# Exploring Intrinsic Structures of Hyperedges as Point Clouds

**Lige Zhang**
Duke Kunshan University
lz245@duke.edu

**Dongmian Zou**
Duke Kunshan University
dongmian.zou@duke.edu

## Abstract

Hypergraph neural networks (HNNs) provide a powerful framework for modeling high-order relationships in complex data. However, existing approaches often overlook the intrinsic patterns carried by hyperedges. Some methods simplify a hyperedge as a fully connected subgraph or treat it as an intermediate node-like entity, which limits the expressivity of the resulting models and neglects the potentially rich information of hyperedges. In this work, we offer a new perspective for hypergraph modeling by modeling a hyperedge as a point cloud with learnable features. Building on this view, we present a novel Hypergraph Kernel Network (HypKN) framework for hypergraph representation learning, which fully exploits the intrinsic hypergraph structure. The core component in HypKN is a Kernel Attention Message Passing (KAMP) module, which mimics the classical convolution operation defined for non-Euclidean data structures and enjoys provable stability results. We evaluate HypKN on ten real-world and synthetic hypergraph datasets for node classification, where it consistently outperforms classical HNN baselines and achieves state-of-the-art performance on several benchmarks.

## 1 Introduction

Most existing hypergraph neural networks (HNNs) fall into two broad categories: Vertex-Centric HNNs (VC-HNNs), Vertex-Hyperedge HNNs (VH-HNNs) [1], which is reviewed in detail in Appendix A. These models are designed specifically for hypergraphs and include foundational works such as [2] and [3]. In contrast, a separate line of research focuses on Unified HNNs (U-HNNs), which are designed to operate on both graphs and hypergraphs, with recent examples including UniGEncoder [4]. In this work, we focus primarily on pure hypergraph methods, with a goal of finding natural and intrinsic representations for propagating the hypergraph structures.

While prior approaches have led to important advances in hypergraph modeling, they are grounded in different underlying assumptions. VC-HNNs typically represent a hyperedge as a fully connected subgraph, usually referred to as clique expansion [5, 6]. However, this formulation creates a bias about the topology of nodes connected by the hyperedge. VH-HNN, on the other hand, treats each hyperedge as a node-like entity—an approach commonly known as star expansion [7]. Introduced in HGNN [2] and later extended in HGNN+ [8], this formulation enables a message-passing mechanism where nodes communicate with their associated hyperedges, which in turn aggregate and return information back to the nodes. Building on this foundation, a number of powerful variants have been developed, such as Hypergraph Attention [9], AllSetTransformer [10] and Equivariant Diffusion HNN [11], achieving impressive results across standard hypergraph benchmarks. It is also possible to consider a spectral view for building VH-HNNs [12, 13]. Although VH-HNNs capture the intrinsic information flow over hypergraph topology, existing implementations of node-to-hyperedge message passing, by either simple averaging or attention-based averaging, is still over-simplification of the underlying pattern. This limitation becomes particularly evident in heterophilic hypergraph-related tasks, where existing VH-HNN-based methods may struggle to learn meaningful representations.

Intuitively, hyperedges can be naturally interpreted as unordered sets of nodes, each representing a unique combination of elements with potentially rich internal structure. This perspective motivates us to draw inspiration from point cloud processing, where similar unordered sets are common. In particular, we build on the idea of point kernels for convolution, particularly KPConv [14], which learns spatially-aware kernels over 3D point clouds in a way that generalizes classical convolution [15]. We adapt this concept to hypergraphs by proposing a kernel that can be transported to different hyperedges. This kernel then serves as a learnable aggregator that passes messages from nodes to their corresponding hyperedges. Based on this idea, we proposed a novel framework, called the Hypergraph Kernel Network (HypKN), which provides a novel perspective on HNNs. We outline our key contributions as follows:

- We propose interpreting hyperedges as unordered point sets, and propose using kernel points to learn the latent patterns contained in the hyperedges. Specifically, we develop a kernel attention message passing (KAMP) module for hypergraph nueral networks (HNN). We prove that KAMP is a stable feature extractor.
- We implement a novel Hypergraph Kernel Network (HypKN), which achieves state-of-the-art performance on several benchmark hypergraph datasets. In particular, HypKN is good for heterophilic hypergraph learning.

## 2 Method: Hypergraph Kernel Networks

### 2.1 Kernel Points

We propose a variant of kernel points, which achieves the geometry-aware feature extraction for hypergraphs. It is built upon the pioneering work for 3D point clouds [14] and its non-Euclidean variant [16]. The kernel configuration adheres to their principles, which require: *Dispersion*, that kernel points should maintain a large pairwise distance to capture diverse local patterns, and *Central proximity*, that kernel points should remain within a bounded distance from the center to preserve locality. This design enhances the kernel's capability to perceive structural variations across hyperedges by distributing kernel points over distinct sub-regions.

In our framework, kernel points are initialized at the origin and translated to align with individual hyperedges, facilitating node-to-edge message propagation. Let $\{\mathbf{z}_k\}_{k=1}^K$ denote the kernel points. We first fix $\mathbf{z}_1 = \mathbf{0}$ as the origin, and then obtain the remaining kernel points by minimizing the following loss function:

$$\mathcal{L}(\{\mathbf{z}_k\}_{k=1}^K) = \sum_{k=1}^{K} \sum_{l \neq k} \frac{1}{d(\mathbf{z}_l, \mathbf{z}_k)} + \sum_{k=1}^{K} d(\mathbf{0}, \mathbf{z}_k), \tag{1}$$

where $d(\cdot,\cdot)$ denotes the Euclidean distance. A more detailed discussion on kernel points, including its geometric shape and properties, can be found in Appendix B.1

### 2.2 KAMP: Kernel Attention Message Passing

We propose a **K**ernel **A**ttention **M**essage **P**assing (KAMP) to extract information about hyperedges from nodes contained within. This process involves three sequential steps. First, the kernel is translated from its initialized position, the origin, to the geometric centroids of the hyperedges. Second, at each hyperedge, we perform attention-based aggregation between the kernel points and transformed node features within the hyperedge group. This step enables the kernel to dynamically perceive distinct node characteristics within the hyperedge group. Finally, we take a simple aggregation of perceptions on individual nodes, and we utilize the resulting integrated signal to iteratively refine the hyperedge representation. We illustrate each step in detail in the following contents.

**Step 1 (kernel translation).** Let the kernel points $\{\tilde{\mathbf{z}}_k\}_{k=1}^K$ be initialized and centered at the origin. We first need to translate them to an hyperedge $e \in \mathcal{E}$ via following translation operation:

$$\mathbf{z}_{ek} = \mathrm{Trans}_{\mathbf{0}\to\overline{\mathbf{x}}_e}(\tilde{\mathbf{z}}_k) = \tilde{\mathbf{z}}_k + \overline{\mathbf{x}}_e, \quad k = 1, \cdots, K, \tag{2}$$

where $\overline{\mathbf{x}}_e$ represents the centroid of node features $\{\mathbf{x}_{ev}\}_{v=1}^{v=|d_e|}$ for hyperedge $e$, which can be directly calculated by taking average.

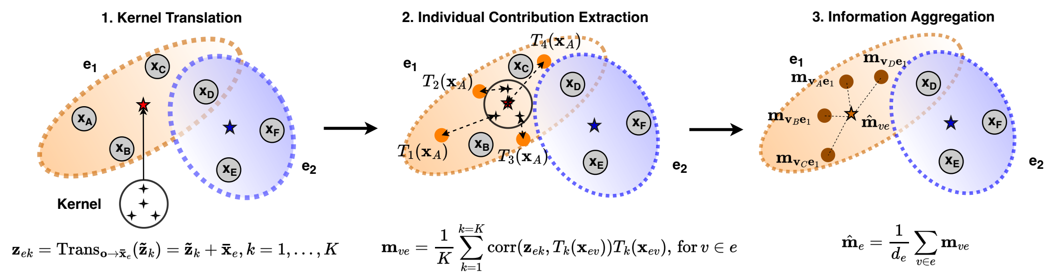
1. Kernel Translation
2. Individual Contribution Extraction
3. Information Aggregation
e1
e2
Kernel
$\mathbf{z}_{ek} = \text{Trans}_{\mathbf{o}\to\bar{\mathbf{x}}_e}(\tilde{\mathbf{z}}_k) = \tilde{\mathbf{z}}_k + \bar{\mathbf{x}}_e, k = 1, \dots, K$
$\mathbf{m}_{ve} = \frac{1}{K}\sum_{k=1}^{k=K} \text{corr}(\mathbf{z}_{ek}, T_k(\mathbf{x}_{ev}))T_k(\mathbf{x}_{ev})$, for $v \in e$
$\hat{\mathbf{m}}_e = \frac{1}{d_e}\sum_{v\in e}\mathbf{m}_{ve}$

**Figure 1:** A visualization that illustrates our KAMP mechanism.

**Step 2 (individual contribution extraction).** We denote nodes within an hyperedge $e \in \mathcal{E}$ as a set $\{\mathbf{x}_{ev}\}_{v=1}^{v=|d_e|}$. Before extracting features, we perform $K$ independent learnable linear transformations for each node $\mathbf{x}_{ev}$:

$$\mathbf{x}_{evk} = T_k(\mathbf{x}_{ev}), \quad k = 1, \dots, K. \tag{3}$$

Then we calculate the correlation between kernel points $\mathbf{z}_{ek}$ and the transformed node embedding $\mathbf{x}_{evk}$. We consider the following alternatives: either adopting the self-attention regime of a transformer [17], or directly using the distance between kernel points and nodes. Specifically:

- **Attention-based**:

$$a_k = \frac{\langle \mathbf{z}_{ek}, \mathbf{x}_{evk} \rangle}{\sqrt{D_{\mathbf{z}_{ek}}}}; \ \text{corr}(\mathbf{z}_{ek}, \mathbf{x}_{evk}) = \frac{\exp(a_k)}{\sum_{k=1}^{K}\exp(a_k)}. \tag{4}$$

  where $D_{\mathbf{z}_{ek}}$ represents the embedding dimension.

