A Appendix

A.1 Proofs

For the proof of Proposition 1 and Theorem 1 (Section 3.2), it is convenient that we first derive maximally expressive equivariant linear layers for *symmetric* input and output tensors. This is because all tensors under consideration in this work are symmetric due to the unorderedness of hypergraphs (see Definition 2).

Let us remind the maximally expressive equivariant linear layers characterized in Maron et. al. [43] (Section 2). Given an order-k input tensor $\mathbf{A} \in \mathbb{R}^{n^k \times d}$, the order-$l$ output tensor of an equivariant linear layer $L_{k\to l}$ is written as follows, with indicator $\mathbb{1}$ and multi-indices $\mathbf{i} \in [n]^k$, $\mathbf{j} \in [n]^l$:

$$L_{k\to l}(\mathbf{A})_{\mathbf{j}} = \sum_{\mu} \sum_{\mathbf{i}} \mathbb{1}_{(\mathbf{i},\mathbf{j})\in\mu} \mathbf{A}_{\mathbf{i}} w_{\mu} + \sum_{\lambda} \mathbb{1}_{\mathbf{j}\in\lambda} b_{\lambda}. \tag{13}$$

where $w_\mu \in \mathbb{R}^{d\times d'}$, $b_\lambda \in \mathbb{R}^{d'}$ are weight and biases, and μ and λ are equivalence classes of order-$(k+l)$ and order-l multi-indices, respectively. As briefly introduced in Section 2, the equivalence classes specify a *partitioning* of a multi-index space; the equivalence classes μ partition the space of order-$(k+l)$ multi-indices $[n]^{k+l}$, and λ partition the space of order-l multi-indices $[n]^l$. To be more specific, the equivalence classes are defined upon *equivalence relation* $\sim$ of multi-indices that specifies the partitioning $[n]^k/_\sim$ of multi-index space $[n]^k$. The equivalence relation $\sim$ is defined as follows: for $\mathbf{i},\mathbf{j} \in [n]^k$, the equivalence relation sets $\mathbf{i} \sim \mathbf{j}$ if and only if $(i_1, ..., i_k) = (\pi(j_1), ..., \pi(j_k))$ for some node permutation $\pi \in S_n$.

In the following Lemma, we show that if input and output of $L_{k\to l}$ (Eq. (13)) are constrained to be *symmetric*, the layer is described with *coarser* partitioning of index spaces (in terms of partition refinement) specified by equivalence relations that are invariant to axis permutation on top of node permutation:

Lemma 1. *If input and output of an equivariant linear layer $L_{k\to l}$ (Eq. (1)) are constrained to be symmetric tensors, it can be reduced to the following $L_{[k]\to[l]}$:*

$$L_{[k]\to[l]}(\mathbf{A})_{\mathbf{j}} = \sum_{\alpha} \sum_{\mathbf{i}} \mathbb{1}_{(\mathbf{i},\mathbf{j})\in\alpha} \mathbf{A}_{\mathbf{i}} w_{\alpha} + \sum_{\beta} \mathbb{1}_{\mathbf{j}\in\beta} b_{\beta}, \tag{14}$$

where α and β are equivalence classes that specify the partitioning $[n]^{k+l}/_{\sim_\alpha}$ and $[n]^l/_{\sim_\beta}$ respectively, defined as the following:

1. *The equivalence classes α are defined upon the equivalence relation $\sim_\alpha$ that, for $\mathbf{i},\mathbf{j} \in [n]^{k+l}$, relates $\mathbf{i} \sim_\alpha \mathbf{j}$ if and only if the following holds for some node permutation $\pi \in S_n$ and axis permutations $\pi_k \in S_k$, $\pi_l \in S_l$:*

$$(i_1, ..., i_{k+l}) = \left(\pi(j_{\pi_k(1)}), ..., \pi(j_{\pi_k(k)}), \pi(j_{k+\pi_l(1)}), ..., \pi(j_{k+\pi_l(l)})\right). \tag{15}$$

2. *The equivalence classes β are defined upon the equivalence relation $\sim_\beta$ that, for $\mathbf{i},\mathbf{j} \in [n]^l$, relates $\mathbf{i} \sim_\beta \mathbf{j}$ if and only if the following holds for some node permutation $\pi \in S_n$ and axis permutation $\pi_l \in S_l$:*

$$(i_1, ..., i_l) = \left(\pi(j_{\pi_l(1)}), ..., \pi(j_{\pi_l(l)})\right). \tag{16}$$

Proof. We begin from $L_{k\to l}$ (Eq. (13)) and reduce its parameters without affecting the output based on symmetry of input and output. For $\pi_k \in S_k$, we denote *axis permutation* of a multi-index $\mathbf{i} \in [n]^k$ as $\pi_k(i_1, ..., i_k) = (i_{\pi_k(1)}, ..., i_{\pi_k(k)})$ and denote axis permutation of a tensor $\mathbf{A} \in \mathbb{R}^{n^k \times d}$ as $(\pi_k \cdot \mathbf{A})_{\mathbf{i}} = \mathbf{A}_{\pi_k^{-1}(\mathbf{i})}$. With symmetry, input and output of $L_{k\to l}$ are constrained to be axis permutation invariant, *i.e.*, the following holds:

$$L_{k\to l}(\mathbf{A}) = \pi_l \cdot L_{k\to l}(\pi_k \cdot \mathbf{A}), \tag{17}$$

for all $\pi_k \in S_k$, $\pi_l \in S_l$, and symmetric $\mathbf{A} \in \mathbb{R}^{n^k \times d}$. In other words:

$$L_{k\to l}(\mathbf{A})_{\mathbf{j}} = \sum_{\mu} \sum_{\mathbf{i}} \mathbb{1}_{(\mathbf{i}, \pi_l^{-1}(\mathbf{j})) \in \mu} \mathbf{A}_{\pi_k^{-1}(\mathbf{i})} w_\mu + \sum_{\lambda} \mathbb{1}_{\pi_l^{-1}(\mathbf{j}) \in \lambda} b_\lambda, \tag{18}$$

which, denoting π_l^{-1} as π_l and noting π_k is a bijection on $[n]^k$, leads to following:

$$L_{k\to l}(\mathbf{A})_{\mathbf{j}} = \sum_{\mu} \sum_{\mathbf{i}} \mathbb{1}_{(\pi_k(\mathbf{i}), \pi_l(\mathbf{j})) \in \mu} \mathbf{A}_{\mathbf{i}} w_\mu + \sum_{\lambda} \mathbb{1}_{\pi_l(\mathbf{j}) \in \lambda} b_\lambda. \tag{19}$$

