Equivariant Hypergraph Neural Networks

Jinwoo Kim[1], Saeyoon Oh[1], Sungjun Cho[2], and Seunghoon Hong[1,2]

[1]KAIST
[2]LG AI Research

Abstract. Many problems in computer vision and machine learning can be cast as learning on hypergraphs that represent higher-order relations. Recent approaches for hypergraph learning extend graph neural networks based on message passing, which is simple yet fundamentally limited in modeling long-range dependencies and expressive power. On the other hand, tensor-based equivariant neural networks enjoy maximal expressiveness, but their application has been limited in hypergraphs due to heavy computation and strict assumptions on fixed-order hyperedges. We resolve these problems and present Equivariant Hypergraph Neural Network (EHNN), the first attempt to realize maximally expressive equivariant layers for general hypergraph learning. We also present two practical realizations of our framework based on hypernetworks (EHNN-MLP) and self-attention (EHNN-Transformer), which are easy to implement and theoretically more expressive than most message passing approaches. We demonstrate their capability in a range of hypergraph learning problems, including synthetic k-edge identification, semi-supervised classification, and visual keypoint matching, and report improved performances over strong message passing baselines. Our implementation is available at `https://github.com/jw9730/ehnn`.

Keywords: hypergraph neural network, graph neural network, permutation equivariance, semi-supervised classification, keypoint matching

1 Introduction

Reasoning about a system that involves a set of entities and their relationships requires relational data structures. Graph represents relational data with nodes and edges, where a node corresponds to an entity and an edge represents a relationship between a pair of nodes. However, pairwise edges are often insufficient to represent more complex relationships. For instance, many geometric configurations of entities such as angles and areas can only be captured by considering higher-order relationships between three or more nodes. Hypergraph is a general data structure that represents such higher-order relationships with hyperedges, *i.e.*, edges associating more than two nodes at a time [6]. Thus, it is widely used to represent various visual data such as scenes [19, 30], feature correspondence [4, 41, 49, 55], and polygonal mesh [8, 46, 58], as well as general relational data such as social networks [10, 37, 52], biological networks [22, 33], linguistic structures [15], and combinatorial optimization problems [28].

To learn deep representation of hypergraphs, recent works developed specialized hypergraph neural networks by generalizing the message passing operator of graph neural networks (GNNs) [2, 3, 13, 16, 18, 26]. In these networks, node and (hyper)edge features are updated recurrently by aggregating features of neighboring nodes and edges according to the connectivity of input (hyper)graph. Despite such simplicity, message passing networks have fundamental limitations. Notably, the local and recurrent operations of message passing prevents them from handling dependencies between any pair of nodes with distance longer than the number of propagation steps [21, 32]. It is also known that this locality is related to oversmoothing that hinders the use of deep networks [11, 26, 39, 47].

A more general and potentially powerful approach for hypergraph learning is to find *all* possible permutation equivariant linear operations on the input (hyper)graph and use them as bases of linear layers that constitute *equivariant GNNs* [43, 54]. While message passing is one specific, locally restricted case of equivariant operation, the maximal set of equivariant operations extends further, involving various global interactions over possibly disconnected nodes and (hyper)edges [32]. The formulation naturally extends to higher-order layers that can handle hypergraphs in principle and even mixed-order layers where input and output are of different orders (e.g., graph in, hypergraph out). Despite the advantages, the actual usage of equivariant GNNs has been mainly limited to sets and graphs [32, 43, 51, 59], and they have not been realized for general hypergraph learning. This is mainly due to the prohibitive parameter dimensionality of higher-order layers and a bound in the input and output hyperedge orders that comes from fixed-order tensor representation.

We propose *Equivariant Hypergraph Neural Network (EHNN)* as the first attempt to realize equivariant GNNs for general hypergraph learning. We begin by establishing a simple connection between sparse, arbitrarily structured hypergraphs and dense, fixed-order tensors, from which we derive the maximally expressive equivariant linear layer for undirected hypergraphs. Then, we impose an intrinsic parameter sharing within the layer via hypernetworks [23], which (1) retains maximal expressiveness, (2) practically bounds the number of parameters, and (3) allows processing hyperedges with arbitrary and possibly unseen orders. Notably, the resulting layer (EHNN-MLP) turns out to be a simple augmentation of an MLP-based message passing with hyperedge order embedding and global pooling. This leads to efficient implementation and also allows incorporation of any advances in the message passing literature. We further extend into a Transformer counterpart (EHNN-Transformer) by introducing self-attention to achieve a higher expressive power with the same asymptotic cost. In a challenging synthetic k-edge identification task where message passing networks fail, we show that the high expressiveness of EHNN allows fine-grained global reasoning to perfectly solve the task, and demonstrate their generalizability towards unseen hyperedge orders. We also demonstrate their state-of-the-art performance in several transductive and inductive hypergraph learning benchmarks, including semi-supervised classification and visual correspondence matching.

2 Preliminary and Related Work

Let us introduce preliminary concepts from permutation equivariant learning [32, 43, 51]. We first describe higher-order tensors, then describe maximally expressive permutation equivariant linear layers that compose equivariant GNNs [43].

We begin with some notations. We denote a set as $\{a, \ldots, b\}$, a tuple as $(a, \ldots, b)$, and $[n] = \{1, \ldots, n\}$. We denote the space of order-k tensors as $\mathbb{R}^{n^k \times d}$ with feature dimension d. For an order-k tensor $\mathbf{A} \in \mathbb{R}^{n^k \times d}$, we use a multi-index $\mathbf{i} = (i_1, \ldots, i_k) \in [n]^k$ to index an element $\mathbf{A}_{\mathbf{i}} = \mathbf{A}_{i_1,\ldots,i_k} \in \mathbb{R}^d$. Let S_n denote all permutations of $[n]$. A node permutation $\pi \in S_n$ acts on a multi-index $\mathbf{i}$ by $\pi(\mathbf{i}) = (\pi(i_1), \ldots, \pi(i_k))$, and acts on a tensor $\mathbf{A}$ by $(\pi \mathbf{A})_{\mathbf{i}} = \mathbf{A}_{\pi^{-1}(\mathbf{i})}$.

Higher-order Tensors Prior work on equivariant learning regard a hypergraph data as $G = (V, \mathbf{A})$ where V is a set of n nodes and $\mathbf{A} \in \mathbb{R}^{n^k \times d}$ is a tensor that encodes hyperedge features [32, 43, 51]. The order k of the tensor $\mathbf{A}$ indicates the type of hypergraph. First-order tensor encodes a set of features (*e.g.*, point cloud) where $\mathbf{A}_i$ is feature of node i. Second-order tensor encodes pairwise edge features (*e.g.*, adjacency) where $\mathbf{A}_{i_1,i_2}$ is feature of edge (i_1, i_2). Generally, an order-k tensor encodes *hyperedge* features (*e.g.*, mesh normal) where $\mathbf{A}_{i_1,\ldots,i_k}$ is feature of hyperedge $(i_1, \ldots, i_k)$. We begin our discussion from tensors, but will arrive at the familiar notion of hypergraphs with any-order undirected hyperedges [18].

Permutation Invariance and Equivariance In (hyper)graph learning, we are interested in building a function f that takes a (higher-order) tensor $\mathbf{A}$ as input and outputs some value Y. Since the tensor representation of the graph changes dramatically with the permutation of node indices, the function f should be invariant or equivariant under node permutations. Formally, if the output Y is a single vector, f is required to be *permutation invariant*, always satisfying $f(\pi \mathbf{A}) = f(\mathbf{A})$; if we want Y to be a tensor $Y = \mathbf{T}$, f is required to be *permutation equivariant*, always satisfying $f(\pi \mathbf{A}) = \pi f(\mathbf{A})$. As a neural network f is often built as a stack of linear layers and non-linearities, its construction reduces to finding invariant and equivariant *linear* layers.