- **Distance-based**:

$$d_k = \|\mathbf{z}_{ek} - \mathbf{x}_{evk}\|; \ \ \text{corr}(\mathbf{z}_{ek}, \mathbf{x}_{evk}) = \frac{\exp(-d_k^2)}{\sum_{k=1}^{K}\exp(-d_j^2)}. \tag{5}$$

Finally, we use the above correlation weights to aggregate the transformed node features and produce the the message that a particular node $v$ passes to the corresponding hyperedge $e$, of which we have placed a kernel on top. Specifically, the message from $v$ to $e$ is:

$$\mathbf{m}_{ve} = \frac{1}{K}\sum_{k=1}^{K} \text{corr}(\mathbf{z}_{ek}, \mathbf{x}_{evk})\mathbf{x}_{evk}, \quad v \in e. \tag{6}$$

**Step 3 (information aggregation).** In the final step, we aggregate all the messages that an edge $e$ receives from its resident nodes $\{v \in e\}$. A simple implementation will be averaging the messages from individual nodes.

$$\hat{\mathbf{m}}_e = \frac{1}{d_e}\sum_{v\in e}\mathbf{m}_{ve}. \tag{7}$$

**KAMP summary.** Altogether, we can represent the KAMP module as a function $\text{KAMP} : \mathbb{R}^{N\times C} \to \mathbb{R}^{E\times C}$, which is summarized as the following:

$$\text{KAMP}(\{x_v\}_{v\in e}) = \frac{1}{d_e K}\sum_{v\in e}\sum_{k=1}^{K} \text{corr}(\mathbf{z}_{ek}, \mathbf{x}_{vk})\mathbf{x}_{vk}, \tag{8}$$

where KAMP takes a set of nodes $\{x_v\}_{v\in e}$ on the same hyperedge to generate messages to be passed to the corresponding hyperedge $e$. The procedure of KAMP is also visualized in Figure 1.

KAMP provides a specialized implementation of the aggregation function $\phi_V$, distinct from existing approaches. Such a design enables KAMP to extract richer patterns from the node embeddings instead of simply relying on graph topology. This significantly improves expressivity in tasks that models high-order relations. This component can also be explained as a Mixture-of-Experts (MoE) [18–20] module. Each kernel point can be viewed as an expert, where the associated transformations act as distinct expert functions. The node-kernel correlation acts as the routing score, like [21]. The final aggregation resembles the weighted mixture of outputs of individual expert.

**Table 1:** Hypergraph node classification results, presented as mean accuracy (%) ± standard deviation. Best result is highlighted in bold and results within one standard deviation of the best are underlined.

| Method | Cora-CC | Citeseer-CC | Pubmed-CC | Cora-CA | ModelNet40 | NTU2012 | Zoo | 20Newsgroups | House | Senate | R |
|---|---|---|---|---|---|---|---|---|---|---|---|
| HGNN | $\underline{79.39 \pm 1.36}$ | $72.45 \pm 1.16$ | $86.44 \pm 0.44$ | $82.64 \pm 1.65$ | $95.44 \pm 0.33$ | $87.72 \pm 1.35$ | $95.50 \pm 4.58$ | $80.33 \pm 0.42$ | $61.39 \pm 2.96$ | $48.59 \pm 4.52$ | 9 |
| HCHA | $79.14 \pm 1.02$ | $72.42 \pm 1.42$ | $86.41 \pm 0.36$ | $82.55 \pm 0.97$ | $94.48 \pm 0.28$ | $87.48 \pm 1.87$ | $93.65 \pm 6.15$ | $80.33 \pm 0.80$ | $61.36 \pm 2.53$ | $48.62 \pm 4.41$ | 12 |
| HNHN | $76.36 \pm 1.92$ | $72.64 \pm 1.57$ | $86.90 \pm 0.30$ | $77.19 \pm 1.49$ | $97.84 \pm 0.25$ | $\underline{89.11 \pm 1.44}$ | $93.59 \pm 5.88$ | $81.35 \pm 0.61$ | $67.80 \pm 2.59$ | $50.93 \pm 6.33$ | 10 |
| AllDeepSets | $76.88 \pm 1.80$ | $70.83 \pm 1.63$ | $88.75 \pm 0.33$ | $81.97 \pm 1.50$ | $96.98 \pm 0.26$ | $\underline{88.09 \pm 1.52}$ | $95.39 \pm 4.77$ | $81.06 \pm 0.54$ | $67.82 \pm 2.40$ | $48.17 \pm 5.67$ | 8 |
| AllSetTransformer | $78.58 \pm 1.47$ | $73.08 \pm 1.20$ | $88.72 \pm 0.37$ | $\underline{83.63 \pm 1.47}$ | $98.20 \pm 0.20$ | $88.69 \pm 1.24$ | $\underline{97.50 \pm 3.59}$ | $81.38 \pm 0.58$ | $69.33 \pm 2.20$ | $51.83 \pm 5.22$ | 4 |
| ED-HNN | $\underline{80.31 \pm 1.35}$ | $\underline{73.70 \pm 1.38}$ | $\mathbf{89.56 \pm 0.62}$ | $\underline{83.97 \pm 1.55}$ | $\underline{98.35 \pm 0.20}$ | $88.07 \pm 1.28$ | $95.77 \pm 3.37$ | $\underline{81.90 \pm 0.55}$ | $72.45 \pm 2.28$ | $64.79 \pm 5.14$ | 2 |
| HyperGCN | $78.45 \pm 1.26$ | $71.28 \pm 0.82$ | $82.84 \pm 8.67$ | $79.48 \pm 2.08$ | $75.89 \pm 5.26$ | $56.36 \pm 4.86$ | $85.38 \pm 6.23$ | $81.05 \pm 0.59$ | $48.32 \pm 2.93$ | $42.45 \pm 3.67$ | 14 |
| LE-GCN | $77.34 \pm 1.10$ | $73.41 \pm 1.15$ | $88.53 \pm 0.48$ | $76.60 \pm 0.48$ | $96.68 \pm 0.16$ | $\underline{89.16 \pm 1.13}$ | $95.00 \pm 4.81$ | $\underline{81.84 \pm 0.34}$ | $\mathbf{78.39 \pm 1.64}$ | $\underline{80.70 \pm 5.67}$ | 5 |
| MLP | $75.17 \pm 1.21$ | $72.67 \pm 1.56$ | $87.47 \pm 0.51$ | $74.31 \pm 1.89$ | $96.14 \pm 0.36$ | $\underline{85.52 \pm 1.49}$ | $87.18 \pm 4.44$ | $81.42 \pm 0.49$ | $\underline{77.21 \pm 3.25}$ | $\underline{78.87 \pm 3.11}$ | 11 |
| UniGCNII | $78.81 \pm 1.05$ | $73.05 \pm 2.21$ | $88.25 \pm 0.40$ | $\underline{83.60 \pm 1.14}$ | $98.07 \pm 0.23$ | $\underline{89.30 \pm 1.33}$ | $93.65 \pm 4.37$ | $81.12 \pm 0.67$ | $67.25 \pm 2.57$ | $49.30 \pm 4.25$ | 7 |
| UniGEncoder | $\mathbf{80.49 \pm 1.30}$ | $\mathbf{74.49 \pm 1.02}$ | $88.71 \pm 0.54$ | $\mathbf{84.17 \pm 1.19}$ | $\underline{98.41 \pm 0.23}$ | $\underline{89.11 \pm 0.85}$ | $95.77 \pm 4.23$ | $\underline{81.51 \pm 0.39}$ | $\underline{77.12 \pm 2.80}$ | $\underline{78.59 \pm 3.97}$ | 3 |
| **HypKN (Ours)** | $\underline{79.27 \pm 1.38}$ | $\underline{74.01 \pm 1.49}$ | $88.87 \pm 0.46$ | $82.71 \pm 1.72$ | $\mathbf{98.44 \pm 0.21}$ | $\mathbf{90.12 \pm 1.14}$ | $96.54 \pm 2.84$ | $\mathbf{81.95 \pm 0.43}$ | $\underline{77.64 \pm 2.75}$ | $\mathbf{80.85 \pm 4.31}$ | 1 |
| **Ablation1:** | | | | | | | | | | | |
| HKN-base (HGNN++) | $77.49 \pm 1.44$ | $72.02 \pm 1.45$ | $87.85 \pm 0.34$ | $81.46 \pm 1.47$ | $97.98 \pm 0.27$ | $\underline{89.38 \pm 0.89}$ | $93.85 \pm 3.72$ | $\underline{81.52 \pm 0.27}$ | $\underline{77.31 \pm 2.75}$ | $\underline{79.72 \pm 2.59}$ | 6 |
| HKN w/o Self_State | $54.51 \pm 1.66$ | $39.67 \pm 2.79$ | $48.06 \pm 0.65$ | $71.49 \pm 1.75$ | $92.14 \pm 0.55$ | $85.68 \pm 1.47$ | $\mathbf{98.08 \pm 2.03}$ | $80.75 \pm 0.31$ | $56.78 \pm 1.93$ | $54.65 \pm 5.67$ | 13 |

## 2.3 HypKN: Hypergraph Kernel Network

The proposed model is built upon a previous baseline HGNN+ [8], which is a simple implementation of the framework of (9). First, we add a self-state residual sum to obtain a new baseline model, namely HGNN++. Then, we replace the naive message generator $\phi_V$ in (9) with the proposed KAMP module, which we refer to as the Hypergraph Kernel Network (HypKN). The overall architecture can be summarized by the following update steps:

$$\mathbf{m}_e^{(l+1)} = \omega \mathbf{m}_e^{(l)} + (1-\omega)\mathrm{KAMP}(\mathbf{x}_v^{(l)}), \qquad \text{(node to hyperedge)}$$

$$\mathbf{x}_v^{(l+1)} = \tilde{\omega} \mathbf{x}_v^{(l)} + (1-\tilde{\omega}) \frac{1}{D_v} \sum_{e \in \mathcal{N}(v)} \mathbf{m}_e^{(l+1)}, \qquad \text{(hyperedge to node)}$$

where $\omega, \tilde{\omega} \in [0,1]$ are the weights for fusion. To initialize, $\mathbf{m}_e^{(0)}$ is taken to be the centroid of initial node attributes $\mathbf{x}_v^{(0)}$ for $v \in e$. Although HypKN is primarily designed for hypergraph node classification, it actually focuses on producing meaningful representations for hyperedges.

