Oversquashing in Hypergraph Neural Networks

Naganand Yadati
National University of Singapore*
y.naganand@gmail.com

Abstract

Message Passing Neural Networks (MPNNs) are a type of Graph Neural Networks (GNNs) that utilise the graph structure to facilitate the exchange of messages along the edges. On the one hand, the inductive bias gives rise to a phenomenon termed "over-squashing," where a vertex's hidden feature becomes insensitive to information present in distant vertices. On the other hand, there has been a recent wave of innovations involving the adaptation of MPNNs and GNNs to hypergraphs, which relax the notion of an edge to a hyperedge containing a subset of vertices. In recent times, MPNNs and GNNs have found application in web datasets, spanning both graph and hypergraph scenarios. However, there exists a research gap regarding the investigation of over-squashing within hypergraph neural networks. Our paper contributes to bridging this precise research gap by investigating several methods each belonging to one of two distinct classes of hypergraph neural networks. To begin with, we introduce three novel tests termed HyperEdgeSingle, HyperEdgePath, and HyperEdgeRing designed specifically for assessing the phenomenon of over-squashing within hypergraph neural networks. Through theoretical and experimental analyses, we reveal a counter-intuitive and significant finding: advanced state-of-the-art hypergraph neural networks are more susceptible to over-squashing than their predecessors. Validation of the findings is reinforced through experiments conducted on real-world datasets.

1 Introduction

In the ever-expanding universe of relational data analysis, the synergy between graph theory and machine learning has paved the way for advanced techniques. Graph Neural Networks (GNNs) [1], a type of Message Passing Neural Networks (MPNNs), have become the cornerstone for understanding intricate relationships within relational datasets.

Hypergraphs, a natural extension of graphs, capture richer interactions beyond the binary edges of graphs among data entities, providing a nuanced perspective. Hypergraphs naturally suit the intricate nature of web data, allowing us to represent group relationships and higher-order interactions seamlessly. For example, hyperedges in hypergraphs can naturally model all the references of a document in web citation graphs, all the documents co-authored by an author in web academic graphs, and all products purchased together by a customer in user-item interaction networks collected from online platforms or websites (e.g., Walmart).

There has been a recent wave of innovations involving the adaptation of MPNNs and GNNs to hypergraphs which we collectively call Hypegraph Neural Networks (HNNs). Most HNNs in existing literature exchange messages among entities, which can occur either between vertices or between vertices and hyperedges. The inductive bias of the message passing mechanism gives rise to a phenomenon termed "over-squashing," where a vertex's hidden feature becomes insensitive to information present in distant vertices. Despite the extensive research on oversquashing in traditional GNNs, the phenomenon remains unexplored in the realm of HNNs, to the best of our knowledge. This paper fills this precise research gap and makes the following contributions:

*work done while at NUS.

Yadati et al., Oversquashing in Hypergraph Neural Networks. *Proceedings of the Third Learning on Graphs Conference (LoG 2024)*, PMLR 269, Virtual Event, November 26–29, 2024.

- We present a theoretical analysis to understand oversquashing in HNNs (Section 3).
- We introduce three novel tests termed HyperEdgeSingle, HyperEdgePath, and HyperEdgeRing to assess the phenomenon of oversquashing in HNNs (Section 4).
- Our categorisation of existing HNNs into two distinct categories, particularly highlighting the newer models, reveals through both theory and practical experimentation that newer models are significantly more prone to oversquashing compared to their predecessors (Sections 2, 4, and 6).
- Our theory motivates a novel HNN, named SensHNN, to mitigate oversquashing (Section 5).
- We show that SensHNN achieves superior performance on the three tests and competitive performance on real-world data compared to state-of-the-art HNNs (Section 6).

2 Related Work

Graph Neural Networks (GNNs). At their core, GNNs aim to learn and propagate information across vertices in a graph, allowing them to capture relationships, dependencies, and patterns within structured datasets [1–7]. Popular GNNs such as GCN [8], GAT [9], GraphSAGE [10], and GIN [11] fall within the broader message passing neural networks [12].

Oversquashing in GNNs. The discovery of oversquashing in GNNs [13] revealed a new challenge in training GNNs to *propagate information across distant nodes* in the input graph. Subsequently, oversquashing has been investigated by evaluating how sensitive GNN hidden node features are to the initial features of a distant node, a process that has included the use of Jacobian analysis [14–18]. There have been attempts to mitigate the phenomenon [19–24] which encompass rewiring [25–27].

Hypergraph Neural Networks (HNNs). In recent times, researchers have enhanced the versatility of GNNs by creating HNNs to handle hyperedges in hypergraphs [28, 29]. Earlier models such as HGNN [30] with weighted clique expansion, HyperGCN [31] featuring non-linear Laplacian, and HCHA [32] incorporating attention on clique expansion are vertex-centric. More recent HNNs such as the AllDeepSets and AllSetTransformer [33], equivariant models [34, 35], attention on star expansion [36–39] among others [40, 41] compute hyperedge embeddings.

3 Theory of Oversquashing in HNNs

3.1 Problem Setup

Let $\mathcal{H} = (V, E)$ be a hyprgraph with vertices $V = \{1, \cdots, n\}$ and edges $E \subseteq 2^V$. We make the assumption that $\mathcal{H}$ includes initial vertex features $\{\mathbf{h}_v^{(0)}\}_{v \in V} \subset \mathbb{R}^d$. We can view HNNs as functions parameterised by θ represented as $\mathsf{HNN}_\theta : (\mathcal{H}, \{\mathbf{h}_v^{(0)}\}_{v \in V}) \mapsto \mathcal{Y}$. We seek to estimate the parameters in θ to generate an output in $\mathcal{Y}$ through training. This output is associated with common tasks, such as vertex classification and hyperedge prediction. The same setup adapts to hypergraph-level tasks, such as hypergraph classification. Furthermore, we let $\phi^{(l)}$ represent the layer-specific aggregation operation and $\psi^{(l)}$ the combine operation at layer l used in a message passing neural network [12].

We categorise HNNs as follows:

Vertex-centric HNNs (VC-HNNs). Models in this category follow the following form:

$$\mathbf{h}_v^{(l+1)} = \psi^{(l)} \left(\mathbf{h}_v^{(l)}, \phi^{(l)} \left(\{\mathbf{h}_u^{(l)} : \{v, u\} \subseteq E\} \right) \right),$$

for $l = 0, \cdots, L-1$ layers. Here, $\phi^{(l)}$ are permutation-invariant aggregation functions while $\psi^{(l)}$ combine the self-state with messages from the aggregated representations.

Vertex-Hyperedge HNNs (VH-HNNs). These models have the specific capability to compute embeddings for hyperedges and can handle initial hyperedge features $\{\mathbf{h}_e^{(0)}\}_{e \in E} \subset \mathbb{R}^d$ [2]. The propagation rules in each hidden layer are of the form:

$$\mathbf{h}_e^{(l+1)} = \psi_V^{(l)} \left(\mathbf{h}_e^{(l)}, \phi_V^{(l)} \left(\{\mathbf{h}_v^{(l)} : v \in e\} \right) \right), \quad \mathbf{h}_v^{(l+1)} = \psi_E^{(l)} \left(\mathbf{h}_v^{(l)}, \phi_E^{(l)} \left(\{\mathbf{h}_e^{(l)} : v \in e\} \right) \right),$$

[2] We assume d is the same for initial vertex and initial hyperedge features for simplicity. Our analysis can be easily extended to handle variable sizes.

Atleast two alternative approaches for generalising HNNs were presented previously, specifically, UniGNN [37] and AllSet [33]. In these generalisations, HNN models from VC-HNNs and VH-HNNs can be viewed as specific instances. Nonetheless, our categorisation of HNNs into VC-HNNs and VH-HNNs is rooted in our observation that these two categories exhibit significantly distinct effects of oversquashing, a conclusion supported by both theoretical and empirical analyses in our paper.

Background on Oversquashing. Both VC-HNNs and VH-HNNs follow the message-passing paradigm of an MPNN [12], where messages are exchanged to update hidden representations. For a vertex $v \in V$ to be influenced by features at a shortest-path distance r in the topology, the MPNN must have at least r layers. A key observation [13] is that expanding a vertex's receptive field in an MPNN causes an exponential increase in the number of messages as r grows.