Invariant and Equivariant Linear Layers Many (hyper)graph neural networks rely on message passing [18, 20], which is a restricted equivariant operator. Alternatively, tensor-based *maximally expressive* linear layers have been characterized by Maron el. al. (2019) [43]. Specifically, invariant linear layers $L_{k \to 0} : \mathbb{R}^{n^k \times d} \to \mathbb{R}^{d'}$ and equivariant linear layers $L_{k \to l} : \mathbb{R}^{n^k \times d} \to \mathbb{R}^{n^l \times d'}$ were identified (note that invariance is a special case of equivariance with $l = 0$). Given an order-k input $\mathbf{A} \in \mathbb{R}^{n^k \times d}$, the order-$l$ output of an equivariant linear layer $L_{k \to l}$ is written as follows, with indicator $\mathbb{1}$ and multi-indices $\mathbf{i} \in [n]^k, \mathbf{j} \in [n]^l$:

$$L_{k \to l}(\mathbf{A})_{\mathbf{j}} = \sum_{\mu} \sum_{\mathbf{i}} \mathbb{1}_{(\mathbf{i},\mathbf{j}) \in \mu} \mathbf{A}_{\mathbf{i}} w_{\mu} + \sum_{\lambda} \mathbb{1}_{\mathbf{j} \in \lambda} b_{\lambda}, \tag{1}$$

where $w_\mu \in \mathbb{R}^{d \times d'}$, $b_\lambda \in \mathbb{R}^{d'}$ are weight and bias parameters, and and are *equivalence classes* of order-$(k+l)$ and order-l multi-indices, respectively.

The equivalence classes can be interpreted as a partitioning of a multi-index space. The equivalence classes for the weight specifies a partitioning of the space of order-$(k+l)$ multi-indices $[n]^{k+l}$, and for the bias specifies a partitioning of the space of order-l multi-indices $[n]^l$. The total number of the equivalence classes (the size of partitioning) depends only on orders k and l. With $\mathrm{b}(k)$ the k-th Bell number, there exist $\mathrm{b}(k+l)$ equivalence classes for the weight and $\mathrm{b}(l)$ equivalence classes for the bias. For first-order layer $L_{1 \to 1}$, there exist $\mathrm{b}(2) = 2$ equivalence classes $_1$, $_2$ for the weight, specifying the partitioning of $[n]^2$ as $\{ _1; \ _2\}$ where $_1 = \{(i,j) \mid i = j\}$ and $_2 = \{(i,j) \mid i \neq j\}$. For further details, we guide the readers to Maron et. al. (2019) [43] and Kim et. al. (2021) [32].

Equivariant GNNs Based on the maximally expressive equivariant linear layers (Eq. (1)), a bouquet of permutation invariant or equivariant neural networks were formulated. A representative example is *equivariant GNN* [43] (also called k-IGN [12, 42]) built by stacking the equivariant linear layers and non-linearity. Their theoretical expressive power has been extensively studied [12, 29, 42, 44, 59], leading to successful variants in set and graph learning [31,32,45,51]. In particular, practical variants such as Higher-order Transformer [32] and TokenGT [31] unified equivariant GNNs and Transformer architecture [36, 53], surpassing the graph learning performance of message passing GNNs by a large margin.

Challenges in Hypergraph Learning Despite the theoretical and practical advantages, to our knowledge, equivariant GNN and its variants were rarely considered for general hypergraph learning with higher-order data [1], and never implemented except for highly restricted k-uniform hyperedge prediction [32, 51]. We identify two main challenges. First, although the asymptotic cost can be reduced to a practical level with recent tricks [32], the number of parameters still grows rapidly to Bell number of input order [5]. This makes any layer $L_{k \to l}$ with $k + l > 4$ challenging to use, as $k + l = 5$ already leads to 52 weight matrices. Second, in inductive learning [24, 60] where a model is tested on unseen nodes or hypergraphs, the model can be required to process unseen-order hyperedges that possibly surpass the max order in the training data. This is not straightforward for equivariant GNNs, because fixed-order tensors that underlie $L_{k \to l}$ require to pre-specify the max hyperedge order $(k; l)$ that the model can process.

3 Equivariant Hypergraph Neural Network

We now proceed to our framework on practical equivariant GNNs for general hypergraph data. All proofs can be found in Appendix A.1. In practical setups that assume undirected hypergraphs [2, 3, 13, 16, 18, 57], a hypergraph $G = (V; E; \mathbf{X})$ is defined by a set of n nodes V, a set of m hyperedges E, and features $\mathbf{X} \in \mathbb{R}^{m \times d}$ of the hyperedges. Each hyperedge $e \in E$ is a subset of node set V, and its order

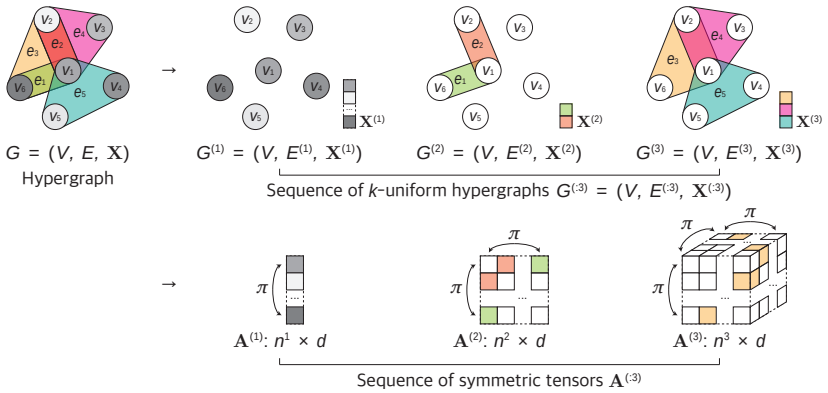

Fig. 1: Example of a hypergraph represented as a sequence of k-uniform hypergraphs (Definition 1), or equivalently a sequence of symmetric higher-order tensors (Definition 3). Note that nodes are handled as first-order hyperedges.

$|e|$ indicates its type. For example, a first-order edge $\{i\}$ represents an i-th node; a second-order edge $\{i, j\}$ represents a pairwise link of i-th and j-th nodes; in general, an order-k edge $\{i_1, \ldots, i_k\}$ represents a hyperedge that links k nodes. By $\mathbf{X}_e \in \mathbb{R}^d$ we denote the feature attached to a hyperedge e. We assume that node and hyperedge features are both d-dimensional [12, 32, 42, 43]; to handle different dimensionalities, we simply let $d = (d_v + d_e)$ and place node features at first d_v channels and hyperedge features at last d_e channels.

Note that above notion of hypergraphs $(V, E, \mathbf{X})$ does not directly align to higher-order tensors $\mathbf{A} \in \mathbb{R}^{n^k \times d}$ (Section 2) – unlike them, hypergraphs of our interest are sparse, undirected, and each hyperedge contains unique node indices. As equivariant GNNs (Section 2) build upon higher-order tensors, it is necessary to establish a connection between hypergraphs and the higher-order tensors.