We now reduce biases. From Eq. (19), we see that by constraining the output to be symmetric, $\sum_\lambda \mathbb{1}_{\mathbf{j} \in \lambda} b_\lambda = \sum_\lambda \mathbb{1}_{\pi_l(\mathbf{j}) \in \lambda} b_\lambda$ holds for all $\mathbf{j} \in [n]^l$ and $\pi_l \in S_l$. We can see that this holds if and only if $b_{\lambda_1} = b_{\lambda_2}$ for all (λ_1, λ_2) such that, for some $\mathbf{j} \in \lambda_1$, $\pi_l(\mathbf{j}) \in \lambda_2$ holds for some $\pi_l \in S_l$. By writing $b_\beta = b_{\lambda_1} = b_{\lambda_2}$, we can interpret β as a new equivalence class of multi-indices formed as a union of all such $\{\lambda_1, \lambda_2, ...\}$ (thereby specifying a coarser partitioning than $\sim$), or equivalently, by collecting all multi-indices related by node and axis permutations, *i.e.*, equivalence relation $\sim_\beta$. Thus, we can reduce biases as $\sum_\beta \mathbb{1}_{\mathbf{j} \in \beta} b_\beta$.

We now reduce weights. From Eq. (19), we see that by constraining the input and output to be symmetric, $\sum_\mu \sum_{\mathbf{i}} \mathbb{1}_{(\mathbf{i},\mathbf{j}) \in \mu} \mathbf{A}_{\mathbf{i}} w_\mu = \sum_\mu \sum_{\mathbf{i}} \mathbb{1}_{(\pi_k(\mathbf{i}), \pi_l(\mathbf{j})) \in \mu} \mathbf{A}_{\mathbf{i}} w_\mu$ holds for all $\mathbf{i} \in [n]^k, \mathbf{j} \in [n]^l$, $\pi_k \in S_k$, $\pi_l \in S_l$, and symmetric $\mathbf{A} \in \mathbb{R}^{n^k \times d}$. Similar to biases, this holds if and only if $w_{\mu_1} = w_{\mu_2}$ for all (μ_1, μ_2) such that, for some $(\mathbf{i}, \mathbf{j}) \in \mu_1$, $(\mathbf{i}, \pi_l(\mathbf{j})) \in \mu_2$ holds for some $\pi_l \in S_l$. By writing $w_\beta = w_{\mu_1} = w_{\mu_2}$, we can interpret β as a new equivalence class of multi-indices formed as a union of all such $\{\mu_1, \mu_2, ...\}$, or equivalently, by collecting all multi-indices related by node and output axis permutations. On top of that, the symmetry of input tensor gives us another room to reduce the parameters. We can see that, for all (β_1, β_2) such that $(\pi_k(\mathbf{i}), \mathbf{j}) \in \beta_2$ holds for some $(\mathbf{i}, \mathbf{j}) \in \beta_1$ and $\pi_k \in S_k$, we can replace both w_{β_1} and w_{β_2} by $w_\alpha = (w_{\beta_1} + w_{\beta_2})/2$ and obtain exactly same output. Extending, for any collection $\{\beta_1, \beta_2, ...\}$ of all such pairwise related β's, we can replace $w_{\beta_1}, w_{\beta_2}, ...$ by $w_\alpha = (w_{\beta_1} + w_{\beta_2} + ...)/|\{\beta_1, \beta_2, ...\}|$ and obtain exactly same output. Thus, we can interpret α as a new equivalence class of multi-indices formed as a union of all such $\{\beta_1, \beta_2, ...\}$, or equivalently, by collecting all multi-indices related by node, input axis, and output axis permutations *i.e.*, equivalence relation $\sim_\alpha$. Thus, we can reduce weights as $\sum_\alpha \sum_{\mathbf{i}} \mathbb{1}_{(\mathbf{i},\mathbf{j}) \in \alpha} \mathbf{A}_{\mathbf{i}} w_\alpha$.

We now prove Proposition 1 and Theorem 1 (Section 3.2).

Proof of Proposition 1 (Section 3.2) We begin from a simple lemma:

Lemma 2. *For a symmetric order-k tensor $\mathbf{A}^{(k)}$ that represents a k-uniform hypergraph (Eq. (2)), all indices $\mathbf{i}$ of nonzero entries contain k distinct elements.*

Proof. In Eq. (2), recall that $\mathbf{A}^{(k)}_{(i_1,\dots,i_k)} \neq 0$ only if $\{i_1,\dots,i_k\} \in E^{(k)}$. As $E^{(k)}$ is a set of k-uniform hyperedges that contain k distinct node indices, every multi-indices $(i_1,\dots,i_k)$ of nonzero entries of $\mathbf{A}^{(k)}$ contains k distinct elements.

Now, we prove Proposition 1.

