomaly detection (e.g. bot detection in social media or abnormal human brain activity). However, there might be some potentially negative impacts, which we list as: ① Learning underlying biased patterns in the training data, which may result in stereotyped predictions. Since CAt-Walk learns underlying dynamic laws in the training data, given biased training data, the predictions of CAt-Walk can be biased. ② Also, powerful dynamic hypergraph models can be used for manipulation in the abovementioned applications (e.g., stock price

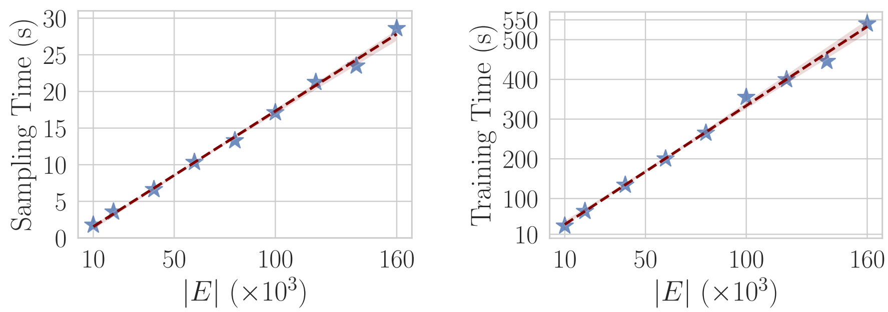

Figure 6: Scalability evaluation: The runtime of (**left**) SetWalk extraction and (**right**) the training time of CAt-Walk over one epoch on High School (using different $|E|$ for training).

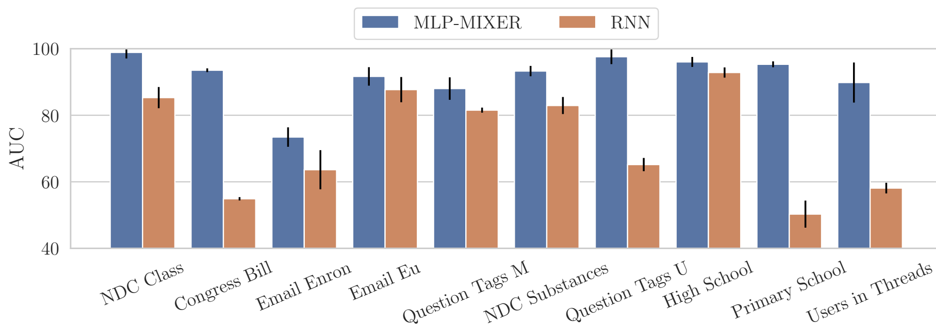

Figure 7: The importance of using MLP-Mixer in CAt-Walk. Using an Rnn instead of MLP-Mixer can damage the performance in *all* datasets. Rnns are sequential encoders and are not able to encode continuous time in the data.

manipulation). Accordingly, to prevent the potential risks in sensitive tasks, e.g., decision-making from graph-structured data in health care, interpretability and explainability of machine learning models on hypergraphs is a critical area for future work.

Furthermore, this work does not perform research on human subjects as part of the study and all used datasets are anonymized and publicly available.

I Reproducibility

All codes and implementations are available in the supplement.

References

[1] Serina Chang, Emma Pierson, Pang Wei Koh, Jaline Gerardin, Beth Redbird, David Grusky, and Jure Leskovec. Mobility network models of covid-19 explain inequities and inform reopening. *Nature*, 589(7840):82–87, 2021.

[2] Martino Ciaperoni, Edoardo Galimberti, Francesco Bonchi, Ciro Cattuto, Francesco Gullo, and Alain Barrat. Relevance of temporal cores for epidemic spread in temporal networks. *Scientific reports*, 10(1):1–15, 2020.

[3] Joakim Skarding, Bogdan Gabrys, and Katarzyna Musial. Foundations and modeling of dynamic networks using dynamic graph neural networks: A survey. *IEEE Access*, 9:79143–79168, 2021.

[4] Antonio Longa, Veronica Lachi, Gabriele Santin, Monica Bianchini, Bruno Lepri, Pietro Lio, Franco Scarselli, and Andrea Passerini. Graph neural networks for temporal graphs: State of the art, open challenges, and opportunities. *arXiv preprint arXiv:2302.01018*, 2023.