3.1 Hypergraph as a Sequence of Higher-order Tensors

To describe hypergraphs $(V, E, \mathbf{X})$ using higher-order tensors $\mathbf{A} \in \mathbb{R}^{n^k \times d}$, it is convenient to introduce *k-uniform hypergraphs*. A hypergraph is k-uniform if all of its hyperedges are exactly of order-k. For example, a graph without self-loops is 2-uniform, and a triangle mesh is 3-uniform. From that, we can define equivalent representation of a hypergraph as a *sequence* of k-uniform hypergraphs:

Definition 1 *The sequence representation of a hypergraph $(V, E, \mathbf{X})$ with max hyperedge order K is a sequence of k-uniform hypergraphs with $k \leq K$, written as $(V, E^{(k)}, \mathbf{X}^{(k)})_{k \leq K} = (V, E^{(:K)}, \mathbf{X}^{(:K)})$ where $E^{(k)}$ as the set of all order-k hyperedges in E and $\mathbf{X}^{(k)}$ as a row stack of features $\{\mathbf{X}_e | e \in E^{(k)}\}$.*

As the collection $(E^{(k)})_{k \leq K}$ forms a partition of E, we can retrieve the original hypergraph $(V, E, \mathbf{X})$ from its sequence representation $(V, E^{(k)}, \mathbf{X}^{(k)})_{k \leq K}$ by using the union of $(E^{(k)})_{k \leq K}$ for E and the concatenation of $(\mathbf{X}^{(k)})_{k \leq K}$ for $\mathbf{X}$.

The concept of uniform hypergraph is convenient because we can draw an equivalent representation as a *symmetric* higher-order tensor [13, 35]. An order-k tensor $\mathbf{A}$ is symmetric if its entries are invariant under reordering of indices, *e.g.*, $\mathbf{A}_{ij} = \mathbf{A}_{ji}$, $\mathbf{A}_{ijk} = \mathbf{A}_{kij} = \cdots$, and so on. From that, we can define the equivalent representation of a k-uniform hypergraph as an order-k symmetric tensor[1]:

Definition 2 *The tensor representation of k-uniform hypergraph $(V; E^{(k)}; \mathbf{X}^{(k)})$ is an order-k symmetric tensor $\mathbf{A}^{(k)} \in \mathbb{R}^{n^k \times d}$ defined as follows:*

$$\mathbf{A}^{(k)}_{(i_1,\ldots,i_k)} = \begin{cases} \mathbf{X}^{(k)}_e & \text{if } e = \{i_1, \ldots, i_k\} \in E^{(k)} \\ 0 & \text{otherwise} \end{cases}. \tag{2}$$

From $\mathbf{A}^{(k)}$, we can retrieve the original k-uniform hypergraph $(V; E^{(k)}; \mathbf{X}^{(k)})$ by first identifying the indices of all nonzero entries of $\mathbf{A}^{(k)}$ to construct $E^{(k)}$, and then using $E^{(k)}$ to index $\mathbf{A}^{(k)}$ to construct $\mathbf{X}^{(k)}$.

Now, directly combining Definition 1 and 2, we can define the equivalent representation of a hypergraph as a sequence of higher-order tensors:

Definition 3 *The tensor sequence representation of a hypergraph $(V; E; \mathbf{X})$ with maximum hyperedge order K is a sequence of symmetric higher-order tensors $(\mathbf{A}^{(k)})_{k=K} = \mathbf{A}^{(:K)}$, where each $\mathbf{A}^{(k)}$ is the tensor representation (Definition 2) of each k-uniform hypergraph $(V; E^{(k)}; \mathbf{X}^{(k)})$ that comes from the sequence representation of the hypergraph $(V; E^{(k)}; \mathbf{X}^{(k)})_{k=K} = (V; E^{(:K)}; \mathbf{X}^{(:K)})$ (Definition 1).*

An illustration is in Fig. 1. Note that we can include node features as $\mathbf{A}^{(1)}$. Now, our problem of interest reduces to identifying a function f that operates on sequences of tensors $\mathbf{A}^{(:K)}$ that represent hypergraphs. The concept of permutation invariance and equivariance (Section 2) applies similarly here. A node permutation $\sigma \in S_n$ acts on a tensor sequence $\mathbf{A}^{(:K)}$ by jointly acting on each tensor, $\sigma \mathbf{A}^{(:K)} = (\sigma \mathbf{A}^{(k)})_{k=K}$. An invariant f always satisfies $f(\sigma \mathbf{A}^{(:K)}) = f(\mathbf{A}^{(:K)})$, and an equivariant f always satisfies $f(\sigma \mathbf{A}^{(:K)}) = \sigma f(\mathbf{A}^{(:K)})$.

3.2 Equivariant Linear Layers for Hypergraphs

In Definition 3, we represented a hypergraph as a sequence of symmetric higher-order tensors $(\mathbf{A}^{(k)})_{k=K}$, each $\mathbf{A}^{(k)}$ representing a k-uniform hypergraph. We now utilize equivariant linear layers $L_{k \to l} : \mathbb{R}^{n^k \times d} \to \mathbb{R}^{n^l \times d'}$ (Eq. (1)) in Section 2 to formalize equivariant linear layers that input and output hypergraphs. The idea is to find and combine all pairwise linear maps between tensors (*i.e.*, k-uniform hypergraphs) of input and output sequences. Although seemingly simple, this gives the *maximally expressive* equivariant linear layer for hypergraphs.

[1] Higher-order tensors can in principle represent directed hypergraphs as well; we constrain them to be symmetric to specifically represent undirected hypergraphs.

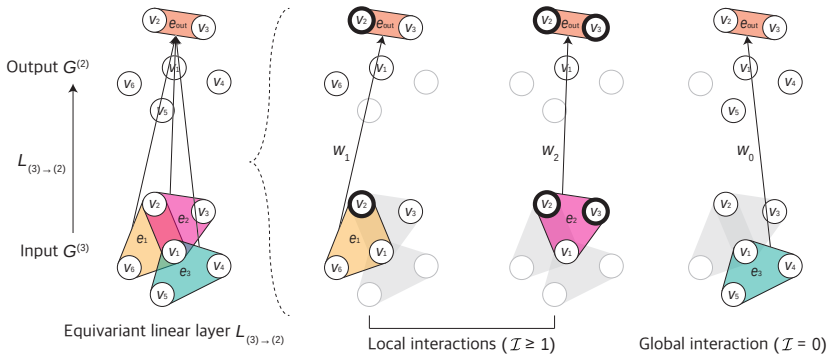

Fig. 2: Conceptual illustration of an equivariant linear layer $L_{(3)\,!\,(2)}$ as in Eq. (3). The layer uses different weights w_l for different overlaps l between input and output hyperedges. This gives rise to local interactions similar to message-passing ($l \quad 1$) and global interactions ($l = 0$) implemented as global sum-pooling.

Equivariant Linear Layers for k-uniform Hypergraphs In Section 2, we argued that the equivariant linear layer $L_{k\,!\,l}$ cannot be practically used due to the prohibitive number of $\mathrm{b}(k+l)$ weights and $\mathrm{b}(l)$ biases. Yet, when the input and output tensors are restricted to k- and l-uniform hypergraphs respectively, we can show that the layer reduces to $O(k+l)$ weights and a single bias:

Proposition 1. *Assume that the input and output of equivariant linear layer $L_{k\,!\,l}$ (Eq. (1)) are constrained to symmetric tensors that represent k- and l-uniform hypergraphs respectively (Eq. (2)). Then it reduces to $L_{(k)\,!\,(l)}$ below:*

$$L_{(k)\,!\,(l)}(\mathbf{A}^{(k)})_{\mathbf{j}} = \mathbb{1}_{j\mathbf{j}j=l}\left(\sum_{l=1}^{\min(k,l)}\sum_{\mathbf{i}}\mathbb{1}_{j\mathbf{i}\setminus\mathbf{j}j=l}\,\mathbf{A}^{(k)}_{\mathbf{i}}w_l + \sum_{\mathbf{i}}\mathbf{A}^{(k)}_{\mathbf{i}}w_0 + b_l\right); \quad (3)$$

where $w_o; w_l \;\; 2 \;\; \mathsf{R}^{d \quad d'}$, $b_l \;\; 2 \;\; \mathsf{R}^{d'}$ are weight and bias, $j\mathbf{i}j$ is number of distinct elements in $\mathbf{i}$, and $j\mathbf{i} \setminus \mathbf{j}j$ is number of distinct intersecting elements in $\mathbf{i}$ and $\mathbf{j}$.