Proof. We begin from $L_{[k]\to[l]}$ in Eq. (14) and reduce the parameters without affecting the output. By further constraining the (already symmetric) input and output to symmetric tensors that represent uniform hypergraphs, we first write:

$$L_{(k)\to(l)}(\mathbf{A}^{(k)})_{\mathbf{j}} = \mathbb{1}_{|\mathbf{j}|=l}\left(\sum_{\alpha}\sum_{\mathbf{i}} \mathbb{1}_{(\mathbf{i},\mathbf{j})\in\alpha}\mathbb{1}_{|\mathbf{i}|=k}\mathbf{A}^{(k)}_{\mathbf{i}} w_\alpha + \sum_{\beta}\mathbb{1}_{\mathbf{j}\in\beta} b_\beta\right). \tag{20}$$

Note that two constraints are added, $\mathbb{1}_{|\mathbf{i}|=k}$ multiplied to the input and $\mathbb{1}_{|\mathbf{j}|=l}$ multiplied to the output. This comes from the fact that the input/output of the layer are order-k/l tensors that represent k/l-uniform hypergraphs, respectively. As Lemma 2 states, the input/output must contain nonzero entry only for indices that contain k/l distinct elements, respectively.

We first reduce biases. The constraint $\mathbb{1}_{|\mathbf{j}|=l}$ leaves only a single bias b_{β_l} of equivalence class β_l and discards the rest (*i.e.*, $|\mathbf{j}| = l$ if and only if $\mathbf{j} \in \beta_l$). This particular equivalence class β_l specifies that all multi-index entries $j_1,\dots,j_l$ are unique. For any other equivalence class $\beta' \neq \beta_l$, the partition that represents it ties at least two entries of $\mathbf{j} \in \beta'$ ($j_a = j_b$ for $a \neq b$). This gives $|\mathbf{j}| < l$ for all $\mathbf{j} \in \beta'$, which leads to $\mathbf{j} \notin \beta'$ for all $|\mathbf{j}| = l$. Thus we have $\mathbb{1}_{|\mathbf{j}|=l}\mathbb{1}_{\mathbf{j}\in\beta'} = 0$ for all $\mathbf{j}$, meaning $b_{\beta'}$ cannot affect the output in Eq. (20). We can safely remove all $\beta' \neq \beta_l$ from the bias, and the layer reduces to the following where $b_l = b_{\beta_l}$:

$$L_{(k)\to(l)}(\mathbf{A}^{(k)})_{\mathbf{j}} = \mathbb{1}_{|\mathbf{j}|=l}\left(\sum_{\alpha}\sum_{\mathbf{i}} \mathbb{1}_{(\mathbf{i},\mathbf{j})\in\alpha}\mathbb{1}_{|\mathbf{i}|=k}\mathbf{A}^{(k)}_{\mathbf{i}} w_\alpha + b_l\right). \tag{21}$$

We now reduce weights. Similar to bias, the joint constraint $\mathbb{1}_{|\mathbf{j}|=l}\mathbb{1}_{|\mathbf{i}|=k}$ leaves exactly $1+\min(k,l)$ weights and discards the rest. We derive this by removing all equivalence classes α that never affect the output. For any equivalence class α' that ties at least two entries within $\mathbf{i}$ or $\mathbf{j}$ for some $(\mathbf{i},\mathbf{j}) \in \alpha'$ ($i_a = i_b$ or $j_a = j_b$ for $a \neq b$), we have $|\mathbf{i}| < k$ or $|\mathbf{j}| < l$ for all $(\mathbf{i},\mathbf{j}) \in \alpha'$, which leads to $(\mathbf{i},\mathbf{j}) \notin \alpha'$ for all $|\mathbf{i}| = k$ and $|\mathbf{j}| = l$. Thus we have $\mathbb{1}_{|\mathbf{j}|=l}\mathbb{1}_{|\mathbf{i}|=k}\mathbb{1}_{(\mathbf{i},\mathbf{j})\in\alpha'} = 0$ for all $(\mathbf{i},\mathbf{j})$, meaning $w_{\alpha'}$ cannot affect the output in Eq. (20) and can be safely removed. We now have a reduced set of equivalence classes α such that for any $(\mathbf{i},\mathbf{j}) \in \alpha$, $|\mathbf{i}| = k$ and $|\mathbf{j}| = l$. Notably, due to axis permutation symmetry described in Lemma 1, we can see that each equivalence class α can be described exactly by

the number of equivalences $i_a = j_b$ between $\mathbf{i}$ and $\mathbf{j}$, which we denote as $\mathcal{I}$ (*i.e.*, if α is described by $\mathcal{I}$, $|\mathbf{i} \cap \mathbf{j}| = \mathcal{I}$ if and only if $(\mathbf{i}, \mathbf{j}) \in \alpha$). Thus, by discarding irrelevant equivalence classes and rewriting in terms of the overlap $\mathcal{I}$, we can reduce the weight as follows, where $w_{\mathcal{I}} = w_\alpha$ for α that corresponds to $\mathcal{I}$:

$$\sum_{\alpha} \sum_{\mathbf{i}} \mathbb{1}_{(\mathbf{i},\mathbf{j}) \in \alpha} \mathbf{A}_{\mathbf{i}}^{(k)} w_\alpha = \sum_{\mathcal{I}=0}^{\min(k,l)} \sum_{\mathbf{i}} \mathbb{1}_{|\mathbf{j}|=l} \mathbb{1}_{|\mathbf{i} \cap \mathbf{j}|=\mathcal{I}} \mathbb{1}_{|\mathbf{i}|=k} \mathbf{A}_{\mathbf{i}}^{(k)} w_{\mathcal{I}}. \tag{22}$$

With this, the layer in Eq. 21 reduces to:

$$L_{(k)\to(l)}(\mathbf{A}^{(k)})_{\mathbf{j}} = \mathbb{1}_{|\mathbf{j}|=l} \left(\sum_{\mathcal{I}=0}^{\min(k,l)} \sum_{\mathbf{i}} \mathbb{1}_{|\mathbf{i} \cap \mathbf{j}|=\mathcal{I}} \mathbf{A}_{\mathbf{i}}^{(k)} w_{\mathcal{I}} + b_l \right). \tag{23}$$

Note that $\mathbb{1}_{|\mathbf{i}|=k}$ in weight application was removed as $\mathbf{A}^{(k)}$ already contains nonzero entries only for $|\mathbf{i}| = k$ (see Lemma 2).