Recent research on GNNs [14] has substantiated that in scenarios where vertices v and u are separated by a large number of hops in the graph, the sensitivity of hidden features of v with respect to the initial features of u is small. Mathematically, the sensitivity measured by the use of Jacobian can be bounded as follows: $\left\|\partial h_v^{(r)}/\partial h_u^{(0)}\right\| \leq c \cdot (A^r)_{vu},$. Here A is the symmetrically normalised adjacency of the input graph and c is a constant determined by the Lipschitz regularity properties specific to the MPNN. When the entry $(A^r)_{vu}$ is tiny, e.g., when the two vertices are connected by a limited number of paths with "bottlenecked" edges [13, 14, 17], it implies that the hidden feature of node v is insensitive to the information originating from node u.

3.2 Oversquashing in HNNs: Challenges

Adapting existing theory from GNNs involves addressing the following technical challenges:

- *Hyperedge Embedding Incorporation:* VH-HNNs can generate embeddings for hyperedges through a hyperedge-specific aggregation ϕ and combine ψ operations, e.g., AllSet [33].
- *Layer-dependent Aggregation:* In contrast to GNNs, certain HNNs depict the input hypergraph using layer-dependent graphs, each characterised by highly distinct neighbourhood properties. This leads to significantly varied aggregation mechanisms across layers, e.g., HyperGCN [31].
- *Hyperedge Size Variability:* Hyperedges in hypergraphs vary significantly in size, making it challenging to generalise oversquashing frameworks on graphs in a straightforward manner.

3.3 Oversquashing in HNNs: Results

The first result sets an upper limit on how much the hidden features of a vertex v can change in response to variations in the initial features of a distant vertex u without making any assumptions about the number of layers in the neural network. In contrast to previous research on GNNs [14], our analysis considers arbitrary layer numbers of the HNN. A pivotal implication of the theorem is that the upper bound is influenced by the hypergraph's topology, emphasising its role as a significant factor in oversquashing.

In the HNN's update function $\psi^{(l)}$, consider that it combines the aggregated representation of other entities with weight $w \leq 1$ while giving the self-representation a weight of $(1-w)$, i.e.,

$$\mathbf{h}_v^{(l+1)} = \psi^{(l)}\left(\mathbf{h}_v^{(l)}, \phi^{(l)}\left(\{\mathbf{h}_u^{(l)} : u \subset V \cup E\}\right)\right) = (1-w)h_v^{(l)} + w\phi^{(l)}\left(\{\mathbf{h}_u^{(l)} : u \subset V \cup E\}\right)$$

This update function is category-agnostic. To clarify, we are examining those HNNs where all instances of $\psi^{(l)}$ in VC-HNNs and all instances of $\psi_V^{(l)}$ and $\psi_E^{(l)}$ in VH-HNNs, for $0, \cdots, L-1$, adhere to the same aforementioned form.

Theorem 3.1. *Let $v, u \in V \cup E$, $l \in \mathbb{N}$. If HNN belongs to the VH-HNN category, assume that $\left\|\nabla\psi_V^{(j)}\right\| \leq C_\psi$, $\left\|\nabla\psi_E^{(j)}\right\| \leq C_\psi$, and $\max\{\left\|\nabla\phi_V^{(j)}\right\|, 1\} \leq C_\phi$, $\max\{\left\|\nabla\phi_E^{(j)}\right\|, 1\} \leq C_\phi$, where ∇f denotes the Jacobian of a map f. If HNN belongs to the VC-HNN category, assume that $\left\|\nabla\psi^{(j)}\right\| \leq C_\psi$ and $\max\{\left\|\nabla\phi^{(j)}\right\|, 1\} \leq C_\phi$ for all $j = 0, \ldots, l$. Further, let the layer-dependent symmetrically degree normalised matrices of $\phi^{(j)}$ be $A^{(j)}$. Then the sensitivity satisfies*

$$\left\|\frac{\partial h_v^{(l)}}{\partial h_u^{(0)}}\right\| \leq (2C_\psi C_\phi)^l \sum_{j=0}^{l} (T^{(j)})_{vu}, \textit{where } T^{(j)} = \prod_{i=0}^{j}\left((1-w)I + wA^{(i)}\right). \tag{1}$$

Please see Appendix Section A.1 for a proof. The oversquashing phenomenon arises when the right-hand side of Eq. (1) is small, esp. when l is set to be at least r, the shortest-path distance between v and u. A small derivative of $\mathbf{h}_v^{(l)}$ with respect to $\mathbf{h}_u^{(0)}$ indicates that, after passing through l HNN layers, *the hidden features at vertex v become insensitive to the initial information originating from vertex u*, signifying an ineffective message propagation.

The second result connects oversquashing to resistance distance. Unlike shortest-path distance, resistance distance decreases when multiple short paths exist between vertices v and u, considering a set of short paths rather than a single route.

Resistance Distance. The "closeness" between vertices v and u depends on whether they are linked by a short path within the same densely connected region of the graph. When v and u fall within the boundaries of a shared high-density cluster in the graph, their resistance distance is evaluated as smaller, indicating greater closeness. Conversely, if they are connected by a short path but are situated in different high-density clusters, their resistance distance is a larger value [42, 43].

Let u and v be two distinct vertices of the (hyper)graph. The ***resistance distance*** between u and v is defined as $R_{v,u} = (1_v - 1_u)^T L^+ (1_v - 1_u)$, where 1_v is the indicator vector of the vertex v and L^+ is the pseudoinverse of the (hyper)graph Laplacian matrix L.

Jacobian Obstruction. Let u and v, be separated by a distance $r \gg 1$. We are interested in the transmission of information from vertex u to vertex v. Within the HNN, specifically at two layers k and l with $k < l$, we envision that $\mathbf{h}_v^{(l)}$ will be more influenced by its own representation at the earlier layer, i.e., $\mathbf{h}_v^{(k)}$, as compared to $\mathbf{h}_u^{(k)}$. In light of this, we bring forth the quantity: $\tilde{\mathbf{J}}_k^{(l)}(v,u) := \frac{1}{d_v}\frac{\partial \mathbf{h}_v^{(l)}}{\partial \mathbf{h}_v^{(k)}} - \frac{1}{\sqrt{d_v d_u}}\frac{\partial \mathbf{h}_v^{(l)}}{\partial \mathbf{h}_u^{(k)}}$.

Intuition. Consider the scenario where, at layer l in the HNN, vertex v exhibits limited sensitivity to the information relayed from the representation of vertex u at layer k. As a general trend, we expect that $\|\partial \mathbf{h}_v^{(l)}/\partial \mathbf{h}_u^{(k)}\| \ll \|\partial \mathbf{h}_v^{(l)}/\partial \mathbf{h}_v^{(k)}\|$. In the opposite scenario, we generally expect that $\|\partial \mathbf{h}_v^{(m)}/\partial \mathbf{h}_u^{(k)}\| \sim \|\partial \mathbf{h}_v^{(m)}/\partial \mathbf{h}_v^{(k)}\|$. Consequently, $\|\tilde{\mathbf{J}}_k^{(l)}(v,u)\|$ will be *large* when the information at v and that at u both at layer k are (approximately) unrelated. We can extend this reasoning to consider messages at each layer $k \leq m$.

Inspired by the symmetry of resistance distance $R_{u,v}$, we will now focus on the bidirectional exchange of information between vertices v and u. This differs from our previous quantity, which centered on vertex v receiving information from vertex u. We introduce the following symmetric quantity:

$$\mathbf{J}_k^{(l)}(v,u) := \tilde{\mathbf{J}}_k^{(l)}(v,u) + \tilde{\mathbf{J}}_k^{(l)}(u,v)\Big(\frac{1}{d_v}\frac{\partial \mathbf{h}_v^{(l)}}{\partial \mathbf{h}_v^{(k)}} - \frac{1}{\sqrt{d_v d_u}}\frac{\partial \mathbf{h}_v^{(l)}}{\partial \mathbf{h}_u^{(k)}}\Big) + \Big(\frac{1}{d_u}\frac{\partial \mathbf{h}_u^{(l)}}{\partial \mathbf{h}_u^{(k)}} - \frac{1}{\sqrt{d_v d_u}}\frac{\partial \mathbf{h}_u^{(l)}}{\partial \mathbf{h}_v^{(k)}}\Big).$$

As before, we anticipate that $\|\mathbf{J}_k^{(l)}(v,u)\|$ will be greater when vertices v and u do not effectively communicate within the HNN. Conversely, it will be smaller when communication is robust. We apply the same idea at each layer $k \leq l$.

Definition 3.2. *The symmetric Jacobian obstruction of vertices v, u at layer l is* $\mathsf{O}^{(l)}(v,u) = \sum_{k=0}^{m} \|\mathbf{J}_k^{(l)}(v,u)\|$.