The idea for the proof is that, if input and output are constrained to tensors from Eq. (2), a large number of parameters in the original layer are tied to adhere to the symmetry. This leads to much less parameters compared to the original Bell number version ($L_{k\,!\,l}$). Still, note that $L_{(k)\,!\,(l)}$ (Eq. (3)) is still a maximally expressive linear layer as it produces the identical outputs with $L_{k\,!\,l}$.

Notably, Eq. (3) reveals that maximal expressiveness is composed of sophisticated local message passing augmented with global interaction. In the first term of Eq. (3), the constraint $\mathbb{1}_{j\mathbf{i}\setminus\mathbf{j}j>0}$ specifies local dependency between *incident* input and output hyperedges having at least one overlapping nodes. Yet, this local interaction is more *fine-grained* than conventional message passing, as it uses separate weights w_l for different numbers of overlapping nodes l (Fig. 2). This is reminiscent of recent work in subgraph message passing [7] that improve

expressive power of GNNs. In addition, the layer contains intrinsic global interaction via pooling in second term of Eq. (3), which reminds of virtual node or global attention [27,34,38,40,48,56] that also improve expressive power [48].

Equivariant Linear Layers for Hypergraphs We now construct maximally expressive equivariant linear layers for undirected hypergraphs. We begin by representing a hypergraph as a sequence of tensors $(\mathbf{A}^{(k)})_{k \quad K} = \mathbf{A}^{(:K)}$ (Definition 3). The goal is to construct the linear layer $L_{(:K) \,!\, (:L)}$ to input and output those tensor sequences while being equivariant $L_{(:K) \,!\, (:L)}(\quad \mathbf{A}^{(:K)}) = \quad L_{(:K) \,!\, (:L)}(\mathbf{A}^{(:K)})$. For this, we use all pairwise linear layers $L_{(k) \,!\, (l)}$ (Eq. (3)) between tensors of input and output sequences:

$$L_{(:K) \,!\, (:L)}(\mathbf{A}^{(:K)}) = \left(\sum_{k \quad K} L_{(k) \,!\, (l)}(\mathbf{A}^{(k)}) \right)_{l \quad L} : \tag{4}$$

For better interpretation, we plug Eq. (3) into Eq. (4) and rewrite it with respect to $\mathbf{j}$-th entry of l-th (order-l) output tensor:

$$\begin{aligned} L_{(:K) \,!\, (:L)}(\mathbf{A}^{(:K)})_{l,\mathbf{j}} = \mathbb{1}_{|\mathbf{j}|=l} \sum_{k \quad K} \sum_{l=1}^{\min(k,l)} \sum_{\mathbf{i}} \mathbb{1}_{|\mathbf{i}| \setminus |\mathbf{j}|=l} \mathbf{A}_{\mathbf{i}}^{(k)} w_{k,l,l} \\ + \mathbb{1}_{|\mathbf{j}|=l} \sum_{k \quad K} \sum_{\mathbf{i}} \mathbf{A}_{\mathbf{i}}^{(k)} w_{k,l,0} + \mathbb{1}_{|\mathbf{j}|=l} b_l : \end{aligned} \tag{5}$$

Note that we added subscripts $(k; l)$ to w_0, w_l to differentiate between weights from each sublayer $L_{(k) \,!\, (l)}$ as they are involved in different computations. On the other hand, the biases from sublayers $(L_{(k) \,!\, (l)})_{k \quad K}$ carry out exactly the same computation, and can be merged to a single bias b_l. As a result, $L_{(:K) \,!\, (:L)}$ contains $\sum_{l \quad L,k \quad K}(1+\min(k; l))$ weights and L biases, achieving a better scalability than the original $L_{K \,!\, L}$ that has exponentially many weights and biases.

Similar to sublayers $L_{(k) \,!\, (l)}$ (Eq. (3)), we see that the combined layer for general hypergraphs $L_{(:K) \,!\, (:L)}$ (Eq. (5)) is a mixture of fine-grained local message passing and global interaction. In this case, the local interactions utilize different weights $w_{k,l,l}$ for each triplet $(k; l; l)$ that specifies dependency between order-k input and order-l output hyperedges with l overlapping nodes. Similarly, global interactions (pooling) utilize different weights $w_{k,l,0}$ for each pair $(k; l)$, specifying global dependency between all order-k input and order-l output hyperedges. Finally, different biases b_l are assigned for each output hyperedge order l.

Importantly, we can show the following:

Theorem 1. *$L_{(:K) \,!\, (:L)}$ (Eq. (4)) is the maximally expressive equivariant linear layer for undirected hypergraphs represented as tensor sequences.*

Similar as in Proposition 1, the idea for the proof is to appropriately constraint the input and output of the maximally expressive equivariant linear layer $L_{K \,!\, L}$, and observe that most of its parameters are tied and reduced, leading to $L_{(:K) \,!\, (:L)}$. Still, the layer retains maximal expressiveness of the original layer ($L_{K \,!\, L}$) as it produces the identical output.

3.3 Equivariant Hypergraph Neural Networks (EHNN)

In Section 3.2, we introduced equivariant linear layers for general undirected hypergraphs $L_{(:K)\,!\,(:L)}$ by composing order-specific sublayers $L_{(k)\,!\,(l)}$ for $k \quad K$, $l \quad L$ and proved their maximal expressiveness. Yet, these layers are still unsuitable to be used in practice because they cannot input or output hypergraphs with orders exceeding $(K; L)$, and the number of weights and biases grows at least linearly to $(K; L)$ that can reach several hundreds in practice. To resolve the problems jointly, we propose the concept of *Equivariant Hypergraph Neural Network* (EHNN) that introduces intrinsic trainable parameter sharing via *hypernetworks* [23]. More specifically, we impose parameter sharing within $L_{(:K)\,!\,(:L)}$ and across all sublayers $L_{(k)\,!\,(l)}$ via two hypernetworks, each for weights and biases[2]. As a result, an EHNN layer is defined as follows, with hypernetworks $W : \mathsf{N}^3 \;!\; \mathsf{R}^{d\;\;d'}$ and $B : \mathsf{N} \;!\; \mathsf{R}^{d'}$ inferring all weights $w_{k,l,I}$ and biases b_l (Eq. (5)) from the subscripts $(k; l; I)$ and (l) respectively:

$$\begin{aligned}\mathrm{EHNN}(\mathbf{A}^{(:K)})_{l,\mathbf{j}} = \mathbb{1}_{\mathbf{j}\mathbf{j}=l} \sum_{k \quad K} \sum_{l=1}^{\min(k,l)} \sum_{\mathbf{i}} \mathbb{1}_{\mathbf{j}\mathbf{i}\setminus\mathbf{j}\mathbf{j}=I} \mathbf{A}_{\mathbf{i}}^{(k)} W(k; l; I) \\ + \mathbb{1}_{\mathbf{j}\mathbf{j}=l} \sum_{k \quad K} \sum_{\mathbf{i}} \mathbf{A}_{\mathbf{i}}^{(k)} W(k; l; 0) + \mathbb{1}_{\mathbf{j}\mathbf{j}=l} B(l): \end{aligned} \tag{6}$$

In principle, this preserves maximal expressiveness of $L_{(:K)\,!\,(:L)}$ when W and B are parameterized as MLPs, as by universal approximation they can learn any lookup table that maps subscripts to weights and biases [25]. Furthermore, as hypernetworks W and B can produce weights for arbitrary hyperedge orders $(k; l; I)$, we can remove the bound in hyperedge orders from the specification of the layer and use a single EHNN layer with bounded parameters to any hypergraphs with unbounded or unseen hyperedge orders. Conclusively, EHNN layer is by far the first attempt that is maximally expressive while being able to process arbitrary-order hypergraphs by construction.