We finish by handling $\mathcal{I} = 0$ in weight application as a special case:

$$\begin{aligned}
&\mathbb{1}_{|\mathbf{j}|=l} \sum_{\mathcal{I}=0}^{\min(k,l)} \sum_{\mathbf{i}} \mathbb{1}_{|\mathbf{i} \cap \mathbf{j}|=\mathcal{I}} \mathbf{A}_{\mathbf{i}}^{(k)} w_{\mathcal{I}} \\
&\quad = \mathbb{1}_{|\mathbf{j}|=l} \left(\sum_{\mathcal{I}=1}^{\min(k,l)} \sum_{\mathbf{i}} \mathbb{1}_{|\mathbf{i} \cap \mathbf{j}|=\mathcal{I}} \mathbf{A}_{\mathbf{i}}^{(k)} w_{\mathcal{I}} + \sum_{\mathbf{i}} \mathbb{1}_{|\mathbf{i} \cap \mathbf{j}|=0} \mathbf{A}_{\mathbf{i}}^{(k)} w_0 \right) && (24) \\
&\quad = \mathbb{1}_{|\mathbf{j}|=l} \left(\sum_{\mathcal{I}=1}^{\min(k,l)} \sum_{\mathbf{i}} \mathbb{1}_{|\mathbf{i} \cap \mathbf{j}|=\mathcal{I}} \mathbf{A}_{\mathbf{i}}^{(k)} (w_{\mathcal{I}} - w_0) + \sum_{\mathbf{i}} \mathbb{1}_{|\mathbf{i} \cap \mathbf{j}| \geq 0} \mathbf{A}_{\mathbf{i}}^{(k)} w_0 \right) && (25) \\
&\quad = \mathbb{1}_{|\mathbf{j}|=l} \left(\sum_{\mathcal{I}=1}^{\min(k,l)} \sum_{\mathbf{i}} \mathbb{1}_{|\mathbf{i} \cap \mathbf{j}|=\mathcal{I}} \mathbf{A}_{\mathbf{i}}^{(k)} w'_{\mathcal{I}} + \sum_{\mathbf{i}} \mathbf{A}_{\mathbf{i}}^{(k)} w_0 \right). && (26)
\end{aligned}$$

This modification is not strictly required, but makes practical implementation much easier as $\mathcal{I} = 0$ case reduces to global pooling. By rewriting $w'_{\mathcal{I}}$ as $w_{\mathcal{I}}$, we finally arrive at the reduced layer in Eq. (3).

Proof of Theorem 1 (Section 3.2) As a direct proof upon tensor sequences is not straightforward, we first *pack* the entire sequence of tensors $\mathbf{A}^{(:K)}$ that represent a hypergraph into an equivalent symmetric order-K tensor $\mathbf{A}^{[K]} \in \mathbb{R}^{n^K \times d}$. Then, we show that the maximally expressive equivariant linear layer $L_{[K]\to[L]} : \mathbb{R}^{n^K \times d} \to \mathbb{R}^{n^L \times d}$ for the packed tensors is equivalent to $L_{(:K)\to(:L)}$ for tensor sequences, and thus they are maximally expressive.

We first characterize the packed tensors with the following lemma:

Lemma 3. *A sequence of tensors $\mathbf{A}^{(:K)} = \left(\mathbf{A}^{(k)}\right)_{k \leq K}$ that represent a hypergraph (Definition 3) can be represented as an equivalent symmetric order-K tensor $\mathbf{A}^{[K]} \in \mathbb{R}^{n^K \times d}$.*

Proof. Let us define $\mathbf{A}^{[K]} \in \mathbb{R}^{n^K \times d}$ as follows:

$$\mathbf{A}^{[K]}_{(i_1,...,i_K)} = \mathbf{A}^{(|\{i_1,...,i_K\}|)}_{\{i_1,...,i_K\}}, \tag{27}$$

where $\{i_1, ..., i_K\}$ denotes the set of unique entries in $(i_1, ..., i_K)$ ordered arbitrarily (note that arbitrary ordering always gives the same value as $\mathbf{A}^{(k)}$ is symmetric). From $\mathbf{A}^{[K]}$, we can retrieve the original sequence of tensors $\left(\mathbf{A}^{(k)}\right)_{k\leq K}$ by using all order-K multi-indices that contain k unique entries to index $\mathbf{A}^{[K]}$ to construct each $\mathbf{A}^{(k)}$.

Now, we prove Theorem 1.

Proof. We prove by showing the equivalence of $L_{[K]\to[L]}$ (Eq. (14)) to the collection $\left(L_{(k)\to(l)}\right)_{k\leq K,l\leq L}$ (Eq. (23)).

Let us begin from the bias. Due to the symmetry of output of $L_{[K]\to[L]}$, the set of all $\mathbf{j} \in [n]^L$ with l unique entries specify a single equivalence class β, which we write β_l. With this, the biases in Eq. (14) can be rewritten as follows:

$$\sum_{\beta} \mathbb{1}_{\mathbf{j}\in\beta} b_\beta = \sum_{l\leq L} \mathbb{1}_{|\mathbf{j}|=l} b_{\beta_l} \tag{28}$$

This is equivalent to the collection of bias applications in $\left(L_{(k)\to(l)}\right)_{k\leq K,l\leq L}$.