Assumption 3.3. *In the HNN's computation graph, every path is activated with an equal probability, p, of success [44, 45].*

With this understanding, we can now introduce the second result:

Theorem 3.4. *[[16]] Consider an HNN with ν the maximal spectral norm of the weight matrices. Let Assumption 3.3 hold. If $\nu \leq 1$, in expectation, we have $\mathsf{O}^{(l)}(v,u) \leq \frac{p}{\nu w} R_{v,u}$.*

Please see Appendix Section A.2 for a proof. A lower value of $O^{(l)}(v,u)$ indicates that the sensitivity between v and u in the HNN is stronger (and conversely, when $O^{(l)}(v,u)$ is higher). Hence, Theorem 3.4 suggests that vertex pairs characterised by shorter resistance distances will facilitate more efficient information exchange within an HNN, while the opposite holds true for vertices with longer distances.

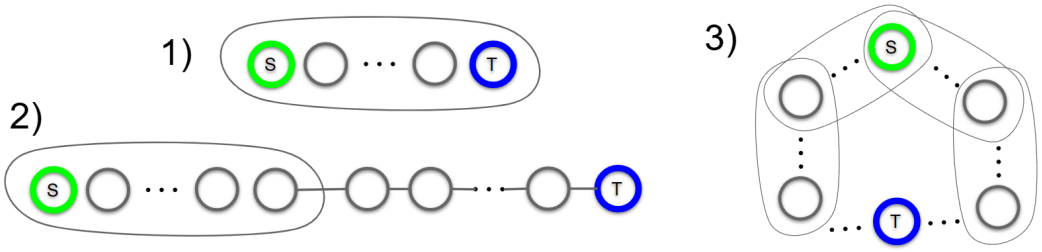

Figure 1: (Best seen in colour) Topological structure of 1) HyperEdgeSingle, 2) HyperEdgePath, and 3) HyperEdgeRing.

4 Tests for Oversquashing in HNNs

In this section, we introduce three novel tests for assessing oversquashing in HNNs summarised pictorially in Figure 1. These tests are inspired by the theory, specifically the Theorem 3.4.

Source, Target Vertices. In all the proposed problems, there are always two specified vertices - one referred to as the source vertex (coloured green in Figure 1) and the other as the target vertex (coloured blue). The input features on each vertex have a dimensionality of $d+2$, where the initial d features are utilised for matching, and the final 2 features represent class information. On the source vertex, the initial d features precisely align with those on the target vertex, and the final 2 features on the source are always [0 0]. In contrast, on the other vertices, the final 2 features represent one of two classes: [0 1] or [1 0], thereby making the task a binary classification task.

Task Details. Formally, we are given a dataset $\mathcal{D}$, consisting of N input-output pairs, where inputs are (hypergraph, vertex features, id of source vertex, id of target vertex) quadruples, and the outputs are binary labels on the target vertices. This is denoted as: $\mathcal{D} = \left\{\Big((\mathcal{H}, X_1, s_1, t_1), Y_1\Big), \cdots, \Big((\mathcal{H}, X_N, s_N, t_N), Y_N\Big)\right\}$. We note that the hypergraph $\mathcal{H}$ is the same for all the input-output instances. We use $X[1:d] \in \mathbb{R}^d$ to represent the first d dimensions of X. Each instance $\Big((\mathcal{H}, X, s, t), Y\Big) \in \mathcal{D}$ is so that

- the n vertex feature vectors, each of dimension $d+2$, are stored in the vertex feature matrix $X \in \mathbb{R}^{n\times d} \times \{0,1\}^{n\times 2}$,
- the first d features of the source are the same as the target, i.e., $X[s][1:d] = X[t][1:d]$, and
- the output label is the same as the final two dimensions of the target, i.e., $Y = X[t][d+1:d+2]$.

In this task, an HNN_θ is trained on the training dataset $\mathcal{D}_{train}$ using parameters θ, while hyperparameters are tuned using the validation dataset $\mathcal{D}_{valid}$. The model's performance is then evaluated on the test dataset $\mathcal{D}_{test}$ where $\mathcal{D} = \mathcal{D}_{train} \cup \mathcal{D}_{valid} \cup \mathcal{D}_{test}$. Representing the model's output as $Z = HNN_\theta(\mathcal{H}, X) \in (0,1)^{n\times 2}$, we calculate the training loss, such as cross-entropy loss, by comparing $Z[s] \in (0,1)^2$ with the corresponding output $Y \in \{[1\ 0] \cup [0\ 1]\}$.

Key Challenges. Firstly, we employ the HNN representation of the source vertex to conduct a binary classification task by predicting the binary label on the target vertex. The HNN needs to implicitly grasp the task of matching the features present on the source to those on the target to "transfer" the label by accessing its features. The inclusion of features on the other vertices of the hypergraph $\mathcal{H}$ introduces complexity and noise to the test, rendering seemingly simple structures unexpectedly challenging.

VC-HNNs vs. VH-HNNs: A Contrast

We offer a brief intuition into the contrasting behaviors of the two HNN categories on the three tests. We use the analogy of electrical network resistance, where message passing corresponds to the flow of current within the network. Subsequently, we provide details of hypergraphs involved in the three tests along with explicit resistance values for structures linked to VC-HNNs and VH-HNNs.

VC-HNNs. In this category, direct pairwise connections between vertices create multiple paths between most pairs of vertices, especially in large hyperedges. This network structure allows messages or 'current' to flow through diverse routes, effectively lowering the resistance. Therefore, we anticipate low resistance distance values for vertex pairs, facilitating message passing with reduced oversquashing as per Theorem 3.4.

VH-HNNs. Within this category, every hyperedge is responsible for 'transporting' messages between its connected vertices. Crucially, the direct link between the source vertex and any of its hyperedges acts as a 'narrow conduit' hindering current flow or message passing from other vertices within the same hyperedge. Hence, we anticipate significantly higher resistance distance values, intensifying the impact of oversquashing, as per Theorem 3.4.

Test 1 - HyperEdgeSingle: HES(n). In this test, the task centers on a single hyperedge of size n where the representation of the source vertex is used to predict the label assigned to the target vertex, both of which are located within the same hyperedge. Typical VC-HNNs (e.g., HGNN) operate on structures with source-target resistance distance $\frac{2}{n}$ whereas typcial VH-HNNs (e.g., UniGCN) with a distance value of 2. Noting the values and later confirming through empirical analysis, even in a hypergraph with a single hyperedge, we see markedly distinct behaviors of VC-HNNs and VH-HNNs.

Test 2 - HyperEdgePath: HEP(n,m). In this test, there exists a hyperedge of size n which contains the source vertex and another vertex that is connected to the target vertex by a path of m hyperedges each of size 2. The resistance distance between source and target vertices is a maximum of $m + \frac{2}{n}$ for structures associated with typical VC-HNNs whereas $2m + 2$ with typical VH-HNNs. Based on the values and Theorem 3.4, we expect this test to be more challenging than the previous HES(n) test and also observe contrasting behaviorus of VC-HNNs and VH-HNNs.

Test 3 - HyperEdgeRing: HER(n,m). Within this test, a ring structure is created by n vertices (assuming an even n), with the source and target vertices positioned at a shortest-path distance of $n/2$. Rather than conventional edges linking the ring's vertices, we utilise hyperedges of size $m \geq 2$, where each hyperedge comprises precisely 2 vertices within the ring. The resistance distance between source and target vertices is a maximum of $\frac{n}{2(m+2)}$ for structures associated with typical VC-HNNs whereas $\frac{n}{2}$ with typical VH-HNNs.

5 SensHNN: Sensitive Hypergraph Neural Network

As established in the previous section and supported by Theorem 3.4, it is clear that typical VC-HNNs exhibit reduced oversquashing effects in comparison to typical VH-HNNs. Drawing from this pivotal observation, this section introduces a novel and effective VC-HNN extension to further mitigate oversquashing in HNNs, leveraging a connection to Theorem 3.1.

Shortest-Path Distance. If $v, u \in V$ are two vertices and $e_1, \cdots, e_k \in E$ are k hyperedges in $\mathcal{H} = (V, E)$, then we say that $P_{vu} : e_1, \cdots, e_k$ is a $v - u$ path of length k, denoted $|P_{vu}| = k$ if the vertices are such that $v \in e_1$, $u \in e_k$, and hyperedges are such that $|e_i \cap e_{i+1}| \geq 1$ for $i = 1, \cdots, k-1$. Assuming $\mathcal{P}_{vu}$ to contain all paths between v and u, the shortest-path distance $S : V \times V \to \mathbb{R}^+$ between v and u is the minimum length of any path between v and u, i.e., $S(v, u) = \min_{P_{vu} \in \mathcal{P}_{vu}} |P_{vu}|$.