3.4 Practical Realization of EHNN

The EHNN layer in Eq. (6) is conceptually elegant, but in practice it can be costly as we need to explicitly hold all output matrices of the hypernetwork $W(k; l; I)$ 2 $\mathsf{R}^{d\;\;d'}$ in memory. This motivates us to seek for simpler realizations of EHNN that can be implemented efficiently while retaining the maximal expressiveness (*i.e.*, being able to model $L_{(:K)\,!\,(:L)}$ (Theorem 1) thus exhausting the full space of equivariant linear layers on undirected hypergraphs). To this end, we propose EHNN-MLP that utilizes three consecutive MLPs to approximate the role of the weight hypernetwork, and also propose its extension EHNN-Transformer with attention mechanism. Then, we finish the section by providing a comparative analysis of EHNN-MLP and EHNN-Transformer with respect to the existing message passing hypergraph neural networks.

[2] Note that we do not share the hypernetworks across different levels of layers.

Realization with MLP We first introduce *EHNN-MLP*, a simple realization of EHNN with three elementwise MLPs $_{1:3}$ where each $_p : \mathbb{N} \quad \mathbb{R}^{d_p} \; ! \; \mathbb{R}^{d'_p}$ takes a positive integer as an auxiliary input. The intuition here is to *decompose* the weight application with hypernetwork $W(k; l; l)$ into three consecutive MLPs $_1(k;)$, $_2(l;)$, and $_3(l;)$, eliminating the need to explicitly store the inferred weights for each triplet $W(k; l; l)$. We characterize EHNN-MLP as follows:

$$\text{EHNN-MLP}(\mathbf{A}^{(:K)})_{l,\mathbf{j}} = \; _3\left(l; \sum_{l \; 0} \; _2\left(l; \sum_{k \; K} \sum_{\mathbf{i}} \mathbf{B}^{l}_{\mathbf{i},\mathbf{j}} \; _1(k; \mathbf{A}^{(k)}_{\mathbf{i}})\right)\right) + B(l); \tag{7}$$

$$\text{where } \mathbf{B}^{l}_{\mathbf{i},\mathbf{j}} = \begin{cases} \mathbb{1}_{j\mathbf{i} \setminus \mathbf{j} j = l} & \text{if } l \quad 1 \\ 1 & \text{if } l = 0 \end{cases}; \tag{8}$$

where we omit the output constraint $\mathbb{1}_{jjj=l}$ for brevity, and introduce a binary scalar $\mathbf{B}^{l}_{\mathbf{i},\mathbf{j}}$ to write local ($l \quad 1$) and global ($l = 0$) interactions together.

Now we show that an EHNN-MLP layer can realize any EHNN layer:

Theorem 2. *An EHNN-MLP layer (Eq.* (7)*) can approximate any EHNN layer (Eq.* (6)*) to an arbitrary precision.*

The proof is done by leveraging the universal approximation property [25] to model appropriate functions with MLPs $_{1:3}$, so that the output of EHNN-MLP (Eq. (7)) accurately approximates the output of EHNN (Eq. (6)). As a result, with EHNN-MLP, we now have a practical model that can approximate the maximally expressive linear layer for general undirected hypergraphs.

In our implementation of the MLPs $_{1:3}$, we first transform the input order (k, l or l) into a continuous vector called *order embedding*, and combine it with the input feature through concatenation. This way, the order embeddings are served similarly as the positional encoding used in Transformer [53] with a subtle difference that it indicates the order of the input or output hyperedges. We employ sinusoidal encoding [53] to obtain order embedding due to its efficiency and, more importantly, to aid extrapolation to unseen hyperedge orders in testing.

Realization as a Transformer While EHNN-MLP (Eq. (7)) theoretically inherits the high expressive power of EHNN, in practice, its static sum-pooling can be limited in accounting for relative importance of input hyperedges. A solution for this is to introduce more sophisticated pooling. In particular, the attention mechanism of Transformers [53] was shown to offer a large performance gain in set and (hyper)graph modeling [13, 31, 32, 36] via dynamic weighting of input elements. Thus, we extend EHNN-MLP with multihead attention coefficients $^{h,l}_{\mathbf{i},\mathbf{j}}$ and introduce *EHNN-Transformer*, an advanced realization of EHNN:

$$\text{Attn}(\mathbf{A}^{(:K)})_{l,\mathbf{j}} = \; _3\left(l; \sum_{l \; 0} \; _2\left(l; \sum_{h=1}^{H} \sum_{k \; K} \sum_{\mathbf{i}} \; ^{h,l}_{\mathbf{i},\mathbf{j}} \; _1(k; \mathbf{A}^{(k)}_{\mathbf{i}}) w^{V}_{h}\right)\right); \tag{9}$$

$$\text{EHNN-Transformer}(\mathbf{A}^{(:K)}) = \text{Attn}(\mathbf{A}^{(:K)}) + \text{MLP}(\text{Attn}(\mathbf{A}^{(:K)})); \tag{10}$$

where we omit the output constraint $\mathbb{1}_{\mathbf{j}\mathbf{i}\mathbf{j}=l}$ and bias $B(l)$ for brevity. H denotes the number of heads and $W_h^V \in \mathbb{R}^{d \times d_v}$ denotes the value weight matrix. To compute attention coefficients $\alpha_{\mathbf{i},\mathbf{j}}^{h,l}$ from the input, we introduce additional query and key (hyper)networks $Q : \mathbb{N} \to \mathbb{R}^{H \times d_H}$ and $K : \mathbb{N} \times \mathbb{R}^d \to \mathbb{R}^{H \times d_H}$ and characterize scaled dot-product attention [53] as follows:

$$\alpha_{\mathbf{i},\mathbf{j}}^{h,l} = \begin{cases} \left(Q(l)_h K\left(l, \beta_1(k; \mathbf{A}_{\mathbf{i}}^{(k)})\right)_h^\top / \sqrt{d_H} \right) \mathbb{1}_{\mathbf{j}\mathbf{i}\setminus\mathbf{j}\mathbf{j}=l} & \text{if } l \neq 1 \\ \left(Q(0)_h K\left(l, \beta_1(k; \mathbf{A}_{\mathbf{i}}^{(k)})\right)_h^\top / \sqrt{d_H} \right) & \text{if } l = 0 \end{cases}; \tag{11}$$

where $\sigma(\cdot)$ denotes activation, often chosen as softmax normalization. Note that the query $Q(l)$ is agnostic to output index $\mathbf{j}$, following prior works on set and (hyper)graph attention [13, 36]. Although this choice of attention mechanism has a drawback that assigning importance to input ($\mathbf{i}$) depending on output ($\mathbf{j}$) is not straightforward, we choose it in favor of scalability.