We now move to the weight. Due to the symmetry of input and output of $L_{[K]\to[L]}$, the set of all $(\mathbf{i}, \mathbf{j})$ for $\mathbf{i} \in [n]^K, \mathbf{j} \in [n]^L$ with k unique entries in $\mathbf{i}$, l unique entries in $\mathbf{j}$, and $\mathcal{I}$ tying relations $i_a = i_b = ... = j_c = j_d$ specify a single equivalence class α, which we write $\alpha_{k,l,\mathcal{I}}$. With this, the weights in Eq. (14) can be rewritten as follows:

$$\sum_{\alpha}\sum_{\mathbf{i}} \mathbb{1}_{(\mathbf{i},\mathbf{j})\in\alpha} \mathbf{A}^{[K]}_{\mathbf{i}} w_\alpha = \sum_{k\leq K}\sum_{l\leq L}\sum_{\mathcal{I}=0}^{\min(K,L)}\sum_{\mathbf{i}} \mathbb{1}_{|\mathbf{j}|=l}\mathbb{1}_{|\mathbf{i}\cap\mathbf{j}|=\mathcal{I}}\mathbb{1}_{|\mathbf{i}|=k}\mathbf{A}^{[K]}_{\mathbf{i}} w_{\alpha_{k,l,\mathcal{I}}} \tag{29}$$

This is computationally equivalent to the collection of weight applications in $\left(L_{(k)\to(l)}\right)_{k\leq K,l\leq L}$.

Proof of Theorem 2 (Section 3.4)

Proof. Our target of approximation is the following, the output of weight application from Eq. (6) (Section 3.3):

$$w(\mathbf{A}^{(:K)})_{l,\mathbf{j}} = \sum_{\mathcal{I}=0}^{K}\sum_{k\leq K}\sum_{\mathbf{i}} \mathbf{B}^{\mathcal{I}}_{\mathbf{i},\mathbf{j}} \mathbf{A}^{(k)}_{\mathbf{i}} \mathcal{W}(k,l,\mathcal{I}), \tag{30}$$

$$\text{where } \mathbf{B}^{\mathcal{I}}_{\mathbf{i},\mathbf{j}} = \begin{cases} \mathbb{1}_{|\mathbf{i}\cap\mathbf{j}|=\mathcal{I}} & \text{if } \mathcal{I} \geq 1 \\ 1 & \text{if } \mathcal{I} = 0 \end{cases}. \tag{31}$$

Note that we moved the summation $\sum_{\mathcal{I}=1}^{\min(k,l)}$ out of $\sum_{k\leq K}$, and adjusted the summation range to $\sum_{\mathcal{I}=1}^{K}$. This does not change the output because for any $\mathcal{I} \geq \min(k,l)$ we have $\mathbb{1}_{|\mathbf{i}\cap\mathbf{j}|=\mathcal{I}} = 0$.

We now introduce MLPs $\phi_1 : \mathbb{N} \times \mathbb{R}^d \to \mathbb{R}^{Kd}$, $\phi_2 : \mathbb{N} \times \mathbb{R}^{Kd} \to \mathbb{R}^{(1+K)Kd}$, and $\phi_3 : \mathbb{N} \times \mathbb{R}^{(1+K)Kd} \to \mathbb{R}^d$ to model appropriate functions based on universal approximation [25].

First, we make ϕ_1 output the following, where ϵ_1 is approximation error:

$$\phi_1(k, \mathbf{X})_{(k-1)d+1:kd} = \mathbf{X} + \epsilon_1, \tag{32}$$

so that $\phi_1(0, \mathbf{X})$ places $\mathbf{X}$ on the first d channels of the output, $\phi_1(1, \mathbf{X})$ places $\mathbf{X}$ on the $d{+}1 : 2d$ channels of the output, and so on. Then, $\sum_{k\leq K} \sum_{\mathbf{i}} \mathbf{B}^{\mathcal{I}}_{\mathbf{i},\mathbf{j}} \phi_1(k, \mathbf{A}^{(k)}_{\mathbf{i}})$ gives channel concatenation of $\left(\sum_{\mathbf{i}} \mathbf{B}^{\mathcal{I}}_{\mathbf{i},\mathbf{j}} \mathbf{A}^{(k)}_{\mathbf{i}}\right)_{k\leq K}$, which we denote $\mathbf{Y}_{\mathcal{I}}$.

Then, we make ϕ_2 output the following, where ϵ_2 is approximation error:

$$\phi_2(\mathcal{I}, \mathbf{Y})_{\mathcal{I}Kd+1:(\mathcal{I}+1)Kd} = \mathbf{Y} + \epsilon_2. \tag{33}$$

Then, $\sum_{\mathcal{I}=0}^{K} \phi_2(\mathcal{I}, \mathbf{Y}_{\mathcal{I}})$ gives a concatenation of $\mathbf{Y}_{0:K}$, which is equivalently concatenation of $\left(\left(\sum_{\mathbf{i}} \mathbf{B}^{\mathcal{I}}_{\mathbf{i},\mathbf{j}} \mathbf{A}^{(k)}_{\mathbf{i}}\right)_{k\leq K}\right)_{\mathcal{I}\leq K}$.

Finally, we let ϕ_3 output the following, where ϵ_3 is approximation error:

$$\phi_3(l, \mathbf{Z}) = \sum_{\mathcal{I}=0}^{K} \sum_{k\leq K} \mathbf{Z}_{(\mathcal{I}K+k-1)d+1:(\mathcal{I}K+k)d} \mathcal{W}(k, l, \mathcal{I}) + \epsilon_3, \tag{34}$$

which, by indexing back through $\mathbf{Z}$, $\mathbf{Y}$, and $\mathbf{X}$, gives $w(\mathbf{A}^{(:K)})_{l,\mathbf{j}}$.

We finish by rewriting $\phi_3(l, \mathbf{Z})$ as follows:

$$\phi_3(l, \mathbf{Z}) = \phi_3\left(l, \sum_{\mathcal{I}\geq 0} \phi_2\left(\mathcal{I}, \sum_{k\leq K} \sum_{\mathbf{i}} \mathbf{B}^{\mathcal{I}}_{\mathbf{i},\mathbf{j}} \phi_1(k, \mathbf{A}^{(k)}_{\mathbf{i}})\right)\right), \tag{35}$$

which is equivalent to the output of ϕ_3 of EHNN-MLP in Eq. (7). Therefore, with ϕ_1, ϕ_2, and ϕ_3 approximating the functions in Eq. (32), Eq. (33), and Eq. (34) respectively, the output of ϕ_3 of EHNN-MLP can approximate the output of weight application of EHNN (Eq. (6)) to arbitrary precision[3].