## 3 Experiments

We evaluate the performance of HypKN on multiple hypergraph benchmark datasets, with a primary focus on hypergraph node classification. All experiments were executed on a GPU server with NVIDIA GeForce RTX 3090 GPUs (24G memory). We perform each experiment using only a single GPU. We use cross-entropy loss with L2-regularization to train each model. We provide a detailed description for the benchmark datasets and baseline models in Appendix C.

Table 1 shows HypKN's performance on ten benchmark datasets, where the proposed model has achieved state-of-the-art performance on four of them. In particular, HypKN has achieved significant improvement on heterophilic hypergraph datasets over existing classical methods. We also assign a cumulative rank to different models based on their performance across multiple datasets, which follows the convention in this field [10, 13].Our proposed HypKN is ranked the first, indicating the overall best performance. Futher, ablation studies can be found in Appendix D. Specific hyperparameter searching and settings are illustrated in detail in Appendix F.

## 4 Conclusion

In this work, we introduced a novel perspective on hypergraph neural network design by treating hyperedges as point clouds with latent geometric structures, rather than as intermediate node-like entities solely for message passing, as is common in prior approaches. Based on this insight, we developed the KAMP module, which forms the core of our proposed HypKN architecture. HypKN consistently outperforms a range of classical and strong baseline HNN models across multiple benchmarks. Beyond empirical results, we also provided a theoretical analysis that proves the stability of our model. One limitation of this work is the use of a fixed kernel applied uniformly across all hyperedges.

## 5 Future Work

While our current implementation employs a fixed kernel uniformly applied across all hyperedges, such rigidity may fail to capture the diverse geometric structures inherent in complex hypergraphs. In future work, one potential direction is to explore adaptive kernel deformation strategies that dynamically reshape kernels during translation to better align with the unique geometry of individual hyperedges. Additionally, extending KAMP's evaluation to broader hypergraph learning tasks, particularly hyperedge prediction as established in [22, 23], would provide crucial insights into its practical utility.

## Author Contributions

**Lige Zhang**: Conceptualization; Methodology; Investigation; Formal analysis; Software; Writing - Original Draft **Dongmian Zou**: Conceptualization; Methodology; Investigation; Formal analysis; Writing - Original Draft; Supervision.

## Acknowledgements

This work is partially supported by the DKU Innovation Incubator (DII) program at Duke Kunshan University.

## References

[1] Naganand Yadati. Oversquashing in hypergraph neural networks. In *The Third Learning on Graphs Conference*, 2025. 1, 7

[2] 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. 1, 14, 15

[3] 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. 1, 14, 15

[4] Minhao Zou, Zhongxue Gan, Yutong Wang, Junheng Zhang, Dongyan Sui, Chun Guan, and Siyang Leng. Unig-encoder: A universal feature encoder for graph and hypergraph node classification. *Pattern Recognition*, 147:110115, 2024. 1, 15

[5] 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. 1, 7

[6] Dengyong Zhou, Jiayuan Huang, and Bernhard Schölkopf. Learning with hypergraphs: Clustering, classification, and embedding. *Advances in neural information processing systems*, 19, 2006. 1, 7

[7] Jason Y Zien, Martine DF Schlag, and Pak K Chan. Multilevel spectral hypergraph partitioning with arbitrary vertex sizes. *IEEE Transactions on computer-aided design of integrated circuits and systems*, 18(9):1389–1399, 1999. 1

[8] Yue Gao, Yifan Feng, Shuyi Ji, and Rongrong Ji. Hgnn+: General hypergraph neural networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(3):3181–3199, 2022. 1, 4

[9] Song Bai, Feihu Zhang, and Philip HS Torr. Hypergraph convolution and hypergraph attention. *Pattern Recognition*, 110:107637, 2021. 1, 15

[10] Eli Chien, Chao Pan, Jianhao Peng, and Olgica Milenkovic. You are allset: A multiset function framework for hypergraph neural networks. *arXiv preprint arXiv:2106.13264*, 2021. 1, 4, 14, 15

[11] Peihao Wang, Shenghao Yang, Yunyu Liu, Zhangyang Wang, and Pan Li. Equivariant hypergraph diffusion neural operators. *arXiv preprint arXiv:2207.06680*, 2022. 1, 15

[12] Ming Li, Yujie Fang, Yi Wang, Han Feng, Yongchun Gu, Lu Bai, and Pietro Liò. Deep hypergraph neural networks with tight framelets. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 18385–18392, 2025. 1

[13] Ming Li, Yongchun Gu, Yi Wang, Yujie Fang, Lu Bai, Xiaosheng Zhuang, and Pietro Lio. When hypergraph meets heterophily: New benchmark datasets and baseline. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 18377–18384, 2025. 1, 4

[14] Hugues Thomas, Charles R Qi, Jean-Emmanuel Deschaud, Beatriz Marcotegui, François Goulette, and Leonidas J Guibas. Kpconv: Flexible and deformable convolution for point clouds. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 6411–6420, 2019. 2, 8, 9

[15] Yann LeCun, Yoshua Bengio, et al. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10):1995, 1995. 2, 8

[16] Eric Qu, Lige Zhang, Habib Debaya, Yue Wu, and Dongmian Zou. Hyperbolic kernel convolution: A generic framework. In *The Third Learning on Graphs Conference*, 2025. 2

[17] A Vaswani. Attention is all you need. *Advances in Neural Information Processing Systems*, 2017. 3

[18] Robert A Jacobs, Michael I Jordan, Steven J Nowlan, and Geoffrey E Hinton. Adaptive mixtures of local experts. *Neural computation*, 3(1):79–87, 1991. 3

[19] Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, Dehao Chen, Orhan Firat, Yanping Huang, Maxim Krikun, Noam Shazeer, and Zhifeng Chen. Gshard: Scaling giant models with conditional computation and automatic sharding. *arXiv preprint arXiv:2006.16668*, 2020.

[20] William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23 (120):1–39, 2022. 3

[21] Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*, 2024. 3, 17

[22] 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. 5

[23] Geon Lee, Soo Yong Lee, and Kijung Shin. Villain: Self-supervised learning on homogeneous hypergraphs without features via virtual label propagation. In *Proceedings of the ACM Web Conference 2024*, pages 594–605, 2024. 5

[24] R Dharmarajan and K Kannan. Hyper paths and hyper cycles. *Int. J. Pure Appl. Math*, 98(3): 309–312, 2015. 7

[25] Karel Devriendt and Piet Van Mieghem. The simplex geometry of graphs. *Journal of Complex Networks*, 7(4):469–490, 2019. 8

[26] Mikhail Nevskii. On properties of a regular simplex inscribed into a ball. *arXiv preprint arXiv:2105.07700*, 2021. 8

[27] Bolin Gao and Lacra Pavel. On the properties of the softmax function with application in game theory and reinforcement learning. *arXiv preprint arXiv:1704.00805*, 2017. 12

[28] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1912–1920, 2015. 14

[29] Ding-Yun Chen, Xiao-Pei Tian, Yu-Te Shen, and Ming Ouhyoung. On visual similarity based 3d model retrieval. In *Computer graphics forum*, volume 22, pages 223–232. Wiley Online Library, 2003. 14

[30] Yifan Feng, Zizhao Zhang, Xibin Zhao, Rongrong Ji, and Yue Gao. Gvcnn: Group-view convolutional neural networks for 3d shape recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 264–272, 2018. 14

[31] Hang Su, Subhransu Maji, Evangelos Kalogerakis, and Erik Learned-Miller. Multi-view convolutional neural networks for 3d shape recognition. In *Proceedings of the IEEE international conference on computer vision*, pages 945–953, 2015. 14

[32] Chaoqi Yang, Ruijie Wang, Shuochao Yao, and Tarek Abdelzaher. Semi-supervised hypergraph node classification on hypergraph line expansion. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, pages 2352–2361, 2022. 14, 15

[33] Dheeru Dua, Casey Graff, et al. Uci machine learning repository, 2017. *URL http://archive. ics. uci. edu/ml*, 7(1):62, 2017. 14

[34] James H Fowler. Legislative cosponsorship networks in the us house and senate. *Social networks*, 28(4):454–465, 2006. 14

[35] Yihe Dong, Will Sawin, and Yoshua Bengio. Hnhn: Hypergraph networks with hyperedge neurons. *arXiv preprint arXiv:2006.12278*, 2020. 15

[36] Jing Huang and Jie Yang. Unignn: a unified framework for graph and hypergraph neural networks. *arXiv preprint arXiv:2105.00956*, 2021. 15

[37] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 2623–2631, 2019. 15, 18

## A Notions and Preliminaries

### A.1 Notation for Hypergraphs

A hypergraph generalizes a classical graph by allowing a single edge to connect more than two nodes. Let $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{X})$ denote an *attributed hypergraph*, where $\mathcal{V}$ represents the set of nodes, $\mathcal{E}$ represents the set of hyperedges where each hyperedge $e \in \mathcal{E}$ connects (and is represented by) a subset of $\mathcal{V}$, and $\mathbf{X} = [\ldots, \mathbf{x}^\top, \ldots] \in \mathbb{R}^{N \times C}$ denotes node attributes. Since each hyperedge could connect more than two nodes, a hypergraph utilizes an incidence matrix $\mathbf{H} \in \{0,1\}^{|V| \times |E|}$ to represents its topological structure rather than a classical adjacency matrix, where $H_{ij} = 1$ if and only if node $i$ is contained in hyperedge $j$. For a node $v$, we use $\mathcal{N}(v) := \{e \in \mathcal{E} : v \in e\}$ to denote the set of hyperedges that contain $v$. Let $d_v = |\mathcal{N}(v)|$ denote the node degree for any $v \in \mathcal{V}$, and $d_e = |\{v : v \in e\}|$ denote the edge degree for any $e \in \mathcal{E}$. We define node degree matrix $\mathbf{D}_v$ and edge degree matrix $\mathbf{D}_e$ as diagonal matrices whose entries are given by node degrees and edge degrees, respectively.