Comparison to Message Passing Networks We finish the section by providing a comparative analysis of EHNNs with respect to the existing message passing networks for hypergraphs. We specifically compare against *AllSet* [13], as it represents a highly general framework that subsumes most existing hypergraph neural networks. Their MLP-based characterization *AllDeepSets* can be written with two MLPs β_1 and β_2 as follows:

$$\text{AllDeepSets}(\mathbf{A}^{(:K)})_{l,\mathbf{j}} = \mathbb{1}_{\mathbf{j}\mathbf{j}=l} \ \beta_2\left(\sum_k \sum_{\mathbf{i}} \mathbb{1}_{\mathbf{j}\mathbf{i}\setminus\mathbf{j}\mathbf{j}} \ \beta_1(\mathbf{A}_{\mathbf{i}}^{(k)}) \right). \tag{12}$$

We show the below by reducing EHNN-MLP to AllDeepSets through ablation:

Theorem 3. *An AllDeepSets layer (Eq.* (12)*) is a special case of EHNN-MLP layer (Eq.* (7)*), while the opposite is not true.*

Finally, Theorem 3 leads to the following corollary:

Corollary 1. *An EHNN-MLP layer is more expressive than an AllDeepSets layer and also all hypergraph neural networks that AllDeepSets subsumes.*

We provide an in-depth discussion including the comparison between EHNN-Transformer and AllSetTransformer [13] in Appendix A.2.

4 Experiments

We test EHNN on a range of hypergraph learning problems including synthetic node classification problem, real-world semi-supervised classification, and visual keypoint matching. For the real-world tasks, we use 10 semi-supervised classification datasets used in Chien et. al. [13] and two visual keypoint matching datasets used in Wang et. al. [55]. Details including the datasets and hyperparameters are in Appendix A.3. Additional experiments including comparative and ablation studies, and runtime and memory cost analysis are in Appendix A.4.

Table 1: Results for synthetic order-k hyperedge identification. We show averaged best test accuracy (%) over 5 runs with standard deviation.

	Test involves only seen k	Test involves unseen k	
		Interpolation	Extrapolation
AllDeepSets	76.99 ± 0.98	79.6 ± 0.86	79.01 ± 2.82
AllSetTransformer	77.61 ± 2.27	78.61 ± 2.367	77.35 ± 1.89
EHNN-MLP	98.02 ± 0.73	90.70 ± 2.90	85.65 ± 2.89
EHNN-Transformer	**99.69 ± 0.31**	**92.31 ± 1.47**	**90.19 ± 5.51**

4.1 Synthetic k-edge Identification

We devise a simple but challenging synthetic node classification task termed k-edge identification to demonstrate how the expressive power of EHNN can help learn complex hypergraph functions. In input hypergraph, we pick a random hyperedge and mark its nodes with a binary label. The task is to identify all other nodes whose hyperedge order is the same with the marked one. The model is required to propagate the information of marked hyperedge globally, while also reasoning about fine-grained structure of individual hyperedges for comparison. We use 100 train and 20 test hypergraphs, each with 100 nodes and randomly wired 10 hyperedges of orders $2 \{2; \dots; 10\}$. To further test generalization to unseen orders, we add two training sets where hyperedges are sampled *without* order-$\{5; 6; 7\}$ hyperedges (interpolation) or order-$\{8; 9; 10\}$ hyperedges (extrapolation). We evaluate the performance of EHNN-MLP/-Transformer with AllDeepSets and AllSetTransformer [13] as message passing baselines.

The test performances are in Table 1. EHNN achieves significant improvement over message passing nets, producing almost perfect prediction. The result advocates that even for simple tasks there are cases where high expressive power of a network is essential. Furthermore, we observe evidences that the model can interpolate or even extrapolate to unseen hyperedge orders. This supports the use of hypernetworks to infer parameters for potentially unseen orders.

4.2 Semi-supervised Classification

To test EHNN in real-world hypergraph learning, we use 10 transductive semi-supervised node classification datasets [13]. The data is randomly split into 50% training, 25% validation, and 25% test. We run the experiment 20 times with random splits and initialization, and report aggregated classification accuracy.

The test performances are in Table 2. Our methods often achieve favorable scores over strong baselines *e.g.*, AllDeepSets and AllSetTransformer – our models improve the state-of-the art by 3.27% in Walmart (1), 1.82% in House (0.6), and 1.54% in Walmart (0.6). Notably, EHNN-Transformer gives state-of-the-art performance in most cases. This supports the notion that attention strengthens equivariant networks [32, 36], and also implies that high expressiveness of EHNN makes it strong on general hypergraph learning setups involving not only social networks but also vision and graphics (NTU2012 and ModelNet40).

Table 2: Results for semi-supervised node classification. Average accuracy (%) over 20 runs are shown, and standard deviation can be found in Appendix A.3. Gray shade indicate the best result, and blue shade indicate results within one standard deviation of the best. Baseline scores are taken from Chien et. al. [13].

	Zoo	20Newsgroups	mushroom	NTU2012	ModelNet40	Yelp	House(1)	Walmart(1)	House(0.6)	Walmart(0.6)	avg. rank (#)
MLP	87.18	**81.42**	**100.00**	85.52	96.14	31.96	67.93	45.51	81.53	63.28	6.4
CEGCN	51.54	OOM	95.27	81.52	89.92	OOM	62.80	54.44	64.36	59.78	11.5
CEGAT	47.88	OOM	96.60	82.21	92.52	OOM	69.09	51.14	77.25	59.47	10.5
HNHN	93.59	81.35	**100.00**	89.11	97.84	31.65	67.80	47.18	78.78	65.80	5.9
HGNN	92.50	80.33	98.73	87.72	95.44	33.04	61.39	62.00	66.16	77.72	7.8
HCHA	93.65	80.33	98.70	87.48	94.48	30.99	61.36	62.45	67.91	77.12	8.1
HyperGCN	N/A	81.05	47.90	56.36	75.89	29.42	48.31	44.74	78.22	55.31	12.4
UniGCNII	93.65	81.12	99.96	89.30	98.07	31.70	67.25	54.45	80.65	72.08	5.8
HAN (full batch)	85.19	OOM	90.86	83.58	94.04	OOM	71.05	OOM	83.27	OOM	9.9
HAN (minibatch)	75.77	79.72	93.45	80.77	91.52	26.05	62.00	48.57	82.04	63.1	10.6
AllDeepSets	95.39	81.06	99.99	88.09	96.98	30.36	67.82	64.55	80.70	78.46	5.4
AllSetTransformer	**97.50**	81.38	**100.00**	88.69	98.20	**36.89**	69.33	65.46	83.14	78.46	2.4
EHNN-MLP	91.15	81.31	99.99	87.35	97.74	35.80	67.41	65.65	82.29	78.80	5.0
EHNN-Transformer	93.27	**81.42**	**100.00**	**89.60**	**98.28**	36.48	**71.53**	**68.73**	**85.09**	**80.05**	**1.6**

Table 3: Hypergraph matching accuracy (%) on Willow test set.

	car	duck	face	motor.	wine.	avg.
GMN	38.85	38.75	78.85	28.08	45.00	45.90
NGM	77.50	85.87	99.81	77.50	89.71	86.08
NHGM	69.13	83.08	99.81	73.37	88.65	82.81
NMGM	74.95	81.33	99.83	78.26	92.06	85.29
IPCA-GM	79.58	80.20	99.70	73.37	83.75	83.32
CIE-H	9.37	8.87	9.88	11.84	9.84	9.96
BBGM	96.15	90.96	**100.00**	96.54	**99.23**	96.58
GANN-MGM	92.11	90.11	**100.00**	96.21	98.26	95.34
NGM-v2	94.81	89.04	**100.00**	96.54	95.87	95.25
NHGM-v2	89.33	83.17	**100.00**	92.60	95.96	92.21
EHNN-MLP	94.71	91.92	**100.00**	97.21	97.79	96.33
EHNN-Transformer	**97.02**	**92.69**	**100.00**	**97.60**	98.08	**97.08**

Table 4: Hypergraph matching accuracy (%) on PASCAL-VOC test set.