[5] 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. ISSN 0027-8424. doi: 10.1073/pnas.1800683115.

[6] Federico Battiston, Enrico Amico, Alain Barrat, Ginestra Bianconi, Guilherme Ferraz de Arruda, Benedetta Franceschiello, Iacopo Iacopini, Sonia Kéfi, Vito Latora, Yamir Moreno, Micah M. Murray, Tiago P. Peixoto, Francesco Vaccarino, and Giovanni Petri. The physics of higher-order interactions in complex systems. *Nature Physics*, 17(10):1093–1098, Oct 2021. ISSN 1745-2481. doi: 10.1038/s41567-021-01371-4. URL https://doi.org/10.1038/s41567-021-01371-4.

[7] Yuanzhao Zhang, Maxime Lucas, and Federico Battiston. Higher-order interactions shape collective dynamics differently in hypergraphs and simplicial complexes. *Nature Communications*, 14(1):1605, Mar 2023. ISSN 2041-1723. doi: 10.1038/s41467-023-37190-9. URL https://doi.org/10.1038/s41467-023-37190-9.

[8] Eun-Sol Kim, Woo Young Kang, Kyoung-Woon On, Yu-Jung Heo, and Byoung-Tak Zhang. Hypergraph attention networks for multimodal learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 14581–14590, 2020.

[9] Yichao Yan, Jie Qin, Jiaxin Chen, Li Liu, Fan Zhu, Ying Tai, and Ling Shao. Learning multi-granular hypergraphs for video-based person re-identification. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2899–2908, 2020.

[10] Can Chen, Chen Liao, and Yang-Yu Liu. Teasing out missing reactions in genome-scale metabolic networks through hypergraph learning. *Nature Communications*, 14(1):2375, 2023.

[11] Guobo Xie, Yinting Zhu, Zhiyi Lin, Yuping Sun, Guosheng Gu, Jianming Li, and Weiming Wang. Hbrwrlda: predicting potential lncrna–disease associations based on hypergraph bi-random walk with restart. *Molecular Genetics and Genomics*, 297(5):1215–1228, 2022.

[12] Junliang Yu, Hongzhi Yin, Jundong Li, Qinyong Wang, Nguyen Quoc Viet Hung, and Xiangliang Zhang. Self-supervised multi-channel hypergraph convolutional network for social recommendation. In *Proceedings of the web conference 2021*, pages 413–424, 2021.

[13] Dingqi Yang, Bingqing Qu, Jie Yang, and Philippe Cudre-Mauroux. Revisiting user mobility and social relationships in lbsns: a hypergraph embedding approach. In *The world wide web conference*, pages 2147–2157, 2019.

[14] Junren Pan, Baiying Lei, Yanyan Shen, Yong Liu, Zhiguang Feng, and Shuqiang Wang. Characterization multimodal connectivity of brain network by hypergraph gan for alzheimer's disease analysis. In *Pattern Recognition and Computer Vision: 4th Chinese Conference, PRCV 2021, Beijing, China, October 29–November 1, 2021, Proceedings, Part III 4*, pages 467–478. Springer, 2021.

[15] Li Xiao, Junqi Wang, Peyman H Kassani, Yipu Zhang, Yuntong Bai, Julia M Stephen, Tony W Wilson, Vince D Calhoun, and Yu-Ping Wang. Multi-hypergraph learning-based brain functional connectivity analysis in fmri data. *IEEE transactions on medical imaging*, 39(5): 1746–1758, 2019.

[16] Yifan Feng, Haoxuan You, Zizhao Zhang, Rongrong Ji, and Yue Gao. Hypergraph neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 3558–3565, 2019.

[17] Dengyong Zhou, Jiayuan Huang, and Bernhard Schölkopf. Learning with hypergraphs: Clustering, classification, and embedding. In B. Schölkopf, J. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems*, volume 19. MIT Press, 2006. URL https://proceedings.neurips.cc/paper_files/paper/2006/file/dff8e9c2ac33381546d96deea9922999-Paper.pdf.