### A.2 Message Passing on Hypergraphs

Hypergraph neural networks primarily adopt two types of message-passing regimes: Vertex-Centric HNNs (VC-HNNs) and Vertex-Hyperedge HNNs (VH-HNNs) [1]. Despite their differences, both frameworks rely on converting hypergraphs into conventional graphs, through either clique expansion or star expansion [5, 6]. These approaches can be summarized as follows:

- **VC-HNNs** treat hyperedges as connected subgraphs. In each layer, the node features are updated according to passing messages from vertices sharing the same hyperedge. That is, for the $l$-th layer,

$$h_v^{(l+1)} = \psi^{(l)}\left(h_v^{(l)}, \phi^{(l)}\left(\{h_u^{(l)} : \{v, u\} \subseteq E\}\right)\right),$$

  where $\phi^{(l)}$ is a permutation-invariant aggregation function, and $\psi^{(l)}$ combines the self-state value $h_v^{(l)}$ with messages received from neighbors.

- **VH-HNNs** treat hyperedges as node-like entities and design message passing according to information flow between nodes and edges along hyperpaths $\mathcal{V} \to \mathcal{E} \to \mathcal{V}$ [24] as follows:

$$\begin{aligned} h_e^{(l+1)} &= \psi_V^{(l)}\left(h_e^{(l)}, \phi_V^{(l)}\left(\{h_v^{(l)} : v \in e\}\right)\right), \\ h_v^{(l+1)} &= \psi_E^{(l)}\left(h_v^{(l)}, \phi_E^{(l)}\left(\{h_e^{(l)} : v \in e\}\right)\right), \end{aligned} \tag{9}$$

  where $\phi_V^{(l)}$ passes messages from nodes to edges and $\phi_E^{(l)}$ passes messages from edges to nodes.

## A.3 Point Kernels and Simplices

One natural way of generalizing classical convolution [15] to non-Euclidean domain is to replace conventional two-dimensional kernels that reside on grids with point kernels that are defined on kernel points. Thomas et al. [14] introduced a point kernel in $\mathbb{R}^3$ for tasks that involve point clouds. Their empirical results suggested that the dispositions of their proposed kernel points form regular polyhedra in $\mathbb{R}^3$.

In our method for hypergraph tasks, we identify a hyperedge as an arbitrary high-dimensional point cloud, which requires generalizing point kernels to higher dimensions. This generalization naturally leads us to consider simplices as the higher-dimensional analogues of regular polyhedra. In this section, we provide a brief review of relevant concepts that are needed for later sections.

**Definition A.1** (Simplex [25]). *Let $\{\mathbf{s}_1, \mathbf{s}_2, \cdots, \mathbf{s}_{n+1}\}$ be a set of $n+1$ affinely independent points, i.e.,* $\mathrm{rank}[\mathbf{s}_2 - \mathbf{s}_1, \cdots, \mathbf{s}_{n+1} - \mathbf{s}_1] = n$. *Their convex hull $S$, defined as:*

$$S := \left\{ p \in \mathbb{R}^D \;\middle|\; p = \sum_{i=1}^{n+1} x_i \mathbf{s}_i,\ x_i \geq 0,\ \sum_{i=1}^{n+1} x_i = 1 \right\},$$

*is called an n-simplex embedded in $\mathbb{R}^D$. $\{\mathbf{s}_1, \mathbf{s}_2, \cdots, \mathbf{s}_{n+1}\}$ are called the vertices of $S$ and the line segment connecting a pair of vertices is called an edge of $S$. Further, if all edges of $S$ have the same lengths, then we say that $S$ is a regular simplex.*

A regular simplex, like regular polyhedra in $\mathbb{R}^3$, can be inscribed in a sphere, as thoroughly discussed in [26]. The following lemma presents the relationship between the radius of the sphere and the edge length of the inscribed simplex.

**Lemma A.2** (cf. proof of Theorem 1 in [26]). *Let a regular $n$-simplex be inscribed in a sphere of radius $R$. The relationship between the simplex edge length $l$ and sphere radius $R$ is given by:*

$$l = R\sqrt{\frac{2(n+1)}{n}}.$$

*Proof.* Denote the vertices of the regular n-simplex as $\mathbf{v}_0, ..., \mathbf{v}_n \in \mathbb{R}^n$, and suppose the simplex is inscribed in a ball of radius $R$. We place the centroid of the simplex at the origin. That is,

$$\sum_{k=0}^{n} \mathbf{v}_k = \mathbf{0}, \quad \text{and} \quad \|\mathbf{v}_k\| = R \text{ for all } k.$$

Since the edge length $\ell$ is the same in a regular simplex, we have:

$$\|\mathbf{v}_i - \mathbf{v}_j\|^2 = \ell^2 \Rightarrow \mathbf{v}_i \cdot \mathbf{v}_j = R^2 - \frac{\ell^2}{2}, \quad \text{for all } i \neq j.$$

Taking the squared norm of the centroid term yields

$$0 = \left\| \sum_{k=0}^{n} \mathbf{v}_k \right\|^2 = \sum_{k=0}^{n} \|\mathbf{v}_k\|^2 + 2 \sum_{0 \leq i < j \leq n} \mathbf{v}_i \cdot \mathbf{v}_j.$$

By expressing $\|\mathbf{v_k}\|$ and $\mathbf{v}_i \cdot \mathbf{v}_j$ in terms of $R$ and $\ell$, we have:

$$(n+1)R^2 + 2 \cdot \binom{n+1}{2} \left( R^2 - \frac{\ell^2}{2} \right) = 0,$$

$$\Rightarrow (n+1)R^2 + 2 \cdot \frac{(n+1)!}{2! \cdot (n-1)!} \left( R^2 - \frac{\ell^2}{2} \right) = 0,$$

$$\Rightarrow (n+1)R^2 + n(n+1)R^2 - \frac{n(n+1)}{2}\ell^2 = 0,$$

$$\Rightarrow \ell = R\sqrt{\frac{2(n+1)}{n}};\ R = l\sqrt{\frac{n}{2(n+1)}}.$$

□

## B Properties for Proposed Methods

### B.1 Properties of Kernel Points

#### B.1.1 Theoretical Analysis

In $\mathbb{R}^3$, minimizing the loss function (1) has been empirically shown to produce kernel point configurations that resemble regular polyhedra [14]. In our case, the kernel points are in $\mathbb{R}^n$. We begin by presenting the following result, which characterizes a class of critical points of the loss function (1):

**Proposition B.1.** *Let $K < n$. Suppose that the kernel points $\{\mathbf{z}_k\}_{k=1}^K \subset \mathbb{R}^n$ form a regular $(K-1)$-simplex centered at the origin. Then with a proper choice of radius for the circumscribed sphere, the gradient of the loss function in (1) vanishes at all kernel points, that is: $\nabla_{\mathbf{z}_k}\mathcal{L}(\{\mathbf{z}_k\}_{k=1}^K) = \mathbf{0}$ for all $k$.*

*Proof.* We first take the gradient of $\mathcal{L}$:

$$\begin{aligned}
\nabla_{\mathbf{z}_k}\mathcal{L} &= \nabla_{\mathbf{z}_k}\left[\sum_{l\neq k}\frac{1}{\|\mathbf{z}_l-\mathbf{z}_k\|} + \sum_{i=1,i\neq k}^{K}\frac{1}{\|\mathbf{z}_k-\mathbf{z}_i\|} + \|\mathbf{z}_k\|\right] \\
&= 2\nabla_{\mathbf{z}_k}\sum_{l\neq k}\|\mathbf{z}_k-\mathbf{z}_l\|^{-1} + \nabla_{\mathbf{z}_k}\|\mathbf{z}_k\| \\
&= 2\sum_{l\neq k}\nabla_{\mathbf{z}_k}\left[(\mathbf{z}_k-\mathbf{z}_l)\cdot(\mathbf{z}_k-\mathbf{z}_l)\right]^{-1/2} + \frac{\mathbf{z}_k}{\|\mathbf{z}_k\|} \\
&= -2\sum_{l\neq k}\frac{\mathbf{z}_k-\mathbf{z}_l}{\|\mathbf{z}_k-\mathbf{z}_l\|^3} + \frac{\mathbf{z}_k}{\|\mathbf{z}_k\|}.
\end{aligned}$$