	aero	bike	bird	boat	botl	bus	car	cat	chair	cow	desk
GMN	40.67	57.62	58.19	51.38	77.55	72.48	66.90	65.04	40.43	61.56	65.17
PCA-GM	51.46	62.43	64.70	58.56	81.94	75.18	69.56	71.05	44.53	65.81	39.00
NGM	12.09	10.01	17.44	21.73	12.03	21.40	20.16	14.26	15.10	12.07	14.50
NHGM	12.09	10.01	17.44	21.73	12.03	21.40	20.16	14.26	15.10	12.07	14.50
IPCA-GM	50.78	62.29	63.87	58.94	79.46	74.18	72.60	71.52	41.42	64.12	36.67
CIE-H	52.26	66.79	69.09	59.76	83.38	74.61	69.93	71.04	43.36	69.20	76.00
BBGM	**60.06**	71.32	78.21	78.97	88.63	**95.57**	**89.52**	80.53	**59.34**	**77.80**	76.00
GANN-MGM	14.75	32.20	21.31	24.43	67.23	36.35	21.09	17.20	25.73	21.00	37.50
NGM-v2	42.88	61.70	63.63	75.62	84.66	90.58	75.34	72.26	44.42	66.67	74.50
NHGM-v2	57.04	71.88	76.06	**79.96**	**89.79**	93.70	86.16	80.76	56.36	76.70	74.33
EHNN-MLP	57.34	**73.89**	76.41	78.41	89.40	94.51	85.58	79.83	56.39	76.56	**91.00**
EHNN-Transformer	60.04	72.36	**78.25**	78.59	87.61	93.77	87.99	**80.78**	58.76	76.29	81.17

	dog	horse	mbk	prsn	plant	sheep	sofa	train	tv	avg.
GMN	61.56	62.18	58.96	37.80	78.39	66.89	39.74	79.84	90.94	61.66
PCA-GM	67.82	65.18	65.71	46.21	83.81	70.51	49.88	80.87	93.07	65.36
NGM	12.83	12.05	15.69	09.76	21.00	17.10	15.12	31.11	24.88	16.52
NHGM	12.83	12.05	15.67	09.76	21.00	17.10	14.66	31.11	24.83	16.49
IPCA-GM	69.11	66.05	65.88	46.97	83.09	68.97	51.83	79.17	92.27	64.96
CIE-H	69.68	71.18	66.14	46.76	87.22	71.08	59.16	82.84	92.60	69.10
BBGM	**80.39**	77.80	76.48	**65.99**	**98.52**	**78.07**	76.65	97.61	94.36	**80.09**
GANN-MGM	16.16	20.16	25.92	19.20	53.76	18.34	26.16	46.30	72.32	30.85
NGM-v2	67.83	68.92	68.86	47.40	96.69	70.57	70.01	95.13	92.49	71.51
NHGM-v2	76.75	77.45	**76.81**	58.56	98.21	75.34	76.42	98.10	94.80	78.76
EHNN-MLP	76.57	**78.65**	75.54	58.92	98.31	76.53	**81.14**	98.08	**95.01**	79.90
EHNN-Transformer	78.30	76.91	75.79	63.78	97.60	76.47	78.04	**98.53**	93.83	79.74

4.3 Visual Keypoint Matching

To test EHNN in computer vision problems represented as hypergraph learning, we tackle visual keypoint matching. The task is considered challenging due to discrepancy between the two images in terms of viewpoint, scale, and lighting. Following previous work [55], we view the problem as *hypergraph matching* where keypoints of each image form an hypergraph. This is considered helpful as the hyperedge features can capture rotation- and scale-invariant geometric features such as angles. We then cast hypergraph matching to binary node classification on a single association hypergraph as in previous work [55].

We use two standard datasets [55]: Willow ObjectClass [14] and PASCAL-VOC [9, 17]. The Willow dataset consists of 256 images with 5 object categories. The PASCAL-VOC dataset contains 11,530 images with 20 object categories, and is considered challenging due to large variance in illumination and pose. We follow the training setup of NHGM-v2 [55] and only replace the hypergraph neural network module to EHNN-MLP/-Transformer. The key difference between NHGM-v2 and our models is that NHGM-v2 utilizes two *separate* message passing networks, one on 2-edges and another on 3-edges, and aggregates node features as a weighted sum. In contrast, EHNN *mixes* the information from 2-edges and 3-edges extensively via shared MLP hypernetworks and global interactions.

The results are in Table 3 and 4. On Willow, EHNN-Transformer gives the best performance, improving over NHGM-v2 by 4.87%. On PASCAL-VOC, EHNNs improve over NHGM-v2 by 1%, and are competitive to the best model (BBGM; 0.19% gap) that relies on sophisticated combinatorial solver [50]. We conjecture that intrinsic and global mix of 2-edge (distance) and 3-edge (angle) feature improves hypergraph learning and consequently also keypoint matching.

5 Conclusion

We proposed a family of hypergraph neural networks coined Equivariant Hypergraph Neural Network (EHNN). EHNN extends theoretical foundations of equivariant GNNs to general undirected hypergraphs by representing a hypergraph as a sequence of tensors and combining equivariant linear layers on them. We further proposed EHNN-MLP/-Transformer, practical realizations of EHNN based on MLP hypernetworks. We show that EHNN is theoretically more expressive than most message passing networks and provide empirical evidences.

Acknowledgement This work was supported by Institute of Information & communications Technology Planning & Evaluation (IITP) (No. 2021-0-00537, 2019-0-00075 and 2021-0-02068), the National Research Foundation of Korea (NRF) (No. 2021R1C1C1012540 and 2021R1A4A3032834), and Korea Meteorological Administration Research and Development Program "'Development of AI techniques for Weather Forecasting" under Grant (KMA2021-00121).

References

1. Albooyeh, M., Bertolini, D., Ravanbakhsh, S.: Incidence networks for geometric deep learning. arXiv (2019)
2. Arya, D., Gupta, D.K., Rudinac, S., Worring, M.: Hypersage: Generalizing inductive representation learning on hypergraphs. arXiv (2020)
3. Bai, S., Zhang, F., Torr, P.H.S.: Hypergraph convolution and hypergraph attention. Pattern Recognit. (2021)
4. Belongie, S.J., Malik, J., Puzicha, J.: Shape matching and object recognition using shape contexts. IEEE Trans. Pattern Anal. Mach. Intell. (2002)
5. Berend, D., Tassa, T.: Improved bounds on bell numbers and on moments of sums of random variables. Probability and Mathematical Statistics. (2010)
6. Berge, C., Minieka., E.: Graphs and Hypergraphs. North-Holland Publishing Company (1981)
7. Bevilacqua, B., Frasca, F., Lim, D., Srinivasan, B., Cai, C., Balamurugan, G., Bronstein, M.M., Maron, H.: Equivariant subgraph aggregation networks. In: ICLR (2022)
8. Botsch, M., Pauly, M., Kobbelt, L., Alliez, P., Lévy, B.: Geometric modeling based on polygonal meshes. In: Eurographics Tutorials (2008)
9. Bourdev, L., Malik, J.: Poselets: Body part detectors trained using 3d human pose annotations. In: ICCV (2009)
10. Bu, J., Tan, S., Chen, C., Wang, C., Wu, H., Zhang, L., He, X.: Music recommendation by unified hypergraph: Combining social media information and music content. In: Proceedings of the 18th International Conference on Multimedia 2010, Firenze, Italy, October 25-29, 2010 (2010)
11. Cai, C., Wang, Y.: A note on over-smoothing for graph neural networks. arXiv (2020)
12. Chen, Z., Chen, L., Villar, S., Bruna, J.: Can graph neural networks count substructures? In: NeurIPS (2020)
13. Chien, E., Pan, C., Peng, J., Milenkovic, O.: You are allset: A multiset function framework for hypergraph neural networks. In: ICLR (2022)
14. Cho, M., Alahari, K., Ponce, J.: Learning graphs to match. In: ICCV (2013)
15. Ding, K., Wang, J., Li, J., Li, D., Liu, H.: 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 2020, Online, November 16-20, 2020 (2020)
16. Dong, Y., Sawin, W., Bengio, Y.: HNHN: hypergraph networks with hyperedge neurons. arXiv (2020)
17. Everingham, M., Van Gool, L., Williams, C.K., Winn, J., Zisserman, A.: The pascal visual object classes (voc) challenge. International Journal of Computer Vision (2010)
18. Feng, Y., You, H., Zhang, Z., Ji, R., Gao, Y.: Hypergraph neural networks. In: AAAI (2019)
19. Gao, Y., Wang, M., Tao, D., Ji, R., Dai, Q.: 3-d object retrieval and recognition with hypergraph analysis. IEEE Trans. Image Process. (2012)
20. Gilmer, J., Schoenholz, S.S., Riley, P.F., Vinyals, O., Dahl, G.E.: Neural message passing for quantum chemistry. In: ICML (2017)
21. Gu, F., Chang, H., Zhu, W., Sojoudi, S., Ghaoui, L.E.: Implicit graph neural networks. In: NeurIPS (2020)