[18] Naganand Yadati, Madhav Nimishakavi, Prateek Yadav, Vikram Nitin, Anand Louis, and Partha Talukdar. Hypergcn: A new method for training graph convolutional networks on hypergraphs. *Advances in neural information processing systems*, 32, 2019.

[19] Sameer Agarwal, Jongwoo Lim, Lihi Zelnik-Manor, Pietro Perona, David Kriegman, and Serge Belongie. Beyond pairwise clustering. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 2, pages 838–845. IEEE, 2005.

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

[21] Matthias Hein, Simon Setzer, Leonardo Jost, and Syama Sundar Rangapuram. The total variation on hypergraphs - learning on hypergraphs revisited. In C.J. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc., 2013. URL https://proceedings.neurips.cc/paper_files/paper/2013/file/8a3363abe792db2d8761d6403605aeb7-Paper.pdf.

[22] Pan Li and Olgica Milenkovic. Submodular hypergraphs: p-laplacians, Cheeger inequalities and spectral clustering. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 3014–3023. PMLR, 10–15 Jul 2018. URL https://proceedings.mlr.press/v80/li18e.html.

[23] I (Eli) Chien, Huozhi Zhou, and Pan Li. hs^2: Active learning over hypergraphs with pointwise and pairwise queries. In Kamalika Chaudhuri and Masashi Sugiyama, editors, *Proceedings of the Twenty-Second International Conference on Artificial Intelligence and Statistics*, volume 89 of *Proceedings of Machine Learning Research*, pages 2466–2475. PMLR, 16–18 Apr 2019. URL https://proceedings.mlr.press/v89/chien19a.html.

[24] Eli Chien, Chao Pan, Jianhao Peng, and Olgica Milenkovic. You are allset: A multiset function framework for hypergraph neural networks. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=hpBTIv2uy_E.

[25] Ruochi Zhang, Yuesong Zou, and Jian Ma. Hyper-sagnn: a self-attention based graph neural network for hypergraphs. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=ryeHuJBtPH.

[26] Yuan Luo. SHINE: Subhypergraph inductive neural network. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=IsHRUzXPqhI.

[27] Jing Huang and Jie Yang. Unignn: a unified framework for graph and hypergraph neural networks. In Zhi-Hua Zhou, editor, *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, pages 2563–2569. International Joint Conferences on Artificial Intelligence Organization, 8 2021. doi: 10.24963/ijcai.2021/353. URL https://doi.org/10.24963/ijcai.2021/353. Main Track.

[28] Ghadeer AbuOda, Gianmarco De Francisci Morales, and Ashraf Aboulnaga. Link prediction via higher-order motif features. In *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2019, Würzburg, Germany, September 16–20, 2019, Proceedings, Part I*, pages 412–429. Springer, 2020.

[29] Mayank Lahiri and Tanya Y Berger-Wolf. Structure prediction in temporal networks using frequent subgraphs. In *2007 IEEE Symposium on computational intelligence and data mining*, pages 35–42. IEEE, 2007.

[30] Mahmudur Rahman and Mohammad Al Hasan. Link prediction in dynamic networks using graphlet. In *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2016, Riva del Garda, Italy, September 19-23, 2016, Proceedings, Part I 16*, pages 394–409. Springer, 2016.

[31] Giang H Nguyen, John Boaz Lee, Ryan A Rossi, Nesreen K Ahmed, Eunyee Koh, and Sungchul Kim. Dynamic network embeddings: From random walks to temporal random walks. In *2018 IEEE International Conference on Big Data (Big Data)*, pages 1085–1092. IEEE, 2018.

[32] Yanbang Wang, Yen-Yu Chang, Yunyu Liu, Jure Leskovec, and Pan Li. Inductive representation learning in temporal networks via causal anonymous walks. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=KYPz4YsCPj.

[33] Ming Jin, Yuan-Fang Li, and Shirui Pan. Neural temporal walks: Motif-aware representation learning on continuous-time dynamic graphs. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=NqbktPUkZf7.

[34] Zhining Liu, Dawei Zhou, Yada Zhu, Jinjie Gu, and Jingrui He. Towards fine-grained temporal network representation via time-reinforced random walk. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 4973–4980, 2020.

[35] 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.

[36] Timoteo Carletti, Federico Battiston, Giulia Cencetti, and Duccio Fanelli. Random walks on hypergraphs. *Physical review E*, 101(2):022308, 2020.