For a regular $(K-1)$-simplex inscribed in a sphere of radius $R$, with centroid placed at the origin, we have

$$\|\mathbf{z}_k\| = R \text{ for all } k,$$

and, by Lemma A.2,

$$\|\mathbf{z}_k-\mathbf{z}_l\| = d := R\sqrt{\frac{2K}{K-1}} \text{ for all } k\neq l.$$

Substituting these into the gradient expression yields

$$\nabla_{\mathbf{z}_k}\mathcal{L} = -2\sum_{l\neq k}\frac{\mathbf{z}_k-\mathbf{z}_l}{d^3} + \frac{\mathbf{z}_k}{R}.$$

Using the property $\sum_{l=1}^K \mathbf{z}_l = \mathbf{0}$ for a centered simplex, we can simplify

$$\sum_{l\neq k}(\mathbf{z}_k-\mathbf{z}_l) = \sum_{l\neq k}\mathbf{z}_k - \sum_{l\neq k}\mathbf{z}_l = (K-1)\mathbf{z}_k - (-\mathbf{z}_k) = K\mathbf{z}_k,$$

which leads to

$$\begin{aligned}
\nabla_{\mathbf{z}_k}\mathcal{L} &= -2\frac{K\mathbf{z}_k}{d^3} + \frac{\mathbf{z}_k}{R} \\
&= \mathbf{z}_k\left(\frac{1}{R} - \frac{2K}{d^3}\right).
\end{aligned}$$

The critical point condition $\nabla_{\mathbf{z}_k}\mathcal{L} = \mathbf{0}$ requires:

$$\frac{1}{R} = \frac{2K}{d^3}.$$

We substitute the value $d = R\sqrt{\frac{2K}{K-1}}$ and solve for $R$, it yields a unique radius

$$R^* = \left(\frac{(K-1)^3}{2K}\right)^{\frac{1}{4}},$$

with which the gradient vanishes for all $\mathbf{z}_k$. □

While a regular simplex centered at the origin provides a symmetric and stationary configuration of kernel points, the asymmetric configuration used in our framework can improve optimization. Namely, prefixing one kernel point at the origin not only simplifies the optimization problem but also yields a lower loss value, as stated in Proposition B.2. Note that under a fixed configuration, the loss function can be viewed as a function of the radius of the circumscribed sphere.

**Proposition B.2.** *Let $K > 3$. The minimum value of the loss function $\mathcal{L}$ defined in (1) under the asymmetric configuration, with $\mathbf{z}_1 = \mathbf{0}$ and the remaining $K-1$ points forming a regular $K-2$ simplex, is strictly smaller than the minimum under a symmetric configuration, where all points are equidistant from the origin on a common sphere. That is,*

$$\mathcal{L}_{asym}(r^*) < \mathcal{L}_{sym}(R^*),$$

*where $r^*$ and $R^*$ denote the optimal radius for both configurations, respectively.*

*Proof.* First, we consider the symmetric kernel configuration, where $K$ points form a regular $(K-1)$-simplex inscribed in a sphere of radius $R^*$. From Lemma A.2 and Proposition B.1, the optimal edge length $d^*$ and radius $R^*$ satisfy:

$$d^* = R^* \sqrt{\frac{2K}{K-1}}; \quad R^* = \left(\frac{(K-1)^3}{2K}\right)^{\frac{1}{4}}.$$

Therefore, a direct calculation yields

$$\begin{aligned}\mathcal{L}_{\text{sym}} &= \binom{K}{2}\frac{1}{d^*} + KR^*,\\ &= \frac{K(K-1)}{2d^*} + KR^*,\\ &= \frac{1}{R^*}\cdot\sqrt{\frac{K(K-1)^3}{8}} + KR^*.\end{aligned}$$

Now, we consider the asymmetric kernel configuration, where one point is fixed at the origin and other $K-1$ points form a $K-2$ regular simplex inscribed in a sphere of radius $r^*$.

$$\tilde{d}^* = r^* \sqrt{\frac{2(K-1)}{K-2}}; \quad r^* = \left(\frac{(K-2)^3}{2(K-1)}\right)^{\frac{1}{4}}.$$

$$\begin{aligned}\mathcal{L}_{\text{asym}} &= (K-1)\frac{1}{r^*} + \binom{K-1}{2}\frac{1}{\tilde{d}^*} + (K-1)r^*,\\ &= (K-1)\frac{1}{r^*} + \frac{1}{r^*}\cdot\sqrt{\frac{(K-1)(K-2)^3}{8}} + (K-1)r^*.\end{aligned}$$

We conclude that $\mathcal{L}_{\text{sym}} \geq \mathcal{L}_{\text{asym}}$ after a direct calculation. For clarity, we visualize the comparison between the values of the two losses in Figure 2. $\square$

### B.1.2 Kernel Points Empirical Validation

While we have provided a theoretical analysis of the kernel point disposition, we further validate our findings through two empirical experiments. Specifically, we generate two sets of kernel points near the origin by minimizing the following loss function as proposed:

$$\mathcal{L}(\{\mathbf{z}_k\}_{k=1}^K) = \sum_{k=1}^{K}\sum_{l\neq k}\frac{1}{d(\mathbf{z}_l, \mathbf{z}_k)} + \sum_{k=1}^{K} d(\mathbf{0}, \mathbf{z}_k).$$

In the first setup, we optimize 5 kernel points in a 16-dimensional space; in the second, we use 16 kernel points in a 32-dimensional space. The pairwise distances among kernel points in both settings are visualized in Figure 4.

The results empirically confirm our theoretical prediction regarding the geometric disposition of kernel points. We also provide a simple visulization of the kernel's shape in Figure 3.

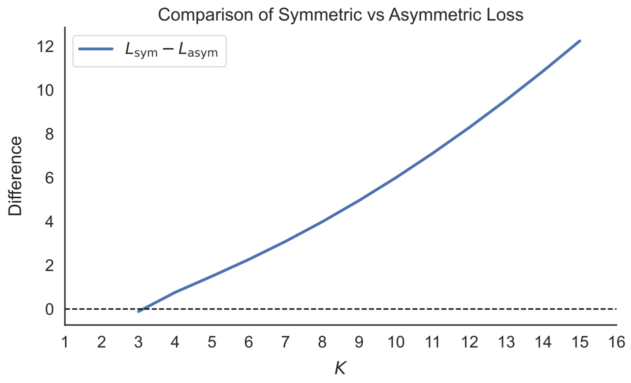

**Figure 2:** The comparison between $\mathcal{L}_{\text{sym}}$ and $\mathcal{L}_{\text{asym}}$.

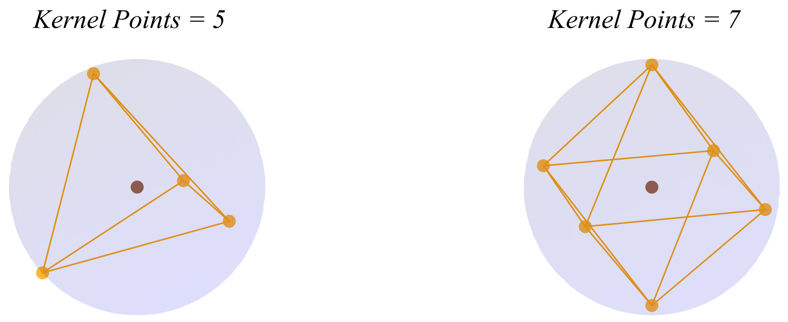

**Figure 3:** Illustration of kernel points, viewed from 2D. Each kernel consists of a center point and other points that form vertices of a simplex.

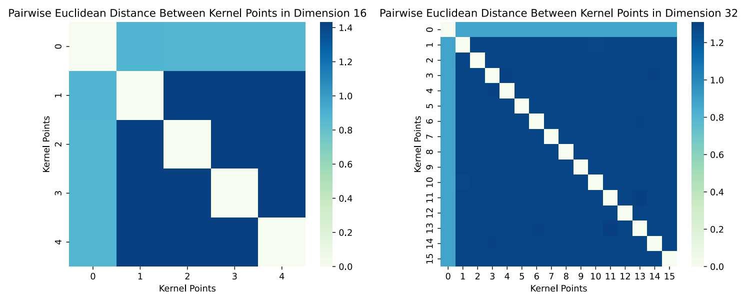

**Figure 4:** The result of minimizing the loss function.

### B.2 Properties on KAMP

One critical aspect of kernel construction is that the optimizer of the loss function $\mathcal{L}$ in (1) is not unique. Even fixing the $\mathbf{z}_1 = \mathbf{0}$, there are infinitely many possible positions for the remaining kernel points to form a regular simplex. For example, a rotation or reflection operation can be applied to the kernel:

$$\mathbf{R} \in O(n), \text{ where } O(n) = \{\mathbf{R} \in \mathbb{R}^{n \times n} | \mathbf{R}\mathbf{R}^\top = \mathbf{I}\}.$$

It is clear that $\{\mathbf{z}_k\}_{k=1}^K$ and $\{\mathbf{R}\mathbf{z}_k\}_{k=1}^K$ yield the same value for $\mathcal{L}$. Fixing other parameters in KAMP, the output will be generally different if we replace $\{\mathbf{z}_k\}_{k=1}^K$ with $\{\mathbf{R}\mathbf{z}_k\}_{k=1}^K$. Nevertheless, other parameters are learned given the kernel points, so the learning regime will also select different parameters under such replacement. Further, we systematically quantify the variation in predictions induced by different kernel configurations. In the following proposition, we establish a theoretical upper bound on the norm of the difference of output of KAMP.