Proof of Theorem 3 (Section 3.4)

Proof. We can reduce an EHNN-MLP layer to an AllDeepSets layer with following procedure. First, from Eq. (7), we let $\phi_1(k, \mathbf{X}) = \phi'_1(\mathbf{X})$, $\phi_3(l, \mathbf{X}) = \phi'_3(\mathbf{X})$, and $\mathcal{B}(l) = 0$ to remove conditioning on hyperedge orders k and l. Then we let $\phi_2(\mathcal{I}, \mathbf{X}) = \mathbb{1}_{\mathcal{I}}\mathbf{X}$ to ablate the global interaction ($\mathcal{I} = 0$) and remove conditioning on $\mathcal{I} \geq 1$. By renaming ϕ'_1 to ϕ_1 and ϕ'_3 to ϕ_2, we get the AllDeepSets layer (Eq. (12)). Yet, an AllDeepSets layer cannot reduce to an EHNN-MLP layer as it cannot model interactions between non-overlapping hyperedges ($\mathcal{I} = 0$).

[3] The overall error depend on the approximation error of MLPs (ϕ_1, ϕ_2, ϕ_3) at each step, and uniform bounds and continuity of the modeled functions [25].

A.2 In-Depth Comparison of EHNN and AllSet (Section 3.4)

Note that in the proof of Theorem 3, we leveraged the simple property that message passing of AllDeepSets cannot account for long-range dependency modeling of EHNN. Then a natural question arises here: if we rule out the presence of global interaction, can the *local message passing* of EHNN be potentially better than that of current message passing networks? Our analysis suggests so. More specifically, in practical scenarios when local message aggregation cannot be multiset universal, the explicit hyperedge order conditioning of EHNN might help. To demonstrate this, we bring a powerful, Transformer-based characterization of AllSet framework called AllSetTransformer:

$$\text{AllSetAttn}(\mathbf{A}^{(:K)})_{l,\mathbf{j}} = \mathbb{1}_{|\mathbf{j}|=l} \sum_{h=1}^{H} \sum_{k \le K} \sum_{\mathbf{i}} \boldsymbol{\alpha}^{h}_{\mathbf{i},\mathbf{j}} \phi_1(\mathbf{A}^{(k)}_{\mathbf{i}}), \tag{36}$$

$$\text{AllSetTransformer}(\mathbf{A}^{(:K)}) = \text{AllSetAttn}(\mathbf{A}^{(:K)}) + \text{MLP}(\text{AllSetAttn}(\mathbf{A}^{(:K)})), \tag{37}$$

with H the number of heads, and attention coefficients $\boldsymbol{\alpha}^{h}_{\mathbf{i},\mathbf{j}}$ computed with a query matrix $Q \in \mathbb{R}^{H \times d_H}$, a key network $\mathcal{K} : \mathbb{R}^d \to \mathbb{R}^{H \times d_H}$, and activation $\sigma(\cdot)$:

$$\boldsymbol{\alpha}^{h}_{\mathbf{i},\mathbf{j}} = \sigma\left(Q_h \mathcal{K}\left(\mathbf{A}^{(k)}_{\mathbf{i}}\right)^{\top}_{h} / \sqrt{d_H} \cdot \mathbb{1}_{|\mathbf{i} \cap \mathbf{j}| = \mathcal{I}} \right). \tag{38}$$

An interesting fact here is that when we take the activation for attention $\sigma(\cdot)$ as *any* kinds of normalization including softmax, attention mechanism of AllSetTransformer can fail to model even trivial multiset functions. A simple example is that they cannot *count* the number of input hyperedges that overlap with each output hyperedge. As a simple demonstration, if input hypergraph is a node set $\mathbf{A}^{(1)} \in \mathbb{R}^{n \times d}$ with identical features $\mathbf{A}^{(1)}_i = \mathbf{1}$, an AllSetTransformer layer that outputs features of hyperedges E cannot learn node counting *i.e.*, for output $\mathbf{Y}$, $\mathbf{Y}_e = |e|$ for $e \in E$. This is because, as attention normalizes over keys, all outputs of AllSetAttn (Eq. (36)) is always $H\phi_1(\mathbf{1})$. On the other hand, the hyperedge order-aware message passing of EHNN-Transformer can easily solve the problem by forwarding l.

Let us finish with stronger expressiveness of EHNN-Transformer (Eq. (10)) compared to AllSetTransformer:

Theorem 4. *An AllSetTransformer layer (Eq.* (37)*) is a special case of EHNN-Transformer layer (Eq.* (10)*), while the opposite is not true.*

Proof. We can reduce an EHNN-Transformer layer to an AllSetTransformer layer with following procedure. First, from Eq. (9), we let $\phi_1(k, \mathbf{X}) = \phi'_1(\mathbf{X})$, $\phi_3(l, \mathbf{X}) = \mathbf{X}$, and $\mathcal{B}(l) = 0$ to remove conditioning on hyperedge orders k and l. Then we let $\phi_2(\mathcal{I}, \mathbf{X}) = \mathbb{1}_{\mathcal{I}}\mathbf{X}$ to ablate the global interaction ($\mathcal{I} = 0$) and remove conditioning on $\mathcal{I} \ge 1$. Lastly, from Eq. (11), we let $\mathcal{Q}(\mathcal{I}) = Q$ and $\mathcal{K}(\mathcal{I}, \mathbf{X}) = \mathcal{K}(\mathbf{X})$. By renaming ϕ'_1 to ϕ_1, we get the AllSetTransformer layer (Eq. (37)). Yet, an AllSetTransformer cannot reduce to an EHNN-Transformer as it cannot model interactions between non-overlapping hyperedges ($\mathcal{I} = 0$).