[37] Koby Hayashi, Sinan G Aksoy, Cheong Hee Park, and Haesun Park. Hypergraph random walks, laplacians, and clustering. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, pages 495–504, 2020.

[38] Josh Payne. Deep hyperedges: a framework for transductive and inductive learning on hypergraphs, 2019.

[39] Uthsav Chitra and Benjamin Raphael. Random walks on hypergraphs with edge-dependent vertex weights. In *International conference on machine learning*, pages 1172–1181. PMLR, 2019.

[40] Sinan G. Aksoy, Cliff Joslyn, Carlos Ortiz Marrero, Brenda Praggastis, and Emilie Purvine. Hypernetwork science via high-order hypergraph walks. *EPJ Data Science*, 9(1):16, Jun 2020. ISSN 2193-1127. doi: 10.1140/epjds/s13688-020-00231-0. URL https://doi.org/10.1140/epjds/s13688-020-00231-0.

[41] Austin R. Benson, David F. Gleich, and Lek-Heng Lim. The spacey random walk: A stochastic process for higher-order data. *SIAM Review*, 59(2):321–345, 2017. doi: 10.1137/16M1074023. URL https://doi.org/10.1137/16M1074023.

[42] Ankit Sharma, Shafiq Joty, Himanshu Kharkwal, and Jaideep Srivastava. Hyperedge2vec: Distributed representations for hyperedges, 2018. URL https://openreview.net/forum?id=rJ5C67-C-.

[43] Ilya Tolstikhin, Neil Houlsby, Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Thomas Unterthiner, Jessica Yung, Andreas Peter Steiner, Daniel Keysers, Jakob Uszkoreit, Mario Lucic, and Alexey Dosovitskiy. MLP-mixer: An all-MLP architecture for vision. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, 2021. URL https://openreview.net/forum?id=EI2KOXKdnP.

[44] Silvio Micali and Zeyuan Allen Zhu. Reconstructing markov processes from independent and anonymous experiments. *Discrete Applied Mathematics*, 200:108–122, 2016. ISSN 0166-218X. doi: https://doi.org/10.1016/j.dam.2015.06.035. URL https://www.sciencedirect.com/science/article/pii/S0166218X15003212.

[45] Youngjoo Seo, Michaël Defferrard, Pierre Vandergheynst, and Xavier Bresson. Structured sequence modeling with graph convolutional recurrent networks. In *International conference on neural information processing*, pages 362–373. Springer, 2018.

[46] Ling Zhao, Yujiao Song, Chao Zhang, Yu Liu, Pu Wang, Tao Lin, Min Deng, and Haifeng Li. T-gcn: A temporal graph convolutional network for traffic prediction. *IEEE Transactions on Intelligent Transportation Systems*, 21(9):3848–3858, 2019.

[47] Hao Peng, Hongfei Wang, Bowen Du, Md Zakirul Alam Bhuiyan, Hongyuan Ma, Jianwei Liu, Lihong Wang, Zeyu Yang, Linfeng Du, Senzhang Wang, et al. Spatial temporal incidence dynamic graph neural networks for traffic flow forecasting. *Information Sciences*, 521:277–290, 2020.

[48] Xiaoyang Wang, Yao Ma, Yiqi Wang, Wei Jin, Xin Wang, Jiliang Tang, Caiyan Jia, and Jian Yu. Traffic flow prediction via spatial temporal graph neural network. In *Proceedings of The Web Conference 2020*, pages 1082–1092, 2020.

[49] Jiaxuan You, Tianyu Du, and Jure Leskovec. Roland: Graph learning framework for dynamic graphs. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '22, page 2358–2366, New York, NY, USA, 2022. Association for Computing Machinery. ISBN 9781450393850. doi: 10.1145/3534678.3539300. URL https://doi.org/10.1145/3534678.3539300.

[50] Farnoosh Hashemi, Ali Behrouz, and Milad Rezaei Hajidehi. Cs-tgn: Community search via temporal graph neural networks. In *Companion Proceedings of the Web Conference 2023*, WWW '23, New York, NY, USA, 2023. Association for Computing Machinery. doi: 10.1145/3543873.3587654. URL https://doi.org/10.1145/3543873.3587654.