**Proposition B.3.** *Let $\{\mathbf{z}_k\}_{k=1}^K$ and $\{\widetilde{\mathbf{z}}_k\}_{k=1}^K$ be two distinct point kernels that achieve the same value on $\mathcal{L}$ in (1). Let $\Delta$ denote the difference in their output produced by the KAMP module, specifically,*

$$\Delta = \left\| \mathrm{KAMP}_{\{\tilde{\mathbf{z}}_k\}}(\{\mathbf{x}_v\}_{v\in e}) - \mathrm{KAMP}_{\{\mathbf{z}_k\}}(\{\mathbf{x}_v\}_{v\in e}) \right\|.$$

*We have the following bounds on $\Delta$ respectively for the distance-based and attention-based correlations:*

- *Distance-based correlation:*
$$\Delta \le 4R_{max}\sqrt{RK(R_{\max}+R)}\max_{v,k}\|\mathbf{x}_{vk}\|;$$
- *Attention-based correlation:*
$$\Delta \le 2R\sqrt{K}\max_{v,k}\|\mathbf{x}_{vk}\|^2,$$

*where $K$ is the number of kernel points, $R$ is the radius of the circumscribed sphere for the $(K-1)$-simplex, $R_{\max} = \max_{v,k}\|\mathbf{z}_k - \mathbf{x}_{vk}\|$ denotes the maximal kernel-node distance.*

Before establishing the upper-bound, we need to prove the following additional lemma.

**Lemma B.4.** *Let $\sigma : \mathbb{R}^K \to \mathbb{R}^K$ denote the softmax function, and let $\mathbf{r} \in \mathbb{R}^K$ be an arbitrary vector where $\sigma_k(\mathbf{r})$ represents the $k$-th component of the output. Define the squared distance mapping $D : \mathbb{R}^K \to \mathbb{R}^K$ as:*

$$D(\mathbf{r}) = (-r_1^2, -r_2^2, \dots, -r_K^2),$$

*with coordinates bounded by $0 \le r_i \le R_{\max}$ for all $i \in \{1, \dots, K\}$. Then for any $\mathbf{r}, \widetilde{\mathbf{r}} \in \mathbb{R}^K$, the following inequality holds:*

$$|\sigma_k(D(\mathbf{r})) - \sigma_k(D(\tilde{\mathbf{r}}))| \le 2R_{\max}\sqrt{\sum_{i=1}^K |r_i^2 - \tilde{r}_i^2|}.$$

*Proof.* First, we bound the normed difference of $D$ as

$$\|D(\mathbf{r}) - D(\widetilde{\mathbf{r}})\|_2 = \left(\sum_{i=1}^K |r_i^2 - \widetilde{r}_i^2|^2\right)^{1/2} = \left(\sum_{i=1}^K |(r_i + \widetilde{r}_i)(r_i - \widetilde{r}_i)|^2\right)^{1/2}$$
$$\le 2R_{\max}\left(\sum_{i=1}^K |r_i - \widetilde{r}_i|^2\right)^{1/2} = 2R_{\max}\|\mathbf{r} - \widetilde{\mathbf{r}}\|_2.$$

Accordingly, the composition $\sigma \circ D$ satisfies

$$|\sigma_k(D(\mathbf{r})) - \sigma_k(D(\widetilde{\mathbf{r}}))| \le \|\sigma(D(\mathbf{r})) - \sigma(D(\widetilde{\mathbf{r}}))\|_2 \le \|D(\mathbf{r}) - D(\widetilde{\mathbf{r}})\|_2$$
$$\le 2R_{\max}\|\mathbf{r} - \widetilde{\mathbf{r}}\|_2 = 2R_{\max}\sqrt{\sum_{i=1}^K |r_i - \tilde{r}_i|^2}$$
$$\le 2R_{\max}\sqrt{\sum_{i=1}^K |r_i^2 - \tilde{r}_i^2|},$$

where the second line follows from the fact that $\sigma$ is a 1-Lipschitz function [27]. □

Now we are ready to prove the main proposition. We deal with the two cases separately.

**For Distance-based correlation.**

*Proof.* For convenience, we denote:

$$\tilde{d}_k = \|\tilde{\mathbf{z}}_k - \mathbf{x}_{vk}\|; \quad d_k = \|\mathbf{z}_k - \mathbf{x}_{vk}\|.$$

Also assume that the distance between linear transformed node feature and kernel points are also bounded by $R_{\max}$, that is:

$$R_{\max} = \max_{v,k} \|\mathbf{z}_k - \mathbf{x}_{vk}\|.$$

From the triangle inequality, we have:

$$|\tilde{d}_k - d_k| \le |\|\tilde{\mathbf{z}}_k - \mathbf{x}_{vk}\| - \|\mathbf{z}_k - \mathbf{x}_{vk}\|| \le \|\tilde{\mathbf{z}}_k - \mathbf{z}_k\| \le 2R.$$

For squared distances:

$$|\tilde{d}_k^2 - d_k^2| \le (\tilde{d}_k + d_k)|\tilde{d}_k - d_k| \le (2d_k + \tilde{d}_k - d_k)|\tilde{d}_k - d_k|.$$

$$\Rightarrow |\tilde{d}_k^2 - d_k^2| \le (2R_{\max} + 2R)2R.$$

Recall that the correlation coefficients are computed as the following:

$$\alpha_{vk} = \frac{\exp(-d_k^2)}{\sum_j \exp(-d_j^2)}, \quad \tilde{\alpha}_{vk} = \frac{\exp(-\tilde{d}_k^2)}{\sum_j \exp(-\tilde{d}_j^2)}.$$

Let $\mathbf{d} = (d_1, ..., d_k)$ and $\tilde{\mathbf{d}} = (\tilde{d}_1, ..., \tilde{d}_k)$. Also let $D : \mathbb{R}^K \to \mathbb{R}^K$ as:

$$D(\mathbf{r}) = (-d_1^2, -d_2^2, \ldots, -d_K^2).$$

Then, $\tilde{\alpha}_{vk} = \sigma_k(D(\tilde{\mathbf{d}}))$ and $\alpha_{vk} = \sigma_k(D(\mathbf{d}))$. Following lemma B.4, we get:

$$|\tilde{\alpha}_{v,k} - \alpha_{v,k}| = |\sigma_k(D(\tilde{\mathbf{d}})) - \sigma_k(D(\mathbf{d}))| \le 2R_{max}\sqrt{\sum_{i=1}^{K} |\tilde{d}_i^2 - d_i^2|}$$
$$\le 2R_{max}\sqrt{K(2R_{max} + 2R)2R}.$$

Now that we bound the correlation, we can bound the difference in the KAMP output under different kernels.

$$\begin{aligned}
\Delta &= \| \operatorname{KAMP}_{\{\tilde{\mathbf{z}}_k\}}(\{\mathbf{x}_v\}_{v\in e}) - \operatorname{KAMP}_{\{\mathbf{z}_k\}}(\{\mathbf{x}_v\}_{v\in e})\| \\
&= \|\frac{1}{d_e K}\sum_{v\in e}\sum_{k=1}^{K}[\tilde{\alpha}_{v,k} - \alpha_{v,k}]\mathbf{x}_{vk}\| \\
&\le \frac{1}{d_e K}\sum_{v,k}|\tilde{\alpha}_{v,k} - \alpha_{v,k}|\,\|\mathbf{x}_{vk}\| \\
&\le \frac{1}{d_e K}\sum_{v,k} 2R_{max}\sqrt{K(2R_{max} + 2R)2R}\,\|\mathbf{x}_{vk}\| \\
&= \frac{4R_{max}\sqrt{RK(R_{max} + R)}}{d_e K}\sum_{v,k}\|\mathbf{x}_{vk}\| \\
&\le 4R_{max}\sqrt{RK(R_{max} + R)}\max_{v,k}\|\mathbf{x}_{vk}\|.
\end{aligned}$$

□

**For Attention-based correlation.**

*Proof.* First, we define

$$\mathbf{u} = \left(\mathbf{z}_1^\top \mathbf{x}_{v1}, \dots, \mathbf{z}_K^\top \mathbf{x}_{vK}\right) \in \mathbb{R}^K,$$

and

$$\tilde{\mathbf{u}} = \left(\tilde{\mathbf{z}}_1^\top \mathbf{x}_{v1}, \dots, \tilde{\mathbf{z}}_K^\top \mathbf{x}_{vK}\right) \in \mathbb{R}^K.$$

Since

$$\left|\mathbf{z}_k^\top \mathbf{x}_{vk} - \tilde{\mathbf{z}}_k^\top \mathbf{x}_{vk}\right| = \left|(\mathbf{z}_k - \tilde{\mathbf{z}}_k)^\top \mathbf{x}_{vk}\right| \le \|\mathbf{z}_k - \tilde{\mathbf{z}}_k\| \, \|\mathbf{x}_{vk}\| \le 2R\|\mathbf{x}_{vk}\|,$$

we have

$$\|\mathbf{u} - \tilde{\mathbf{u}}\| = \sqrt{\sum_{k=1}^K \left(\mathbf{z}_k^\top \mathbf{x}_{vk} - \tilde{\mathbf{z}}_k^\top \mathbf{x}_{vk}\right)^2} \le \sqrt{\sum_{k=1}^K \left(2R\|\mathbf{x}_{vk}\|\right)^2} = 2R\|\mathbf{x}_{vk}\|\sqrt{K}.$$

Accordingly, we can bound the softmax output as follows:

$$|\sigma_k(\mathbf{u}) - \sigma_k(\tilde{\mathbf{u}})| \le \|\sigma(\mathbf{u}) - \sigma(\tilde{\mathbf{u}})\|_2 \le \|\mathbf{u} - \tilde{\mathbf{u}}\|_2 \le 2R\,\|\mathbf{x}_{vk}\|\,\sqrt{K}.$$