22. Gu, S., Yang, M., Medaglia, J.D., Gur, R.C., Gur, R.E., Satterthwaite, T.D., Bassett, D.S.: Functional hypergraph uncovers novel covariant structures over neurodevelopment. Human Brain Mapp. (2017)
23. Ha, D., Dai, A.M., Le, Q.V.: Hypernetworks. In: ICLR (2017)
24. Hamilton, W.L., Ying, Z., Leskovec, J.: Inductive representation learning on large graphs. In: NeurIPS (2017)
25. Hornik, K., Stinchcombe, M.B., White, H.: Multilayer feedforward networks are universal approximators. Neural Networks (1989)
26. Huang, J., Yang, J.: Unignn: a unified framework for graph and hypergraph neural networks. In: IJCAI (2021)
27. Ishiguro, K., ichi Maeda, S., Koyama, M.: Graph warp module: an auxiliary module for boosting the power of graph neural networks in molecular graph analysis. arXiv (2019)
28. Kalai, G.: Linear programming, the simplex algorithm and simple polytopes. Math. Program. (1997)
29. Keriven, N., Peyré, G.: Universal invariant and equivariant graph neural networks. In: NeurIPS (2019)
30. Kim, E., Kang, W., On, K., Heo, Y., Zhang, B.: Hypergraph attention networks for multimodal learning. In: CVPR (2020)
31. Kim, J., Nguyen, T.D., Min, S., Cho, S., Lee, M., Lee, H., Hong, S.: Pure transformers are powerful graph learners. arXiv (2022)
32. Kim, J., Oh, S., Hong, S.: Transformers generalize deepsets and can be extended to graphs and hypergraphs. In: NeurIPS (2021)
33. Klimm, F., Deane, C.M., Reinert, G., Estrada, E.: Hypergraphs for predicting essential genes using multiprotein complex data. Journal of Complex Networks (2021)
34. Knyazev, B., Taylor, G.W., Amer, M.R.: Understanding attention and generalization in graph neural networks. In: NeurIPS (2019)
35. Kofidis, E., Regalia, P.A.: On the best rank-1 approximation of higher-order supersymmetric tensors. Siam J. Matrix Anal. Appl (2002)
36. Lee, J., Lee, Y., Kim, J., Kosiorek, A.R., Choi, S., Teh, Y.W.: Set transformer: A framework for attention-based permutation-invariant neural networks. In: ICML (2019)
37. Li, D., Xu, Z., Li, S., Sun, X.: Link prediction in social networks based on hypergraph. In: 22nd International World Wide Web Conference, WWW '13, Rio de Janeiro, Brazil, May 13-17, 2013, Companion Volume (2013)
38. Li, J., Cai, D., He, X.: Learning graph-level representation for drug discovery. arXiv (2017)
39. Li, Q., Han, Z., Wu, X.: Deeper insights into graph convolutional networks for semi-supervised learning. In: AAAI (2018)
40. Louis, S.M., Zhao, Y., Nasiri, A., Wong, X., Song, Y., Liu, F., Hu, J.: Global attention based graph convolutional neural networks for improved materials property prediction. arXiv (2020)
41. Lowe, D.G.: Object recognition from local scale-invariant features. In: ICCV (1999)
42. Maron, H., Ben-Hamu, H., Serviansky, H., Lipman, Y.: Provably powerful graph networks. In: NeurIPS (2019)
43. Maron, H., Ben-Hamu, H., Shamir, N., Lipman, Y.: Invariant and equivariant graph networks. In: ICLR (2019)
44. Maron, H., Fetaya, E., Segol, N., Lipman, Y.: On the universality of invariant networks. In: ICML (2019)

45. Maron, H., Litany, O., Chechik, G., Fetaya, E.: On learning sets of symmetric elements. In: ICML (2020)
46. Milano, F., Loquercio, A., Rosinol, A., Scaramuzza, D., Carlone, L.: Primal-dual mesh convolutional neural networks. In: NeurIPS (2020)
47. Oono, K., Suzuki, T.: Graph neural networks exponentially lose expressive power for node classification. In: ICLR (2020)
48. Puny, O., Ben-Hamu, H., Lipman, Y.: From graph low-rank global attention to 2-fwl approximation. In: ICML (2020)
49. Ray, L.A.: 2-d and 3-d image registration for medical, remote sensing, and industrial applications. J. Electronic Imaging (2005)
50. Rolinek, M., Swoboda, P., Zietlow, D., Paulus, A., Musil, V., Martius, G.: Deep graph matching via blackbox differentiation of combinatorial solvers. In: ECCV (2020)
51. Serviansky, H., Segol, N., Shlomi, J., Cranmer, K., Gross, E., Maron, H., Lipman, Y.: Set2graph: Learning graphs from sets. In: NeurIPS (2020)
52. Tan, S., Bu, J., Chen, C., He, X.: Using rich social media information for music recommendation via hypergraph model. In: Social Media Modeling and Computing. Springer (2011)
53. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I.: Attention is all you need. In: NeurIPS (2017)
54. Velikovic, P.: Message passing all the way up. arXiv (2022)
55. Wang, R., Yan, J., Yang, X.: Neural graph matching network: Learning lawler's quadratic assignment problem with extension to hypergraph and multiple-graph matching. IEEE Transactions on Pattern Analysis and Machine Intelligence (2021)
56. Wu, Z., Jain, P., Wright, M.A., Mirhoseini, A., Gonzalez, J.E., Stoica, I.: Representing long-range context for graph neural networks with global attention. arXiv (2021)
57. Yadati, N., Nimishakavi, M., Yadav, P., Nitin, V., Louis, A., Talukdar, P.P.: Hypergcn: A new method for training graph convolutional networks on hypergraphs. In: NeurIPS (2019)
58. Yavartanoo, M., Hung, S., Neshatavar, R., Zhang, Y., Lee, K.M.: Polynet: Polynomial neural network for 3d shape recognition with polyshape representation. In: 3DV (2021)
59. Zaheer, M., Kottur, S., Ravanbakhsh, S., Póczos, B., Salakhutdinov, R., Smola, A.J.: Deep sets. In: NeurIPS (2017)
60. Zhang, M., Cui, Z., Neumann, M., Chen, Y.: An end-to-end deep learning architecture for graph classification. In: AAAI (2018)