[51] 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 & Data Mining*, KDD '19, page 1269–1278, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450362016. doi: 10.1145/3292500.3330895. URL https://doi.org/10.1145/3292500.3330895.

[52] Ali Behrouz and Margo Seltzer. Anomaly detection in multiplex dynamic networks: from blockchain security to brain disease prediction. In *NeurIPS 2022 Temporal Graph Learning Workshop*, 2022. URL https://openreview.net/forum?id=UDGZDfwmay.

[53] Sudhanshu Chanpuriya, Ryan A. Rossi, Sungchul Kim, Tong Yu, Jane Hoffswell, Nedim Lipka, Shunan Guo, and Cameron N Musco. Direct embedding of temporal network edges via time-decayed line graphs. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=Qamz7Q_Ta1k.

[54] Weilin Cong, Si Zhang, Jian Kang, Baichuan Yuan, Hao Wu, Xin Zhou, Hanghang Tong, and Mehrdad Mahdavi. Do we really need complicated model architectures for temporal networks? In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=ayPPc0SyLv1.

[55] Yuhong Luo and Pan Li. Neighborhood-aware scalable temporal network representation learning. In *The First Learning on Graphs Conference*, 2022. URL https://openreview.net/forum?id=EPUtNe7a9ta.

[56] Benjamin Paul Chamberlain, Sergey Shirobokov, Emanuele Rossi, Fabrizio Frasca, Thomas Markovich, Nils Yannick Hammerla, Michael M. Bronstein, and Max Hansmire. Graph neural networks for link prediction with subgraph sketching. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=m1oqEOAozQU.

[57] Can Chen and Yang-Yu Liu. A survey on hyperlink prediction. *arXiv preprint arXiv:2207.02911*, 2022.

[58] Devanshu Arya, Deepak Gupta, Stevan Rudinac, and Marcel Worring. Hyper{sage}: Generalizing inductive representation learning on hypergraphs, 2021. URL https://openreview.net/forum?id=cKnKJcTPRcV.

[59] Devanshu Arya, Deepak K Gupta, Stevan Rudinac, and Marcel Worring. Adaptive neural message passing for inductive learning on hypergraphs. *arXiv preprint arXiv:2109.10683*, 2021.

[60] Sameer Agarwal, Kristin Branson, and Serge Belongie. Higher order learning with graphs. In *Proceedings of the 23rd international conference on Machine learning*, pages 17–24, 2006.

[61] Chaoqi Yang, Ruijie Wang, Shuochao Yao, and Tarek Abdelzaher. Hypergraph learning with line expansion. *arXiv preprint arXiv:2005.04843*, 2020.

[62] Kaize Ding, Jianling Wang, Jundong Li, Dingcheng Li, and Huan Liu. Be more with less: Hypergraph attention networks for inductive text classification. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4927–4936, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.399. URL https://aclanthology.org/2020.emnlp-main.399.

[63] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 855–864, 2016.

[64] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 701–710, 2014.

[65] Xin-Jian Xu, Chong Deng, and Li-Jie Zhang. Hyperlink prediction via local random walks and jensen–shannon divergence. *Journal of Statistical Mechanics: Theory and Experiment*, 2023(3):033402, mar 2023. doi: 10.1088/1742-5468/acc31e. URL https://dx.doi.org/10.1088/1742-5468/acc31e.

[66] Valerio La Gatta, Vincenzo Moscato, Mirko Pennone, Marco Postiglione, and Giancarlo Sperlí. Music recommendation via hypergraph embedding. *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–13, 2022. doi: 10.1109/TNNLS.2022.3146968.

[67] Dingqi Yang, Bingqing Qu, Jie Yang, and Philippe Cudre-Mauroux. Revisiting user mobility and social relationships in lbsns: A hypergraph embedding approach. In *The World Wide Web Conference*, WWW '19, page 2147–2157, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450366748. doi: 10.1145/3308558.3313635. URL https://doi.org/10.1145/3308558.3313635.