Connection to Theory. Theorem 3.1 implies that the upper bound on the sensitivity is influenced by the hypergraph's topology. In particular, when the layer-dependent adjacency matrices are the same, we notice that for message passing between distant vertices to occur, the HNN must traverse intermediate vertices along each connecting path. Under the condition of degree normalisation, the upper bound tends to decrease exponentially with the increase in the shortest-path distance between vertex pairs.

Motivation. In typical VC-HNNs, vertices v and u separated by k hops, i.e., $S(v, u) = k$, initiate their interaction exclusively from the k-th layer onward and do not interact at any preceding layers. Drawing inspiration from our proposed 'long-range' problems, e.g., HyperEdgeRing, we propose that an HNN should engage in direct interaction of distant vertices, in addition to the need to transmit messages through immediate neighbours.

SensHNN Details. Define $\hat{A}_k$ as the adjacency of the graph obtained by connecting vertices exactly k-hops away. Let A_k be the symmetrically normalised adjacency of $\hat{A}_k$, i.e., $A_k(v,u) = \frac{1}{\sqrt{d_v d_u}}$ if $S(v,u) = k$ and 0, otherwise, where $d_v = |e \in E : v \in e|$ is the degree of the vertex v. We propose the following propagation rule:

$$H^{(l+1)} = (1-w)H^{(l)} + w\sum_{k=1}^{l+1} A_k H^{(l+1-k)} W_k^{(l)}, \text{ for } l = 0, \cdots, L-1. \tag{2}$$

Equation 2 ensures that vertices v and u separated by L hops, i.e., $S(v,u) = L$, can directly interact by transforming input vertex features themselves during the final layer. This interaction is facilitated by the presence of the $A_L H^{(0)} W_k^{(l)}$ term. The inclusion of the other terms allows the model to handle tasks that necessitate shorter-range dependencies.

Oversquashing Mitigation Intuition. For the scenario where $w = 1$ in Equation 2, the sensitivity can be bounded, through a proof similar to that of Theorem 3.1 and that in prior work [46], as:

$$\left\| \frac{\partial h_v^{(l)}}{\partial h_u^{(0)}} \right\| \leq (2C_\psi C_\phi)^l \Big(\sum_{k_1+\cdots+k_l=l} \Big(\prod_{k_1,\ldots,k_l} (A_k)_{vu} \Big) \Big).$$

Unlike typical HNNs including VC-HNNs, vertices separated by a distance of l can now interact using message-passing matrices that involve *less than* l factors (including a direct interaction). Using matrices like A_k, which are not powers of the same adjacency matrix, mitigates over-squashing.

Differences with Multi-hop Models and Models with Skip Connections. In our novel model, distant vertices interact directly through transformed input features. Unlike traditional multi-hop models that gather information across multiple hops and vertices, our approach emphasizes direct communication between distant vertices. While skip connections in other models use residues from prior layers for residual learning, our model ensures that interactions between distant vertices occur solely through transformations in deeper hidden layers, not through direct changes in input features.

6 Empirical Analyses

Our experiments aim to address the following research queries (Rs):

- **R1**: What are the contrasting effects in the empirical performance of VC-HNNs and VH-HNNs on the three tests introduced in the paper (HyperEdgeSingle, HyperEdgePath, and HyperEdgeRing)?
- **R2**: How effective is our proposed SensHNN on the three tests in mitigating oversquashing?
- **R3**: Can SensHNN handle heterophilic hypergraphs?
- **R4**: How does SensHNN perform on real-world hypergraph vertex classification datasets?
- **R5**: What is the tradeoff between accuracy and running time on a large real-world dataset?

We use HGNN [30], HyperGCN [31], and HCHA [32] as representative VC-HNNs and UniGCNII [37], AllDeepSets, AllSetTransformers [33], ED-HNN [34] as representative VH-HNNs. Through experiments designed to address questions **R1** and **R2**, we substantiate the connections between the three tests and the theoretical analysis, particularly Theorem 3.4 in Section 4 of the paper. We also validate the connection between the proposed SensHNN and Theorem 3.1 in Section 5 of the paper. Please see Appendix Section A.3 for dataset details, Section A.4 for visulaisation, Section A.5 for complexity analysis, and Section A.6 for details on the hyperparameters.

Model	HES(n)	HEP(n,4)	HES(n,3)
VC-HNNs	$\frac{2}{n}$	$4 + \frac{2}{n}$	$\frac{n}{10}$
VH-HNNs	2	10	$\frac{n}{2}$

Table 1: Resistance distances of source-target pairs in structures typical of VC-HNNs and VH-HNNs.

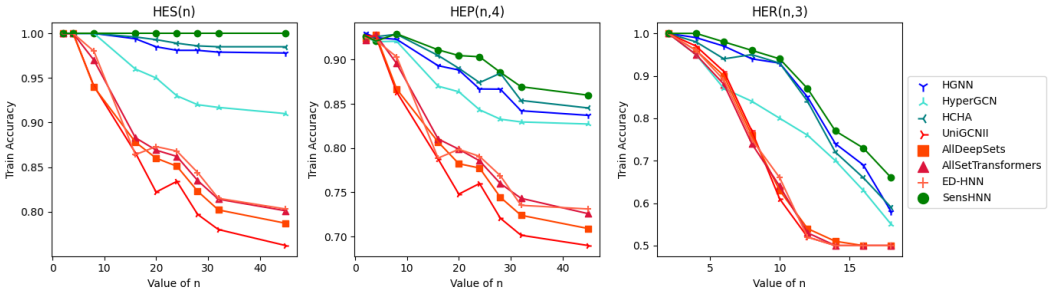

Figure 2: (Best seen in colour) Train Accuracy of different models on the three tests introduced.

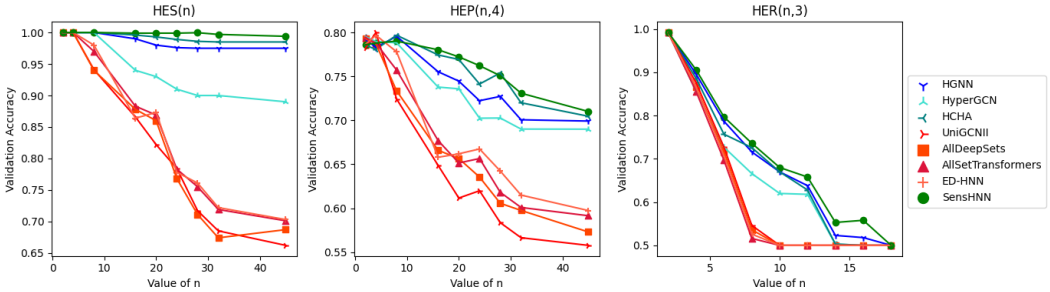

Figure 3: (Best seen in colour) Validation Accuracy of different models on the three tests introduced.

R1: Contrasting Behaviours of VC-HNNs and VH-HNNs. We set $m = 4$ in HyperEdgePath (HEP(n,m)) and $m = 3$ in HyperEdgeRing (HER(n,m)) to highlight the contrast. With these fixed values, the resistance distances for typical VC-HNNs and VH-HNNs are shown in Table 1.

Figures 2 and 3 display the training and validation accuracies, respectively. These figures clearly demonstrate the performance differences between VC-HNNs and VH-HNNs. When cross-referenced with Table 1, it is evident that shorter resistance distances improve information exchange efficiency within an HNN, leading to higher accuracies, while longer distances impede this efficiency.

Increasing m in HyperEdgePath (HEP(n,m)) presents greater challenges for both HNN types due to issues like vanishing gradients with long-range dependencies. However, increasing m in HyperEdgeRing (HER(n,m)) does not change the inherent test challenges, as the test already represents a long-range scenario even for small n values (e.g., 15), where all models achieve a random guessing accuracy of 50%. The test dataset results (not shown) followed the same trends as validation.

R2: Effectiveness of SensHNN. SensHNN achieves the highest training and validation accuracies in all tests, further validating Section 5. In these tests, training accuracy is crucial because the HNN model must retrieve information from a distant vertex. Stronger oversquashing effects reduce the information gathered from the relevant distant vertex, resulting in lower training accuracy.

R3: Experiments on Hypergraphs with Varying Heterophily Levels.