Table 5: Statistics of the datasets.

(a) Statistics of k-edge identification dataset.

Dataset	Test involves only seen k	Test involves unseen k	
		Interpolation	Extrapolation
# nodes	100	100	100
# edges	10	10	10
train edge orders	2-10	2-4, 8-10	2-7
test edge orders	2-10	2-10	2-10

(b) Statistics of node classification dataset.

Dataset	Zoo	20Newsgroups	Mushroom	NTU2012	ModelNet40	Yelp	House	Walmart
# nodes	101	16242	8124	2012	12311	50758	1290	88860
# edges	43	100	298	2012	12311	679302	341	69906
# feature	16	100	22	100	100	1862	100	100
# class	7	4	2	67	40	9	2	11

(c) Statistics of visual keypoint matching dataset.

Dataset	Willow	PASCAL-VOC
# categories	5	20
# images	256	11,530
# keypoints/image	¿10	6-23

A.3 Experimental Details (Section 4)

We provide detailed experimental details including dataset statistics and hyperparameter search procedure. We provide dataset statistics in Table 5 and optimal hyperparameter settings in Table 6.

Synthetic k-edge Identification For synthetic k-edge identification, we used small datasets composed of 100 train and 20 test hypergraphs, each with 100 nodes and randomly wired 10 hyperedges. In input hypergraph, we pick a random hyperedge and mark its nodes with a binary label. The task is to identify (classify) every other nodes whose hyperedge order is the same with the marked one. For default training set, we sample hyperedges of orders $\in \{2, ..., 10\}$. To further test generalization of models to unseen orders, we use two additional training sets where hyperedges are sampled *without* order-$\{5, 6, 7\}$ hyperedges (interpolation) or order-$\{8, 9, 10\}$ hyperedges (extrapolation) as in Table 5a. For all models, we use simple two-layer architecture that first converts the node labels to hyperedge features ($V \to E$) then maps them back to nodes ($E \to V$) for classification. For all models we set hidden dimension to 64, and for EHNN-Transformer and AllSetTransformer we set number of attention heads to 4.

Table 6: Optimal hyperparameters for each dataset. lr, wd, d, h each refers to learning rate, weight decay, hidden dimension, and number of heads. do_{mlp} refers to dropout rate on MLP and do_{local}/do_{global} refers to dropout rate on local/global interactions. d_c refers to classifier hidden dimension.

(a) Optimal hyperparameter for node classification datasets.

	EHNN-MLP				EHNN-Transformer							
	lr	wd	d	do_{mlp}	lr	wd	h	d	do_{mlp}	do_{local}	do_{global}	d_c
Zoo	0.001	0	128	0	0.001	0	8	256	0	0	0	128
20Newsgroups	0.001	$1e^{-5}$	256	0	0.001	$1e^{-5}$	8	256	0	0	0	64
Mushroom	0.001	$1e^{-5}$	128	0	0.001	$1e^{-5}$	4	256	0	0	0	128
NTU2012	0.001	0	256	0.2	0.001	$1e^{-5}$	8	256	0.2	0.1	0.1	64
ModelNet40	0.001	$1e^{-5}$	256	0.2	0.001	$1e^{-5}$	4	256	0	0	0	64
Yelp	0.001	0	64	0	0.001	0	8	64	0	0	0	128
House(1)	0.001	0	256	0.2	0.001	$1e^{-5}$	4	64	0.2	0	0	64
Walmart(1)	0.001	0	256	0.2	0.001	0	8	256	0.2	0	0	64
House(0.6)	0.001	0	128	0	0.001	0	4	256	0	0	0	64
Walmart(0.6)	0.001	$1e^{-5}$	256	0.2	0.001	$1e^{-5}$	8	128	0.2	0	0	64

(b) Optimal hyperparameter for visual keypoint matching datasets.

	EHNN-MLP				EHNN-Transformer						
	lr	wd	d	do_{mlp}	lr	wd	h	d	do_{mlp}	do_{local}	do_{global}
Willow	$2e^{-4}$	0.5	32	0	$2e^{-4}$	0.5	4	32	0.1	0	0.5
PASCAL-VOC	$2e^{-4}$	0.5	32	0	$2e^{-4}$	0.5	4	128	0	0	0

Table 7: Results for semi-supervised node classification. Average accuracy (%) over 20 runs are shown with standard deviation.