[68] Aurélien Ducournau and Alain Bretto. Random walks in directed hypergraphs and application to semi-supervised image segmentation. *Computer Vision and Image Understanding*, 120:91–102, 2014. ISSN 1077-3142. doi: https://doi.org/10.1016/j.cviu.2013.10.012. URL https://www.sciencedirect.com/science/article/pii/S1077314213002038.

[69] Lei Ding and Alper Yilmaz. Interactive image segmentation using probabilistic hypergraphs. *Pattern Recognition*, 43(5):1863–1873, 2010. ISSN 0031-3203. doi: https://doi.org/10.1016/j.patcog.2009.11.025. URL https://www.sciencedirect.com/science/article/pii/S0031320309004440.

[70] Sai Nageswar Satchidanand, Harini Ananthapadmanaban, and Balaraman Ravindran. Extended discriminative random walk: A hypergraph approach to multi-view multi-relational transductive learning. In *Proceedings of the 24th International Conference on Artificial Intelligence*, IJCAI'15, page 3791–3797. AAAI Press, 2015. ISBN 9781577357384.

[71] Jie Huang, Chuan Chen, Fanghua Ye, Jiajing Wu, Zibin Zheng, and Guohui Ling. Hyper2vec: Biased random walk for hyper-network embedding. In Guoliang Li, Jun Yang, Joao Gama, Juggapong Natwichai, and Yongxin Tong, editors, *Database Systems for Advanced Applications*, pages 273–277, Cham, 2019. Springer International Publishing. ISBN 978-3-030-18590-9.

[72] Jie Huang, Xin Liu, and Yangqiu Song. Hyper-path-based representation learning for hyper-networks. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, CIKM '19, page 449–458, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450369763. doi: 10.1145/3357384.3357871. URL https://doi.org/10.1145/3357384.3357871.

[73] Jie Huang, Chuan Chen, Fanghua Ye, Weibo Hu, and Zibin Zheng. Nonuniform hyper-network embedding with dual mechanism. *ACM Trans. Inf. Syst.*, 38(3), may 2020. ISSN 1046-8188. doi: 10.1145/3388924. URL https://doi.org/10.1145/3388924.

[74] Yunyu Liu, Jianzhu Ma, and Pan Li. Neural predicting higher-order patterns in temporal networks. In *Proceedings of the ACM Web Conference 2022*, WWW '22, page 1340–1351, New York, NY, USA, 2022. Association for Computing Machinery. ISBN 9781450390965. doi: 10.1145/3485447.3512181. URL https://doi.org/10.1145/3485447.3512181.

[75] Jiying Zhang, Fuyang Li, Xi Xiao, Tingyang Xu, Yu Rong, Junzhou Huang, and Yatao Bian. Hypergraph convolutional networks via equivalency between hypergraphs and undirected graphs, 2022. URL https://openreview.net/forum?id=zFyCvjXof60.

[76] Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus), 2020.

[77] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[78] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.

[79] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[80] da Xu, chuanwei ruan, evren korpeoglu, sushant kumar, and kannan achan. Inductive representation learning on temporal graphs. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=rJeW1yHYwH.

[81] Da Xu, Chuanwei Ruan, Evren Korpeoglu, Sushant Kumar, and Kannan Achan. Self-attention with functional time representation learning. *Advances in neural information processing systems*, 32, 2019.

[82] Seyed Mehran Kazemi, Rishab Goel, Sepehr Eghbali, Janahan Ramanan, Jaspreet Sahota, Sanjay Thakur, Stella Wu, Cathal Smyth, Pascal Poupart, and Marcus Brubaker. Time2vec: Learning a vector representation of time. *arXiv preprint arXiv:1907.05321*, 2019.

[83] Naganand Yadati, Vikram Nitin, Madhav Nimishakavi, Prateek Yadav, Anand Louis, and Partha Talukdar. Nhp: Neural hypergraph link prediction. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, pages 1705–1714, 2020.

[84] Tarun Kumar, K Darwin, Srinivasan Parthasarathy, and Balaraman Ravindran. Hpra: Hyperedge prediction using resource allocation. In *12th ACM conference on web science*, pages 135–143, 2020.

[85] Ye Xu, Dan Rockmore, and Adam M Kleinbaum. Hyperlink prediction in hypernetworks using latent social features. In *Discovery Science: 16th International Conference, DS 2013, Singapore, October 6-9, 2013. Proceedings 16*, pages 324–339. Springer, 2013.