	Heterophily Level						
Name	1	2	3	4	5	6	7
HGNN	98.4	83.7	79.4	74.5	69.5	66.9	63.8
HyperGCN	83.9	69.4	72.9	75.9	70.5	67.3	66.5
HCHA	98.1	81.8	78.3	75.88	74.1	71.1	70.8
AllDeepSets	99.8	86.8	82.4	78.6	77.4	74.9	73.4
AllSetTransformer	99.9	86.5	82.9	79.1	77.7	75.2	73.0
ED-HNN	99.9	91.3	88.4	84.1	80.7	78.8	76.5
SensHNN (Ours)	**100**	**93.7**	**90.1**	**85.7**	**82.4**	**79.4**	**77.6**

Figure 4: Heterophily Experiments.

We investigate hypergraph datasets with different heterophily levels, inspired by prior work [34]. Using a contextual hypergraph stochastic block model, we generate hypergraphs with 5000 nodes (split equally between two classes) and 1000 hyperedges, each containing 15 vertices sampled from class 0. We vary the heterophily level by adjusting the number of vertices from class 0. Table 4 shows the average accuracy across 10 runs, demonstrating the method's effectiveness.

Table 2: Performance on real-world datasets. SensHNN method demonstrates competitive performance.

Name	Cora-CC	Citeseer-CC	Pubmed-CC	Cora-CA	DBLP-CA	Senate	House	Congress
HGNN	79.39 ±1.36	72.45 ±1.16	86.44 ±0.44	82.64 ±1.65	91.03 ±0.20	48.59 ±4.52	61.39 ±2.96	91.26 ±1.15
HyperGCN	78.36 ±2.01	71.01 ±2.21	80.81 ±12.4	79.50 ±2.11	89.42 ±0.16	51.13 ±4.15	69.29 ±2.05	89.67 ±1.22
HCHA	79.14 ±1.02	72.42 ±1.42	86.41 ±0.36	82.55 ±0.97	90.92 ±0.22	48.62 ±4.41	61.36 ±2.53	90.43 ±1.20
HyperND	79.20 ±1.14	72.62 ±1.49	86.68 ±0.43	80.62 ±1.32	90.35 ±0.26	52.82 ±3.20	51.70 ±3.37	74.63 ±3.62
HNHN	76.36 ±1.92	72.64 ±1.57	86.90 ±0.30	77.19 ±1.49	86.78 ±0.29	50.93 ±6.33	67.8 ±2.59	53.35 ±1.45
UniGCNII	78.81 ±1.05	73.05 ±2.21	88.25 ±0.33	83.60 ±1.14	91.69 ±0.19	49.30 ±4.25	67.25 ±2.57	94.81 ±0.81
AllDeepSets	76.88 ±1.80	70.83 ±1.63	88.75 ±0.33	81.97 ±1.50	91.27 ±0.27	48.17 ±5.67	67.82 ±2.40	91.80 ±1.53
AllSetTransformers	78.58 ±1.47	73.08 ±1.20	88.72 ±0.37	83.63 ±1.47	91.53 ±0.23	51.83 ±5.22	69.33 ±2.20	92.16 ±1.05
ED-HNN	80.31 ±1.35	73.70 ±1.38	**89.03 ±0.53**	83.97 ±1.55	**91.90 ±0.19**	64.79 ±5.14	**72.45 ±2.28**	**95.00 ±0.99**
SensHNN (Ours)	**81.19 ±1.52**	**74.67 ±1.28**	87.93 ±0.48	**85.89 ±1.15**	91.20 ±0.28	**68.36 ±4.12**	72.11 ±2.02	91.97 ±1.42

R4: Experiments on Real-World Hypergraphs. Our SensHNN is evaluated on eight real-world datasets, covering a wide range of domains, scales, and heterophily levels. These datasets are commonly used for hypergraph benchmarking (please see the appendix for descriptions).

To ensure rigorous baseline comparisons, we follow the training procedures from a prior work [34]. Data is split into 50% training, 25% validation, and 25% test samples. Each model is run 10 times with different random splits, and results are reported as average accuracy with standard deviation.

Table 2 compares the performance of representative VC-HNNs, VH-HNNs, and our SensHNN. SensHNN outperforms competitors on 4 datasets, demonstrating its effectiveness in handling both long-range tasks and shorter-range hypergraphs.

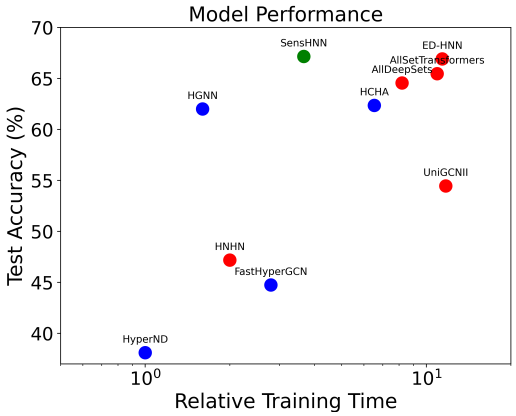

Figure 5: Accuracy - training time tradeoff analysis

R5: Accuracy-Time Tradeoff Analysis. Figure 5 illustrates the relationship between accuracy and training time for various methods applied to the large Walmart dataset [33]. The horizontal axis represents training time relative to the quickest method, HyperND, with time measured in multiples. Our proposed method, SensHNN, demonstrates an impressive balance by achieving high accuracy while maintaining a reasonably fast training time, making it a competitive choice for large-scale data analysis.

7 Concluding Remarks

In this paper, we investigated oversquashing in HNNs. Our theoretical analyses reveal that oversquashing is influenced by the topologuy and resistance distance of vertex pairs induced by message passing operations in HNNs. We introduced three novel tests to assess oversquashing and classified HNNs into VC-HNNs and VH-HNNs, finding that VH-HNNs are particularly prone to oversquashing. We proposed SensHNN, a novel extension to VC-HNNs, as a solution to oversquashing, demonstrating its effectiveness and competitiveness with existing HNNs on real-world hypergraph datasets.

Limitations and Future Research. There are multiple avenues for extending our work, and we outline three specific directions for future research. Firstly, the three tests introduced to assess oversquashing are synthetic in nature. Exploring and isolating oversquashing tendencies within the more nuanced real-world hypergraph datasets could provide valuable insights. Transformer-based models provide an alternative to HNNs for hypergraph learning tasks, though they come with limitations [47]. Lastly, exploring the interplay between oversmoothing and oversquashing presents a compelling avenue for future research.

References

[1] Lingfei Wu, Peng Cui, Jian Pei, and Liang Zhao. *Graph Neural Networks: Foundations, Frontiers, and Applications*. Springer Singapore, 2022. 1, 2

[2] Yao Ma and Jiliang Tang. *Deep Learning on Graphs*. Cambridge University Press, 2020.

[3] William L. Hamilton. Graph representation learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 14(3):1–159, 2020.

[4] Michael M. Bronstein, Joan Bruna, Taco Cohen, and Petar Veličković. Geometric deep learning: Grids, groups, graphs, geodesics, and gauges. *Computing Research Repository (CoRR)*, abs/2104.13478, 2021.

[5] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S. Yu. A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, pages 4–24, 2021.

[6] Si Zhang, Hanghang Tong, Jiejun Xu, and Ross Maciejewski. Graph convolutional networks: a comprehensive review. *Computational Social Networks*, 2019.

[7] Michael M. Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. Geometric deep learning: Going beyond euclidean data. *IEEE Signal Process.*, 34(4):18–42, 2017. 2

[8] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations (ICLR)*, 2017. 2

[9] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *International Conference on Learning Representations (ICLR)*, 2018. 2

[10] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems (NeurIPS) 30*, pages 1024–1034. Curran Associates, Inc., 2017. 2

[11] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *International Conference on Learning Representations (ICLR)*, 2019. 2

[12] Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural message passing for quantum chemistry. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, pages 1263–1272, 2017. 2, 3

[13] Uri Alon and Eran Yahav. On the bottleneck of graph neural networks and its practical implications. In *International Conference on Learning Representations (ICLR)*, 2021. 2, 3

[14] Jake Topping, Francesco Di Giovanni, Benjamin Paul Chamberlain, Xiaowen Dong, and Michael M. Bronstein. Understanding over-squashing and bottlenecks on graphs via curvature. In *International Conference on Learning Representations (ICLR)*, 2022. 2, 3

[15] Rongqin Chen, Shenghui Zhang, Leong Hou U, and Ye Li. Redundancy-free message passing for graph neural networks. In *Advances in Neural Information Processing Systems (NeurIPS) 35*, pages 4316–4327. Curran Associates, Inc., 2022.