Applying it to the node-kernel correlation

$$\alpha_{vk} = \frac{\exp\left(\mathbf{z}_k^\top \mathbf{x}_{vk}\right)}{\sum_{j=1}^K \exp\left(\mathbf{z}_j^\top \mathbf{x}_{vj}\right)},$$

we have:

$$|\tilde{\alpha}_{vk} - \alpha_{vk}| \le 2R\|\mathbf{x}_{vk}\|\sqrt{K}.$$

Finally, we obtain

$$\begin{aligned}\Delta &= \left\|\text{KAMP}_{\{\tilde{\mathbf{z}}_k\}}(\{\mathbf{x}_v\}) - \text{KAMP}_{\{\mathbf{z}_k\}}(\{\mathbf{x}_v\})\right\| \\ &\le \frac{1}{d_e K}\sum_{v,k} |\tilde{\alpha}_{v,k} - \alpha_{v,k}| \, \|\mathbf{x}_{vk}\| \le \frac{2LR\sqrt{K}}{d_e K}\sum_{v,k}\|\mathbf{x}_{vk}\|^2 \\ &\le 2LR\sqrt{K}\max_{v,k}\|\mathbf{x}_{vk}\|^2.\end{aligned}$$

□

**Summary of Proposition B.3.** Proposition B.3 indicates that the output is more stable if the number of kernel points $K$ is smaller, and if the kernel radius $R$ is smaller. This result is intuitive because in the extreme case, if only one kernel point is used, then the kernel's "shape" is indeed invariant under rotations; if the kernel radius is 0, then all kernel points collapse to a single point, leading to the same case. Therefore, we believe that Proposition B.3 illustrates a trade-off between expressivity and stability. On one hand, we wish to use more kernels to capture the complex patterns intrinsically contained in a hyperedge. On the other hand, if we use too many kernels, the result might become less stable.

## C Benchmarks and Baselines

Node classification on hypergraphs is a classical semi-supervised learning task, where the model classifies each node as some known labels. In the following sections, we briefly introduce some representative hypergraph datasets and existing baselines.

### C.1 Datasets

We test the proposed model on both homophilic and heterophilic hypergraph datasets. Homophilic hypergraph datasets, including cocitation networks and coauthorship networks including Cocitation-Cora, Cocitation-Citeseer, Cocitaion-Pubmed, and Coauthorship-Cora, are all obtained from [3]. Other homophilic hypergraphs, ModelNet40 [28] and NTU2012 [29], comes from 3D object datasets widely used in computer vision tasks. Group-View Convolutional Neural Network (GVCNN) [30] and Multi-View Convolutional Neural Network (MVCNN) [31] are used for feature extraction, and then hypergraphs are constructed following the methods described in [2] and [32]. Heterophilic hypergraph datasets, including 20Newsgroups and Zoo are obtained from [33]. The remaining House and Senate datasets are introduced in [10] [34]. The detailed information about the datasets is summarized in Table 2.

**Table 2:** Statistics of the benchmark datasets

| **Dataset** | **Cora-CC** | **Citeseer-CC** | **Pubmed-CC** | **Cora-CA** | **ModelNet40** |
|---|---|---|---|---|---|
| $\|\mathcal{V}\|$ | 2,708 | 3,312 | 19,717 | 2,708 | 12,311 |
| $\|\mathcal{E}\|$ | 1,579 | 1,079 | 7,963 | 1,072 | 12,311 |
| #Features | 1,433 | 3,703 | 500 | 1,433 | 100 |
| #Classes | 7 | 6 | 3 | 7 | 40 |
| Homophily | 0.897 | 0.893 | 0.952 | 0.803 | 0.853 |
| **Dataset** | **NTU2012** | **Zoo** | **20Newsgroups** | **House** | **Senate** |
| $\|\mathcal{V}\|$ | 2,012 | 101 | 16,242 | 1,290 | 282 |
| $\|\mathcal{E}\|$ | 2,012 | 42 | 100 | 341 | 315 |
| #Features | 100 | 16 | 100 | 100 | 2 |
| #Classes | 67 | 7 | 4 | 2 | 2 |
| Homophily | 0.752 | 0.241 | 0.461 | 0.509 | 0.498 |

### C.2 Baselines

We compare our HypKN model with three categories of baselines. (1) Representative models specifically designed for hypergraph neural network, including HGNN [2], HCHA [9], HNHN [35], AllDeepSets & AllSetTransformer [10], ED-HNN [11], HyperGCN [3], and LE-GCN [32]; (2) MLP; (3) General framework that operates on both graph and hypergraph, including UniGCNII [36], and UniGEncoder [4]. To make a fair comparison, we adopt the same experimental condition of AllSetTransformer and ED-HNN. Therefore, we also divide the dataset into train, validation, and test sets by a proportion of 50:25:25. We also use mean accuracy and standard deviation, based on ten random trails as the evaluation metric.

## D More Experiments and Ablation Studies

### D.1 Ablation on Number of Kernels

We conduct additional experiments to explore the impact of the number of kernel points $K$ on model performance. Table 3 presents results using $K \in \{2, 3, 4, 5, 6, 7\}$ kernel points. We observe that the best performance is achieved with a moderate value of $K$. This suggests that a sufficient number of kernel points is necessary to effectively extract the inner latent structure of hyperedges, while too many kernel points may also lead to overfitting. This reflects a trade-off between model complexity and generalization capability. In addition, the experimental results demonstrate that using kernels to extract hyperedge information is both feasible and effective.

We also clarify that the number of kernel points $K$ in our model is different from the traditional concept of kernel size commonly used in convolution operations. In traditional convolutions, kernel size defines the receptive field, which determines the spatial scope of feature extraction. In contrast, in our model, the number of kernel points primarily governs the calculation complexity of the node-kernel correlation.

Additionally, increasing the number of kernel points enhances the directional resolution of our aggregator, enabling it to more finely perceive variations within local regions of the hyperedge. Geometrically, this corresponds to placing more vertices on a regular simplex, which allows the model to probe the shape from a richer set of directions. Thus, the aggregator tends to be more sensitive to the shape of hyperedge. However, it is also critical to balance the trade-off between geometric expressivity and optimization. With more kernel points involved, the overall aggregation becomes more complex, making it harder for the model to converge to an optimal solution during training.

### D.2 Further Ablation on the Effectiveness of KAMP

To demonstrate the effectiveness of the proposed KAMP module, we conduct two sets of ablation experiments, evaluating both component-wise contributions and robustness across settings.

**Complete Search.** We apply identical hyperparameter optimization for all ablation models using Optuna [37], and report results in Table 1. The following two variants are considered:

**Table 3:** Results of the HypKN performance, presented as mean accuracy (%) ± standard deviation, when selecting different number of kernel points. The experiment is conducted on all 10 benchmark datasets.

| $K$ | **Cora-CC** | **Citeseer-CC** | **Pubmed-CC** | **Cora-CA** | **ModelNet40** |
|---|---|---|---|---|---|
| 2 | 78.94 ± 0.82 | 73.82 ± 1.08 | **88.87 ± 0.46** | 82.02 ± 1.45 | **98.44 ± 0.21** |
| 3 | 78.52 ± 1.02 | 73.60 ± 1.28 | 88.43 ± 0.47 | 81.68 ± 1.66 | 98.39 ± 0.25 |
| 4 | **79.27 ± 1.38** | **74.01 ± 1.49** | 88.82 ± 0.47 | 82.54 ± 1.22 | 98.44 ± 0.28 |
| 5 | 78.30 ± 1.18 | 73.68 ± 1.30 | 88.86 ± 0.35 | 81.96 ± 0.72 | 98.41 ± 0.24 |
| 6 | 79.01 ± 0.90 | 73.44 ± 1.31 | 88.86 ± 0.43 | **82.71 ± 1.72** | 98.43 ± 0.21 |
| 7 | 78.60 ± 1.01 | 73.50 ± 1.05 | 88.81 ± 0.43 | 81.85 ± 1.07 | 98.30 ± 0.27 |
| $K$ | **NTU2012** | **Zoo** | **20Newsgroups** | **House** | **Senate** |
| 2 | 89.64 ± 1.23 | 95.00 ± 3.16 | 81.37 ± 0.46 | 77.34 ± 2.26 | **80.85 ± 4.31** |
| 3 | 89.76 ± 1.14 | 95.38 ± 3.53 | **81.95 ± 0.43** | **77.64 ± 2.75** | 78.73 ± 4.22 |
| 4 | 89.74 ± 0.69 | **96.54 ± 2.84** | 81.46 ± 0.49 | 77.49 ± 3.08 | 79.15 ± 3.68 |
| 5 | 89.68 ± 0.90 | 95.77 ± 2.84 | 81.46 ± 0.49 | 77.46 ± 2.74 | 79.15 ± 4.69 |
| 6 | **90.12 ± 1.14** | 95.38 ± 3.03 | 81.45 ± 0.43 | 77.55 ± 2.66 | 78.87 ±3.87 |
| 7 | 89.68 ± 1.19 | 94.62 ± 2.69 | 81.46 ± 0.49 | 77.62 ± 2.84 | 79.44 ± 3.59 |

**Table 4:** Ablation2: KAMP aggregator. Presented as mean accuracy (%) ± standard deviation. For each dataset, the best result is underlined, while the best results in each "KAMP setting" are highlighted in bold.

| **Data\Model** | HGNN++ | +KAMP (K=4) | | +KAMP (K=6) | |
|---|---|---|---|---|---|
| | | Distance | Attention | Distance | Attention |
| ModeNet40 | 97.25±0.29 | **97.77±0.21** | 97.71±0.27 | 97.56±0.30 | **97.59±0.29** |
| Zoo | 91.92±4.60 | 94.23±3.27 | **94.62±3.72** | **93.85±4.87** | 93.08±5.68 |
| House | 76.78±3.25 | 77.03±3.09 | **77.21±2.96** | 76.44±3.63 | **76.75±3.17** |
| NTU2012 | 86.98±1.53 | **88.69±1.36** | **88.69±1.36** | 88.87±1.26 | **88.89±1.41** |
| Senate | 77.75±4.03 | 77.61±3.00 | **78.31±3.40** | **78.59±5.30** | 78.45±4.88 |
| Pubmed-CC | 86.80±0.40 | **88.32±0.42** | **88.32±0.45** | 88.34±0.60 | **88.49±0.52** |

- **HKN-base**: Rather than using KAMP as operator $\phi_V$, we consider the alternative of taking a simple average aggregation, while keeping other settings identical to HypKN.
- **HKN w/o Self_State** : We simply follow the hyperpath $(\mathcal{V} \to \mathcal{E} \to \mathcal{V})$ message propagation while neglecting self-state values, while other settings are identical to HypKN.