[86] Aldo Pareja, Giacomo Domeniconi, Jie Chen, Tengfei Ma, Toyotaro Suzumura, Hiroki Kanezashi, Tim Kaler, Tao Schardl, and Charles Leiserson. Evolvegcn: Evolving graph convolutional networks for dynamic graphs. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 5363–5370, 2020.

[87] Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. Simplifying graph convolutional networks. In *International conference on machine learning*, pages 6861–6871. PMLR, 2019.

[88] Rossana Mastrandrea, Julie Fournet, and Alain Barrat. Contact patterns in a high school: A comparison between data collected using wearable sensors, contact diaries and friendship surveys. *PLOS ONE*, 10(9):e0136497, 2015. doi: 10.1371/journal.pone.0136497. URL https://doi.org/10.1371/journal.pone.0136497.

[89] Juliette Stehlé, Nicolas Voirin, Alain Barrat, Ciro Cattuto, Lorenzo Isella, Jean-François Pinton, Marco Quaggiotto, Wouter Van den Broeck, Corinne Régis, Bruno Lina, and Philippe Vanhems. High-resolution measurements of face-to-face contact patterns in a primary school. *PLoS ONE*, 6(8):e23176, 2011. doi: 10.1371/journal.pone.0023176. URL https://doi.org/10.1371/journal.pone.0023176.

[90] James H. Fowler. Connecting the congress: A study of cosponsorship networks. *Political Analysis*, 14(04):456–487, 2006. doi: 10.1093/pan/mpl002. URL https://doi.org/10.1093/pan/mpl002.

[91] James H. Fowler. Legislative cosponsorship networks in the US house and senate. *Social Networks*, 28(4):454–465, oct 2006. doi: 10.1016/j.socnet.2005.11.003. URL https://doi.org/10.1016/j.socnet.2005.11.003.

[92] Hao Yin, Austin R. Benson, Jure Leskovec, and David F. Gleich. Local higher-order graph clustering. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM Press, 2017. doi: 10.1145/3097983.3098069. URL https://doi.org/10.1145/3097983.3098069.

[93] Fan RK Chung. The laplacian of a hypergraph. In "", 1993.

[94] Timoteo Carletti, Duccio Fanelli, and Renaud Lambiotte. Random walks and community detection in hypergraphs. *Journal of Physics: Complexity*, 2(1):015011, 2021.

[95] Linyuan Lu and Xing Peng. High-order random walks and generalized laplacians on hypergraphs. *Internet Mathematics*, 9(1):3–32, 2013.

[96] Chenzi Zhang, Shuguang Hu, Zhihao Gavin Tang, and TH Hubert Chan. Re-revisiting learning on hypergraphs: confidence interval and subgradient method. In *International Conference on Machine Learning*, pages 4026–4034. PMLR, 2017.

[97] T-H Hubert Chan, Zhihao Gavin Tang, Xiaowei Wu, and Chenzi Zhang. Diffusion operator and spectral analysis for directed hypergraph laplacian. *Theoretical Computer Science*, 784: 46–64, 2019.

[98] Pan Li and Olgica Milenkovic. Inhomogeneous hypergraph clustering with applications. *Advances in neural information processing systems*, 30, 2017.

[99] Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. *Advances in neural information processing systems*, 30, 2017.

[100] Marta Garnelo, Dan Rosenbaum, Christopher Maddison, Tiago Ramalho, David Saxton, Murray Shanahan, Yee Whye Teh, Danilo Rezende, and SM Ali Eslami. Conditional neural processes. In *International conference on machine learning*, pages 1704–1713. PMLR, 2018.

[101] David Lopez-Paz, Robert Nishihara, Soumith Chintala, Bernhard Scholkopf, and Léon Bottou. Discovering causal signals in images. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6979–6987, 2017.

[102] Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Russ R Salakhutdinov, and Alexander J Smola. Deep sets. *Advances in neural information processing systems*, 30, 2017.

[103] Juho Lee, Yoonho Lee, Jungtaek Kim, Adam Kosiorek, Seungjin Choi, and Yee Whye Teh. Set transformer: A framework for attention-based permutation-invariant neural networks. In *International conference on machine learning*, pages 3744–3753. PMLR, 2019.