[16] Francesco Di Giovanni, Lorenzo Giusti, Federico Barbero, Giulia Luise, Pietro Lio, and Michael M. Bronstein. On over-squashing in message passing neural networks: The impact of width, depth, and topology. In *Proceedings of the 40th International Conference on Machine Learning (ICML)*, pages 7865–7885, 2023. 4

[17] Khang Nguyen, Nong Minh Hieu, Vinh Duc Nguyen, Nhat Ho, Stanley Osher, and Tan Minh Nguyen. Revisiting over-smoothing and over-squashing using ollivier-ricci curvature. In *Proceedings of the 40th International Conference on Machine Learning (ICML)*, pages 25956–25979, 2023. 3

[18] Mitchell Black, Zhengchao Wan, Amir Nayyeri, and Yusu Wang. Understanding oversquashing in GNNs through the lens of effective resistance. In *Proceedings of the 40th International Conference on Machine Learning (ICML)*, pages 2528–2547, 2023. 2

[19] Qingyun Sun, Jianxin Li, Haonan Yuan, Xingcheng Fu, Hao Peng, Cheng Ji, Qian Li, and Philip S. Yu. Position-aware structure learning for graph topology-imbalance by relieving under-reaching and over-squashing. In *Proceedings of the 31st ACM International Conference on Information and Knowledge Management (CIKM)*, page 1848–1857, 2022. 2

[20] Han Gao, Xu Han, Jiaoyang Huang, Jian-Xun Wang, and Liping Liu. PatchGT: Transformer over non-trainable clusters for learning graph representations. In *Proceedings of the First Learning on Graphs Conference (LOG)*, pages 27:1–27:25, 2022.

[21] Andreea Deac, Marc Lackenby, and Petar Veličković. Expander graph propagation. In *Proceedings of the First Learning on Graphs Conference (LOG)*, pages 38:1–38:18, 2022.

[22] Xiaoxin He, Bryan Hooi, Thomas Laurent, Adam Perold, Yann Lecun, and Xavier Bresson. A generalization of ViT/MLP-mixer to graphs. In *Proceedings of the 40th International Conference on Machine Learning (ICML)*, pages 12724–12745, 2023.

[23] Alessio Gravina, Davide Bacciu, and Claudio Gallicchio. Anti-symmetric DGN: a stable architecture for deep graph networks. In *The Eleventh International Conference on Learning Representations (ICLR)*, 2023.

[24] Yang Liu, Chuan Zhou, Shirui Pan, Jia Wu, Zhao Li, Hongyang Chen, and Peng Zhang. Curvdrop: A ricci curvature based approach to prevent graph neural networks from over-smoothing and over-squashing. In *Proceedings of the ACM Web Conference (TheWebConf)*, pages 221–230, 2023. 2

[25] Adrián Arnaiz-Rodríguez, Ahmed Begga, Francisco Escolano, and Nuria M Oliver. Diffwire: Inductive graph rewiring via the lovász bound. In *Proceedings of the First Learning on Graphs Conference (LOG)*, pages 15:1–15:27, 2022. 2

[26] Pradeep Kr. Banerjee, Kedar Karhadkar, Yu Guang Wang, Uri Alon, and Guido Montúfar. Oversquashing in gnns through the lens of information contraction and graph expansion. In *2022 58th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pages 1–8, 2022.

[27] Kedar Karhadkar, Pradeep Kr. Banerjee, and Guido Montufar. FoSR: First-order spectral rewiring for addressing oversquashing in GNNs. In *Eleventh International Conference on Learning Representations (ICLR)*, 2023. 2

[28] Mathilde Papillon, Sophia Sanborn, Mustafa Hajij, and Nina Miolane. Architectures of topological deep learning: A survey on topological neural networks. *CoRR, arXiv:2304.10031*, 2023. 2

[29] Mustafa Hajij, Ghada Zamzmi, Theodore Papamarkou, Nina Miolane, Aldo Guzman-Saenz, Karthikeyan Natesan Ramamurthy, Tolga Birdal, Tamal K. Dey, Soham Mukherjee, Shreyas N. Samaga, Neal Livesay, Robin Walters, Paul Rosen, and Michael T. Schaub. Topological deep learning: Going beyond graph data. *CoRR, arXiv:2206.00606*, 2022. 2

[30] Yifan Feng, Haoxuan You, Zizhao Zhang, Rongrong Ji, and Yue Gao. Hypergraph neural networks. In *Proceedings of the Thirty-Third Conference on Association for the Advancement of Artificial Intelligence (AAAI)*, pages 3558–3565, 2019. 2, 7

[31] Naganand Yadati, Madhav Nimishakavi, Prateek Yadav, Vikram Nitin, Anand Louis, and Partha Talukdar. HyperGCN: A new method of training graph convolutional networks on hypergraphs. In *Advances in Neural Information Processing Systems (NeurIPS) 32*, pages 1509–1520. Curran Associates, Inc., 2019. 2, 3, 7, 14

[32] Song Bai, Feihu Zhang, and Philip H.S. Torr. Hypergraph convolution and hypergraph attention. *Pattern Recognition*, page 107637, 2021. 2, 7

[33] 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 (ICLR)*, 2022. 2, 3, 7, 9, 14

[34] Peihao Wang, Shenghao Yang, Yunyu Liu, Zhangyang Wang, and Pan Li. Equivariant hypergraph diffusion neural operators. In *International Conference on Learning Representations (ICLR)*, 2023. 2, 7, 8, 9, 15

[35] Jinwoo Kim, Saeyoon Oh, Sungjun Cho, and Seunghoon Hong. Equivariant hypergraph neural networks. In *The European Conference on Computer Vision (ECCV)*, pages 86–103, 2022. 2

[36] Yuan Luo. Shine: Subhypergraph inductive neural network. In *Advances in Neural Information Processing Systems (NeurIPS) 35*, pages 18779–18792. Curran Associates, Inc., 2022. 2

[37] Jing Huang and Jie Yang. Unignn: a unified framework for graph and hypergraph neural networks. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence (IJCAI)*, pages 2563–2569, 2021. 3, 7

[38] Dobrik Georgiev, Marc Brockschmidt, and Miltiadis Allamanis. HEAT: hyperedge attention networks. *Trans. Mach. Learn. Res. (TMLR)*, 2022.

[39] 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, 2020. 2

[40] Iulia Duta, Giulia Cassara, Fabrizio Silvestri, and Pietro Lio. Sheaf hypergraph networks. In *Advances in Neural Information Processing Systems (NeurIPS) 36*. Curran Associates, Inc., 2023. 2

[41] Yuxin Wang, Quan Gan, Xipeng Qiu, Xuanjing Huang, and David Wipf. From hypergraph energy functions to hypergraph neural networks. In *Proceedings of the 40th International Conference on Machine Learning (ICML)*, pages 35605–35623, 2023. 2

[42] László Lovász. Random walks on graphs. *Combinatorics, Paul erdos is eighty*, 2(1-46):4, 1993. 4

[43] A. K. Chandra, P. Raghavan, W. L. Ruzzo, and R. Smolensky. The electrical resistance of a graph captures its commute and cover times. In *Proceedings of the Twenty-First Annual ACM Symposium on Theory of Computing*, page 574–586, New York, NY, USA, 1989. Association for Computing Machinery. 4

[44] Kenji Kawaguchi. Deep learning without poor local minima. In *Proceedings of the 30th International Conference on Neural Information Processing Systems (NeurIPS)*, page 586–594. Curran Associates Inc., 2016. 4

[45] Keyulu Xu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken-ichi Kawarabayashi, and Stefanie Jegelka. Representation learning on graphs with jumping knowledge networks. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*, pages 5453–5462, 2018. 4

[46] Benjamin Gutteridge, Xiaowen Dong, Michael M. Bronstein, and Francesco Di Giovanni. DRew: Dynamically rewired message passing with delay. In *Proceedings of the 40th International Conference on Machine Learning (ICML)*, pages 12252–12267, 2023. 7

[47] Haiteng Zhao, Shuming Ma, Dongdong Zhang, Zhi-Hong Deng, and Furu Wei. Are more layers beneficial to graph transformers? In *International Conference on Learning Representations (ICLR)*, 2023. 9