The results are also summarized in the same Table 1. Notably, HGNN++ ranks 6th overall, while HKN w/o Self_State drops to 13th, despite achieving the best result on one dataset. This demonstrates the critical role both KAMP and the self-state mechanism play in achieving consistent performance across datasets. Meanwhile, HKN-base underperforms the HypKN model in 9 out of 10 benchmarks, highlighting the value of integrating the geometry-aware KAMP module in improving overall accuracy across datasets.

**Fixed Hyperparameter.** To further isolate the effect of KAMP, we start from a fixed hyperparameter configuration of **HGNN++**, and replace the standard aggregator $\phi_V$ with KAMP using a fixed number of kernel points. All other components are held constant. Detailed configurations are provided in Appendix F. Experiments are conducted on a subset of benchmarks.

Table 4 reports an ablation study evaluating the impact of (different) KAMP configurations on several benchmarks. We compare HGNN++ as the baseline with two KAMP variants: one using a distance-based correlation function, and the other using attention-based correlation. Each variant is tested with two kernel sizes, $K = 4$ and $K = 6$, while keeping all other components fixed. Across all datasets, incorporating KAMP leads to consistent improvements over HGNN++, demonstrating the benefit of KAMP's geometry-aware aggregation. We also observe that attention-based correlation often performs better than distance-based variants, suggesting that attention mechanisms may provide a more expressive way to capture geometric variation. Overall, these results further confirm the effectiveness of the KAMP module.

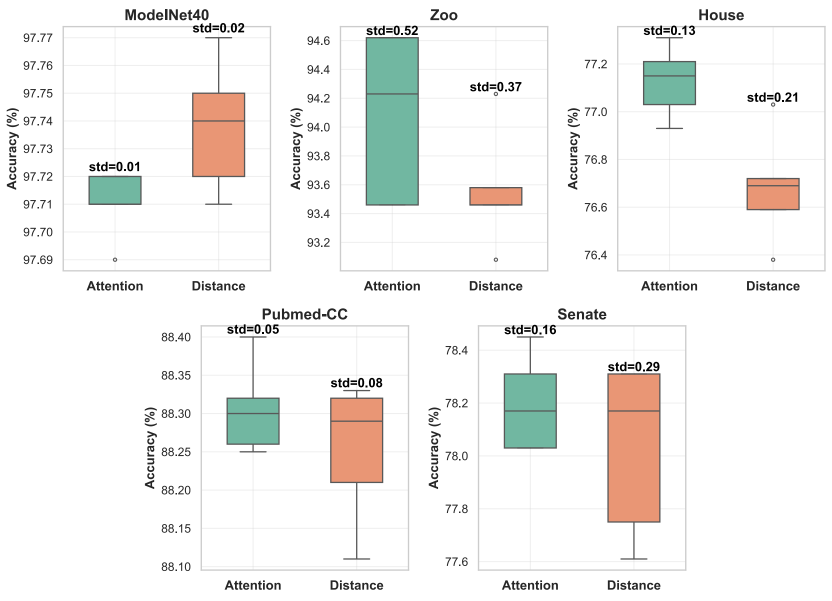

**Figure 5:** Impact of Kernel Disposition to Downstream Performance.

### D.3 Ablation on Kernel Dispositions

Proposition B.3 has shown the stability benefits that attention-based aggregation benefits, when the kernel points are initialized differently. We also conduct experiments to show how different kernel configurations impact the overall downstream performance. We take the same hyperparameter setting as in the ablation study in Table 4, and use different seed to initialize the kernel (leading to similar kernel shape but potentially different orientations). We run the model of different kernel setting by five times and visualized the perofrmance in Figure 5.

We observe that the attention-based correlation usually yields more stable results (lower standard deviation) and slightly higher mean accuracy among benchmarks, and this is especially notable on small datasets like Zoo and House. These results support our theoretical analysis in proposition B.3.

## E Complexity and Speed Anaysis

We analyze the computational complexity of the aggregation function $\phi_V$ on a single hyperedge. Given the embedding dimension $D$, hyperedge size $d_e$ and $K$ kernel points, the computation complexity is $O(d_e K D^2)$, whereas a conventional attention mechanism requires $O(d_e D^2)$. The additional complexity arises mainly from applying $K$ independent linear transformations per node. However, as shown in Table 3, a small $K$ is used to achieve optimal performance. Since $K$ is typically much smaller than $d_e$ and $D$, the additional cost remains modest in practice. We also compare the training speed (per-epoch) when using KAMP in the model, with other well-known architectures in the Appendix E.

One possible mitigation of complexity is to mimic modern MoE designs [21] and employ Top-K hard "kernel expert" selection, where only a sparse subset of transformations is activated for each node. This can substantially reduce the actual computation cost. However, this sparsity may reduce KAMP's sensitivity to the geometric shape of each hyperedge, potentially limiting its ability to capture subtle hyperedge patterns.

Table 5 reports the average per-epoch training time (in seconds) for our proposed HypKN model under different kernel sizes (K=2,4,6), compared with three representative baselines: LEGNN, EDHNN, and HyperGCN. As expected, increasing the number of kernel points in KAMP results in higher computational cost. For example, on ModelNet40, the per-epoch training time increases from 0.108s (K=2) to 0.154s (K=6). However, for datasets with smaller hypergraphs—such as Zoo, House, and

**Table 5:** Per-Epoch Training Time (in seconds).

| **Dataset** | HypKN (K=2) | HypKN (K=4) | HypKN (K=6) | LEGCN | EDHNN | HyperGCN |
|---|---|---|---|---|---|---|
| ModelNet40 | 0.108 | 0.130 | 0.154 | 0.008 | 0.023 | 0.010 |
| House | 0.018 | 0.018 | 0.019 | 0.005 | 0.017 | 0.008 |
| NTU2012 | 0.023 | 0.024 | 0.026 | 0.005 | 0.022 | 0.009 |
| Pubmed-CC | 0.119 | 0.142 | 0.166 | 0.009 | 0.023 | 0.011 |
| Senate | 0.018 | 0.018 | 0.019 | 0.005 | 0.022 | 0.008 |
| Zoo | 0.017 | 0.017 | 0.018 | 0.005 | 0.022 | 0.007 |

**Table 6:** Hyperparameter Settings for the ablation study.

| **Dataset** | **K** | **#Layers** | **Hidden Dim** | **LR** | **Dropout** | **Weight Decay** |
|---|---|---|---|---|---|---|
| Zoo | 4 | 3 | 32 | 0.02 | 0.5 | 0.0005 |
| NTU2012 | 4 | 2 | 256 | 0.01 | 0.3 | 0.0 |
| House Committees | 4 | 3 | 16 | 0.10 | 0.5 | 0.0005 |
| Senate Committees | 4 | 2 | 256 | 0.01 | 0.5 | 0.005 |
| ModelNet40 | 4 | 2 | 256 | 0.001 | 0.0 | 0.00005 |
| Pubmed-CC | 4 | 3 | 256 | 0.001 | 0.0 | 0.005 |

Senate—the training time remains largely unaffected by the number of kernel points, indicating that KAMP introduces minimal overhead when the hyperedge size is small.

## F Hyperparameter Searching and Settings

### F.1 Hyperparameter Search for Main Results

**Optuna** [37], a widely-used framework for efficient hyperparameter search, is adopted to automatically tune model hyperparameters for each dataset. Specifically, the following hyperparameters are optimized:

- `learning_rate` $\in \{1\text{e}{-4}, 1\text{e}{-3}, 1\text{e}{-2}, 2\text{e}{-2}, 0.1\}$
- `dropout` $\in \{0.0, 0.3, 0.5, 0.7, 0.9\}$
- `hidden` $\in \{16, 24, 32, 48, 64, 128, 256\}$
- `num_layers` $\in \{2, 3\}$
- `weight_decay` $\in \{0.0, 5\text{e}{-3}, 5\text{e}{-4}, 5\text{e}{-5}\}$
- `kernel_size` $\in \{2, 3, 4, 5, 6, 7, 8\}$ (fixed)

The search objective is set to `maximize` the mean validation accuracy averaged over 10 different random splits (with fixed train/val/test ratios), and the best test accuracy corresponding to the highest validation accuracy is also recorded. Each trial trains the model for 500 epochs, and a total of 100 trials are conducted per experiment.

Each dataset uses default data splits with:

- Training ratio: `train_prop = 0.5`
- Validation ratio: `valid_prop = 0.25`

We use such searching to conduct experiments for main results in Table 1.

### F.2 Hyperparameter Setting for Ablation

For the ablation experiments reported in Table 4 and Figure 5, we adopt a fixed hyperparameter configuration based on HGNN++, as summarized in Table 6. We then replace the simple aggregator ($\phi_V$) with our proposed KAMP module (using K=4 kernel points), while keeping all other settings unchanged.