[104] Jinheon Baek, Minki Kang, and Sung Ju Hwang. Accurate learning of graph representations with graph multiset pooling. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=JHcqXGaqiGn.

[105] Ronald Harry Atkin. *Mathematical structure in human affairs*. Heinemann Educational London, 1974.

[106] David I Spivak. Higher-dimensional models of networks. *arXiv preprint arXiv:0909.4314*, 2009.

[107] Jacob Charles Wright Billings, Mirko Hu, Giulia Lerda, Alexey N Medvedev, Francesco Mottes, Adrian Onicas, Andrea Santoro, and Giovanni Petri. Simplex2vec embeddings for community detection in simplicial complexes. *arXiv preprint arXiv:1906.09068*, 2019.

[108] Mustafa Hajij, Kyle Istvan, and Ghada Zamzmi. Cell complex neural networks. In *TDA & Beyond*, 2020. URL https://openreview.net/forum?id=6Tq18ySFpGU.

[109] Celia Hacker. k-simplex2vec: a simplicial extension of node2vec. In *TDA & Beyond*, 2020. URL https://openreview.net/forum?id=Aw9DUXPjq55.

[110] Stefania Ebli, Michaël Defferrard, and Gard Spreemann. Simplicial neural networks. In *TDA & Beyond*, 2020. URL https://openreview.net/forum?id=nPCt39DVIfk.

[111] Maosheng Yang, Elvin Isufi, and Geert Leus. Simplicial convolutional neural networks. In *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 8847–8851. IEEE, 2022.

[112] Eric Bunch, Qian You, Glenn Fung, and Vikas Singh. Simplicial 2-complex convolutional neural networks. In *TDA & Beyond*, 2020. URL https://openreview.net/forum?id=TLbnsKrt6J-.

[113] Christopher Wei Jin Goh, Cristian Bodnar, and Pietro Lio. Simplicial attention networks. In *ICLR 2022 Workshop on Geometrical and Topological Representation Learning*, 2022. URL https://openreview.net/forum?id=ScfRNWkpec.

[114] Mustafa Hajij, Karthikeyan Natesan Ramamurthy, Aldo Guzmán-Sáenz, and Ghada Za. High skip networks: A higher order generalization of skip connections. In *ICLR 2022 Workshop on Geometrical and Topological Representation Learning*, 2022.

[115] Matthias Hein, Simon Setzer, Leonardo Jost, and Syama Sundar Rangapuram. The total variation on hypergraphs - learning on hypergraphs revisited. In C.J. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc., 2013. URL https://proceedings.neurips.cc/paper_files/paper/2013/file/8a3363abe792db2d8761d6403605aeb7-Paper.pdf.

[116] Sameer Agarwal, Kristin Branson, and Serge Belongie. Higher order learning with graphs. In *Proceedings of the 23rd International Conference on Machine Learning*, ICML '06, page 17–24, New York, NY, USA, 2006. Association for Computing Machinery. ISBN 1595933832. doi: 10.1145/1143844.1143847. URL https://doi.org/10.1145/1143844.1143847.

[117] Edmund Ihler, Dorothea Wagner, and Frank Wagner. Modeling hypergraphs by graphs with the same mincut properties. *Inf. Process. Lett.*, 45(4):171–175, mar 1993. ISSN 0020-0190. doi: 10.1016/0020-0190(93)90115-P. URL https://doi.org/10.1016/0020-0190(93)90115-P.

[118] Patrick Kidger, James Morrill, James Foster, and Terry Lyons. Neural controlled differential equations for irregular time series. *Advances in Neural Information Processing Systems*, 33: 6696–6707, 2020.

[119] 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*, WWW '18, page 969–976, Republic and Canton of Geneva, CHE, 2018. International World Wide Web Conferences Steering Committee. ISBN 9781450356404. doi: 10.1145/3184558.3191526. URL https://doi.org/10.1145/3184558.3191526.

[120] Weilin Cong, Si Zhang, Jian Kang, Baichuan Yuan, Hao Wu, Xin Zhou, Hanghang Tong, and Mehrdad Mahdavi. Do we really need complicated model architectures for temporal networks? In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=ayPPc0SyLv1.