[48] Ralph Abboud, Radoslav Dimitrov, and Ismail Ilkan Ceylan. Shortest path networks for graph property prediction. In *Proceedings of the First Learning on Graphs Conference (LoG)*, pages 5:1–5:25, 2022. 16

A Appendix

A.1 Statement and Proof of Theorem 3.1

Theorem 3.1. *Let $v, u \in V \cup E$, $l \in \mathbb{N}$. If HNN belongs to the VH-HNN category, assume that $\left\|\nabla\psi_V^{(j)}\right\| \leq C_\psi$, $\left\|\nabla\psi_E^{(j)}\right\| \leq C_\psi$, and $\max\{\left\|\nabla\phi_V^{(j)}\right\|, 1\} \leq C_\phi$, $\max\{\left\|\nabla\phi_E^{(j)}\right\|, 1\} \leq C_\phi$, where ∇f denotes the Jacobian of a map f. If HNN belongs to the VC-HNN category, assume that $\left\|\nabla\psi^{(j)}\right\| \leq C_\psi$ and $\max\{\left\|\nabla\phi^{(j)}\right\|, 1\} \leq C_\phi$ for all $j = 0, \ldots, l$. Further, let the layer-dependent symmetrically degree normalised matrices of $\phi^{(j)}$ be $A^{(j)}$. Then the sensitivity satisfies*

$$\left\|\frac{\partial h_v^{(l)}}{\partial h_u^{(0)}}\right\| \leq (2C_\psi C_\phi)^l \sum_{j=0}^{l} (T^{(j)})_{vu}, \textit{where } \; T^{(j)} = \prod_{i=0}^{j} \left((1-w)I + wA^{(i)}\right). \tag{1}$$

Proof. We prove this by induction on the layer r. For the base case of $r = 0$, either $v = u$ or $v \neq u$; in the first case

$$\frac{\partial h_v^{(0)}}{\partial h_u^{(0)}} = \frac{\partial h_v^{(0)}}{\partial h_v^{(0)}} = \mathrm{Id}_{d\times d},$$

and in the second case,

$$\frac{\partial h_v^{(0)}}{\partial h_u^{(0)}} = 0_{d\times d}.$$

Therefore,

$$\left\| \frac{\partial h_v^{(0)}}{\partial h_u^{(0)}} \right\| \leq \max\{\|\mathrm{Id}_{d\times d}\|, \|0_{d\times d}\|\} = 1. \tag{3}$$

Assume that the statement holds for some $k \geq 0$. We assume that the update function is $\psi^{(k)}$ and message function is $\phi^{(k)}$ corresponding to the VC-HNN category. In clear contexts, they can be replaced by $\psi_V^{(k)}$, $\psi_E^{(k)}$ and $\phi_V^{(k)}$, $\phi_E^{(k)}$ respectively of VH-HNNs. We now prove the inductive case of $k+1$.

$$\begin{aligned}
\left\| \frac{\partial h_v^{(k+1)}}{\partial h_u^{(0)}} \right\| &= \left\| \nabla_1 \psi^{(k)} \cdot \frac{\partial h_v^{(k)}}{\partial h_u^{(0)}} + \nabla_2 \psi^{(k)} \cdot \sum_{t\in\mathcal{N}(u)} A_{tu} \cdot \nabla \phi^{(k)} \cdot \frac{\partial h_t^{(k)}}{\partial h_u^{(0)}} \right\| \\
&\leq \left\| \nabla_1 \psi^{(k)} \right\| \cdot \left\| \frac{\partial h_v^{(k)}}{\partial h_u^{(0)}} \right\| + \left\| \nabla_2 \psi^{(k)} \right\| \cdot \sum_{t\in\mathcal{N}(u)} A_{tu} \left\| \nabla \phi^{(k)} \right\| \cdot \left\| \frac{\partial h_t^{(k)}}{\partial h_u^{(0)}} \right\| \\
&\qquad\qquad \text{(as } T_{tu} \geq= 0, \forall\, t,\, u) \\
&\leq C_\psi \cdot \left\| \frac{\partial h_v^{(k)}}{\partial h_u^{(0)}} \right\| + C_\psi C_\phi \cdot \sum_{t\in\mathcal{N}(u)} A_{tu} \cdot \left\| \frac{\partial h_t^{(r)}}{\partial h_u^{(0)}} \right\| \qquad \text{(Theorem hypothesis)} \\
&\leq 2^k (C_\psi C_\phi)^{k+1} \sum_{j=0}^{k} (T^{(j)})_{vu} + 2^k (C_\psi C_\phi)^{k+1} \sum_{j=0}^{k} \sum_{t\in\mathcal{N}(u)} T_{tu} (T^{(k)})_{tv} \quad \text{(induction)} \\
&= 2^k (C_\psi C_\phi)^{k+1} \sum_{j=0}^{k} (T^{(j)})_{vu} + 2^k (C_\psi C_\phi)^{k+1} \sum_{j=1}^{k+1} (T^{(j)})_{vu} \quad \text{(matrix multiplication)} \\
&\leq (2 C_\psi C_\phi)^{k+1} \sum_{j=0}^{k+1} (T^{(j)})_{vu}.
\end{aligned}$$

Here $\nabla\psi^{(k)} = [\nabla_1\psi^{(k)} | \nabla_2\psi^{(k)}]$ and $\nabla\phi^{(k)}$ denote the Jacobian matrices for $\psi^{(k)}$ and $\phi^{(k)}$, respectively. $\nabla_1\psi^{(k)}$ corresponds to partial derivatives w.r.t. the first several arguments in $\psi^{(k)}$ corresponding to $h_v^{(r)}$ and $\nabla_2\psi^{(k)}$ is defined similarly. In the second inequality, we have used the fact for 2-norm that $\|[A|B]\| \geq \max\{\|A\|, \|B\|\}$. In the third inequality, we used the fact that $\beta \geq 1$, and we have that $\alpha \leq \alpha\beta$.

□

A.2 Statement and Proof of Theorem 3.4

Assumption. Assume that all paths in the computation graph of the model are activated with the same probability of success p. When we refer to computing the expectation $\mathbb{E}[\partial \mathbf{h}v^{(l)} / \partial \mathbf{h}u^{(k)}]$, it indicates that we are essentially finding the average across these Bernoulli variables.

Theorem. Consider an HNN with ν the maximal spectral norm of the weight matrices. Let Assumption 3.3 hold. If $\nu \leq 1$, in expectation,

$$\mathsf{O}^{(l)}(v,u) \leq \frac{p}{\nu w} R_{v,u}.$$

Proof. Using the assumption, we can write

$$
\begin{aligned}
&\mathbb{E}\Big[\mathbf{J}_k^{(l)}(v,u)\Big] \\
&= \mathbb{E}\Big[\frac{1}{d_v}\frac{\partial \mathbf{h}_v^{(l)}}{\partial \mathbf{h}_v^{(k)}} - \frac{1}{\sqrt{d_v d_u}}\frac{\partial \mathbf{h}_v^{(l)}}{\partial \mathbf{h}_u^{(k)}} + \frac{1}{d_u}\frac{\partial \mathbf{h}_u^{(l)}}{\partial \mathbf{h}_u^{(k)}} - \frac{1}{\sqrt{d_v d_u}}\frac{\partial \mathbf{h}_u^{(l)}}{\partial \mathbf{h}_v^{(k)}}\Big] \\
&= p\prod_{i=k+1}^{l} W^{(i)}\left(\tfrac{1}{d_v}\left(\Big((1-w)I+wA\Big)^{l-k}\right)_{vv} + \tfrac{1}{d_u}\left(\Big((1-w)I+wA\Big)^{l-k}\right)_{uu} - 2\left(\Big((1-w)I+wA\Big)^{l-k}\right)_{vu}\right) \\
&= p\prod_{i=k+1}^{l} W^{(i)}\langle \frac{1_v}{\sqrt{d_v}} - \frac{1_u}{\sqrt{d_u}}, \Big((1-w)I+wA\Big)^{l-k}\Big(\frac{1_v}{\sqrt{d_v}} - \frac{1_u}{\sqrt{d_u}}\Big)\rangle
\end{aligned}
$$

We rely on the spectral decomposition

$$(1-w)I + wA = \sum_{i=0}^{n-1}\left((-w\lambda_i)\right) e_i e_i^{\top},$$ where e_i are the eigenvectors of the Laplacian, $Le_i = \lambda_i e_i$

We can then bound in **expectation** the obstruction by

$$
\begin{aligned}
O^{(l)}(v,u) &= \sum_{k=0}^{l}\|\mathbf{J}_k^{(l)}(v,u)\| \leq \sum_{k=0}^{l} p\nu^{l-k}\sum_{i=0}^{n-1}(-w\lambda_i))^{l-k}\left(\frac{e_i(v)}{\sqrt{d_v}} - \frac{e_i(u)}{\sqrt{d_u}}\right)^2 \\
&= p\sum_{i=0}^{n-1}\left(\sum_{k=0}^{l}\nu^{l-k}\left(-w\lambda_i)\right)^{l-k}\right)\left(\frac{e_i(v)}{\sqrt{d_v}} - \frac{e_i(u)}{\sqrt{d_u}}\right)^2 \\
&= p\sum_{i=1}^{n-1}\left(\sum_{k=0}^{l}\nu^{l-k}\left(-w\lambda_i)\right)^{l-k}\right)\left(\frac{e_i(v)}{\sqrt{d_v}} - \frac{e_i(u)}{\sqrt{d_u}}\right)^2 \\
&\leq p\sum_{i=1}^{n-1}\frac{1-\left(-\nu(w\lambda_i)\right)^{l+1}}{1-\nu+\nu w\lambda_i}\left(\frac{e_i(v)}{\sqrt{d_v}} - \frac{e_i(u)}{\sqrt{d_u}}\right)^2 \\
&\leq \sum_{i=1}^{n-1}\frac{p}{\nu w\lambda_i}\left(\frac{e_i(v)}{\sqrt{d_v}} - \frac{e_i(u)}{\sqrt{d_u}}\right)^2 \\
&= \frac{p}{\nu w}\sum_{i=1}^{n-1}\frac{1}{\lambda_i}\left(\frac{e_i(v)}{\sqrt{d_v}} - \frac{e_i(u)}{\sqrt{d_u}}\right)^2 \\
&\leq \frac{p}{\nu w}R_{v,u}
\end{aligned}
$$

□

A.3 Dataset Details

Datasets for the Three Tests. We use $N = 10,000$ data instances, i.e., quadruples in the set $\{(\mathcal{H}, X_i, s_i, t_i)\}_{i=1}^{N}$ in all the three tests, viz., HyperEdgeSingle, HyperEdgePath, HyperEdgeRing. The initial d vertex features in each vertex are randomly assigned while the label on the target vertex is randomly assigned one of two binary classes.

Real-World Datasets. Table 3 summarizes the statistics of the real-world hypergraph datasets utilized in this study. We assess our model's performance on well-known datasets, including Cora-CC, Citeseer-CC, Pubmed-CC, Cora-CA, and DBLP-CA [31], as well as House, Senate, and Congress [33]. These datasets represent various network types, such as co-citation, co-authorship, and political affiliation networks, each with distinct vertex and hyperedge features.

In co-citation networks (Cora-CC, Citeseer-CC, Pubmed-CC), hyperedges connect all documents cited by a specific document. In co-authorship networks (Cora-CA, DBLP-CA), hyperedges group all

Table 3: Dataset statistics. CE homophily is the homophily score based on the clique expansion of hypergraphs.

	Cora-CC	Citeseer-CC	Pubmed-CC	Cora-CA	DBLP-CA	Congress	Senate	Walmart	House
# nodes	2708	3312	19717	2708	41302	1718	282	88860	1290
# edges	1579	1079	7963	1072	22363	83105	315	69906	340
# classes	7	6	3	7	6	2	2	11	2
avg. $\lvert e \rvert$	1.748	1.541	1.963	1.929	2.213	8.654	9.531	3.461	8.056
CE Homophily	0.897	0.893	0.952	0.803	0.869	0.555	0.498	0.530	0.509

documents co-authored by a particular author. Vertex features in both types of networks are derived from the bag-of-words of the associated documents, while vertex labels indicate paper classes.

In the House dataset, each vertex represents a member of the US House of Representatives, with hyperedges linking members of the same committee. The Congress and Senate datasets follow similar configurations as previous studies [34]. In the Congress dataset, vertices represent US Congress members, and hyperedges consist of sponsors and co-sponsors of legislative bills. In the Senate dataset, hyperedges include sponsors and co-sponsors of bills introduced solely in the Senate, with each vertex labeled by its political party affiliation.

A.4 Vertex Representation Visualisation

Figure 6 shows the t-SNE visualisations of SensHNN (leftmost) and five competitive baselines. The color code indicates the vertex classes in the Cora-CA dataset. We selected the vertex embeddings that yielded the highest vertex classification scores across all competing methods. It is evident that all methods produce interpretable visualisations, demonstrating distinct inter-class separation. Notably, SensHNN achieves compact, well-separated clusters with the same class labels, exhibiting visually improved separation compared to other methods.

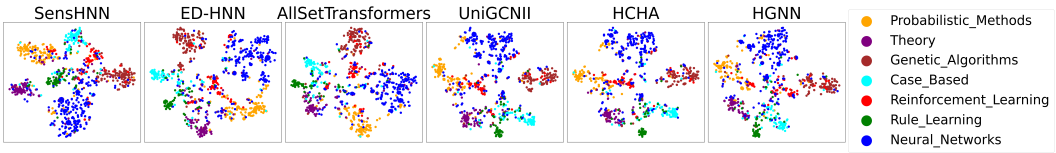

Figure 6: t-SNE visualisation on Cora-CA.

A.5 Complexity Analysis

SensHNN utilises shortest paths, necessitating up to k-hop adjacency information for layer k. Using breadth-first search, pre-computation can be done in worst-case time $\mathcal{O}(|V||E|)$, and the resulting data can be employed for all subsequent runs.

In the worst-case scenario, where l exceeds the graph's maximum diameter, SensHNN aggregates $\mathcal{O}(|V|^2)$ elements, encompassing all possible vertex pairs. Fortunately, the computation of each k-neighborhood aggregation can be performed simultaneously, mitigating this concern in practical applications. Moreover, SensHNN constructs the computational graph incrementally at each layer, ensuring efficient performance in practice.

A.6 Hyperparameters

To ensure fairness in our evaluation, we apply identical training procedures to all models. We adhere closely to the hyperparameter configurations outlined in a previous study for our baseline models [34]. We employ the Adam optimizer with a constant learning rate of 0.001 and no weight decay in training SensHNN. The training process is conducted over 500 epochs across all datasets. To calculate the standard deviation, we conduct experiments across ten distinct data splits and report the resulting variability. We set the input dropout rate at 0.2 and the dropout rate at 0.3. Model sizes and other parameters are determined through a grid search. The layer number is chosen from 1, 2, 4, 6, 8 and hidden dimensions from 96, 128, 256, 512. Our findings show that model size is directly proportional to the dataset scale, with heterophilic data generally requiring deeper architectures.

A.7 Differences with Shortest Path Message Passing

Shortest Path Message Passing Neural Networks (SP-GNNs) [48] aim to overcome the limitations of traditional message passing neural networks (MPNNs), such as over-squashing, by using shortest path neighborhoods instead of relying solely on direct neighborhoods. SP-GNNs propagate information through shortest paths, allowing direct communication between nodes that are multiple hops apart, thereby improving the learning of graph representations and alleviating information loss across layers. The two models, each centered on shortest paths, have subtle differences.

SensHNN employs a *weighted transformation* approach. Specifically, it multiplies the adjacency matrix A_k, which encodes k-hop neighbours, with the hidden state $H^{(l+k-1)}$ from previous layers. After this, a weight matrix $W_k^{(l)}$ is applied to the result, allowing for learnable feature transformation at each hop. This matrix $W_k^{(l)}$ introduces a layer-specific transformation, making it flexible in how the model processes and learns from different neighborhood distances. This weight matrix controls how features from different k-hop neighborhoods are transformed, offering a more complex feature mapping with greater flexibility in learning relationships between nodes at varying distances.

SP-GNN, in contrast, focuses purely on aggregation without involving any weight matrices for feature transformation. It aggregates information directly from the k-hop neighbors A_k using the hidden vectors $H^{(k)}$ at the current layer k. In this model, the hidden states are propagated through layers, but the aggregation process does not involve an additional learnable transformation (like a weight matrix). By avoiding weight matrices, SP-GNN simplifies the message-passing process, focusing purely on neighborhood aggregation based on shortest paths, thus prioritizing efficiency over complexity in feature transformation.

In a nut shell, SensHNN incorporates learnable weights to transform features after multiplying adjacency matrices, providing greater flexibility in adjusting how node features evolve across layers. SP-GNN avoids this complexity by directly aggregating hidden states from shortest path neighborhoods without applying extra transformations, making the model simpler and potentially faster.