	Zoo	20Newsgroups	mushroom	NTU2012	ModelNet40	Yelp	House(1)	Walmart(1)	House(0.6)	Walmart(0.6)	avg. rank (↑)
MLP	87.18 ± 4.44	**81.42 ± 0.49**	**100.00 ± 0.00**	85.52 ± 1.49	96.14 ± 0.36	31.96 ± 0.44	67.93 ± 2.33	45.51 ± 0.24	81.53 ± 2.26	63.28 ± 0.37	6.4
CEGCN	51.54 ± 11.19	OOM	95.27 ± 0.47	81.52 ± 1.43	89.92 ± 0.46	OOM	62.80 ± 2.61	54.44 ± 0.24	64.36 ± 2.41	59.78 ± 0.32	11.5
CEGAT	47.88 ± 14.03	OOM	96.60 ± 1.67	82.21 ± 1.23	92.52 ± 0.39	OOM	69.09 ± 3.00	51.14 ± 0.56	77.25 ± 2.53	59.47 ± 1.05	10.5
HNHN	93.59 ± 5.88	81.35 ± 0.61	**100.00 ± 0.01**	89.11 ± 1.44	97.84 ± 0.25	31.65 ± 0.44	67.80 ± 2.59	47.18 ± 0.35	78.78 ± 1.88	65.80 ± 0.39	5.9
HGNN	92.50 ± 4.58	80.33 ± 0.42	98.73 ± 0.32	87.72 ± 1.35	95.44 ± 0.33	33.04 ± 0.62	61.39 ± 2.96	62.00 ± 0.24	66.16 ± 1.80	77.72 ± 0.21	7.8
HCHA	93.65 ± 6.15	80.33 ± 0.80	98.70 ± 0.39	87.48 ± 1.87	94.48 ± 0.28	30.99 ± 0.72	61.36 ± 2.53	62.45 ± 0.26	67.91 ± 2.26	77.12 ± 0.26	8.1
HyperGCN	N/A	81.05 ± 0.59	47.90 ± 1.04	56.36 ± 4.86	75.89 ± 5.26	29.42 ± 1.54	48.31 ± 2.93	44.74 ± 2.81	78.22 ± 2.46	55.31 ± 0.30	12.4
UniGCNII	93.65 ± 4.37	81.12 ± 0.67	99.96 ± 0.05	89.30 ± 1.33	98.07 ± 0.23	31.70 ± 0.52	67.25 ± 2.57	54.45 ± 0.37	80.65 ± 1.96	72.08 ± 0.28	5.8
HAN (full batch)	85.19 ± 8.18	OOM	90.86 ± 2.40	83.58 ± 1.46	94.04 ± 0.41	OOM	71.05 ± 2.26	OOM	83.27 ± 1.62	OOM	9.9
HAN (minibatch)	75.77 ± 7.10	79.72 ± 0.62	93.45 ± 1.31	80.77 ± 2.36	91.52 ± 0.96	26.05 ± 1.37	62.00 ± 9.06	48.57 ± 1.04	82.04 ± 2.68	63.1 ± 0.96	10.6
AllDeepSets	95.39 ± 4.77	81.06 ± 0.54	99.99 ± 0.02	88.09 ± 1.52	96.98 ± 0.26	30.36 ± 1.57	67.82 ± 2.40	64.55 ± 0.33	80.70 ± 1.59	78.46 ± 0.26	5.4
AllSetTransformer	**97.50 ± 3.59**	81.38 ± 0.58	**100.00 ± 0.00**	88.69 ± 1.24	98.20 ± 0.20	**36.89 ± 0.51**	69.33 ± 2.20	65.46 ± 0.25	83.14 ± 1.92	78.46 ± 0.26	2.4
EHNN-MLP	91.15 ± 6.13	81.31 ± 0.43	99.99 ± 0.03	87.35 ± 1.42	97.74 ± 0.21	35.80 ± 0.77	67.41 ± 2.83	65.65 ± 0.36	82.29 ± 1.87	78.80 ± 0.18	5.0
EHNN-Transformer	93.27 ± 6.59	**81.42 ± 0.53**	**100.00 ± 0.00**	**89.60 ± 1.36**	**98.28 ± 0.18**	36.48 ± 0.40	**71.53 ± 2.59**	**68.73 ± 0.35**	**85.09 ± 2.05**	**80.05 ± 0.27**	**1.6**

Semi-supervised Classification We use 8 datasets used in Chien et. al. [13] (Table 5b). Among them, three datasets (20Newsgroups, Mushroom, Zoo) are from the UCI Categorical Machine Learning Repository, and two (ModelNet40, NTU2012) are from computer vision domain where the objective is to classify visual objects. Other three (Yelp, House, Walmart) were crafted in Chien et. al. [13]. In Yelp, the nodes correspond to restaurants where the node labels correspond to the number of stars provided in the yelp "restaurant" catalog. For House,

each node and label is a member of the US House of Representatives and their political party. Nodes of Walmart represent products where node label corresponds to product category, and set of products purchased together are tied with a hyperedge. Since House and Walmart does not contain node features, the node features are created by adding Gaussian noise to one-hot node labels. We fix node feature dimension to 100 as in prior work.

We use a composition of two layers that maps node features to hyperedge features ($V \to E$) and maps the features back to nodes ($E \to V$) as a module, and use a single module for EHNN-MLP and use a stack of two modules for EHNN-Transformer. For hyperparameter search, we fix learning rate as 0.001 and run grid search over hidden dimension {64, 128, 256, 512}, weight decays {0, 0.00001}, and MLP dropout {0, 0.2}. For EHNN-Transformer, we also search number of heads over {4, 8} for multi-head attention, attention output dropout {0, 0.1}, and classifier hidden dimension {64, 128}. The final selection of hyperparameters based on grid search are outlined in Table 6a.

Along with the average accuracy reported in the main text, we report the standard deviation over 20 runs with random train/val/test splits and model initialization in Table 7.

Visual Keypoint Matching For evaluation under inductive setting, we test the performance of EHNN on keypoint matching benchmarks implemented in the ThinkMatch repository [55]. We borrow two real image datasets, Willow ObjectClass [14] and PASCAL-VOC [9,17] with Berkeley annotations (Table 5c), as well as provided train/test splitting pipelines from the repository. For training, we use binary cross entropy loss between two permutation matrices: one from ground-truth matching and another from node classification on the association hypergraph. Performance at test time is evaluated by measuring the matching accuracy via F1-score. We compare our EHNN methods against 10 different methods, by reproducing their performance based on implementation in ThinkMatch[4]. Several methods are excluded due to numerical instability errors. For EHNN methods, we follow the same keypoint feature extraction and association hypergraph construction procedure as in NHGM-v2: we simply replace the local message-passing GNN module in NHGM-v2 with an EHNN-MLP/Transformer.

To find optimal hyperparameters for EHNN-MLP, we run grid search over hidden dimension sizes {16, 32, 64, 96, 128}. For the Transformer variant, we search through number of layers {1, 2, 3} and number of attention heads {2, 4, 8, 16} in addition to hidden dimension sizes. For EHNN-Transformer we also check applying dropouts within $[0.1, 0.5]$ separately to global and local attention outputs respectively. The final selected hyperparameters are outlined in Table 6b. For training EHNN methods, we use the default setting used for NHGM-v2 in ThinkMatch. For Willow, we train for 10 epochs with learning rate that starts at $2e^{-4}$ and decays into half at epoch 2. For PASCAL-VOC, we train for 20 epochs with learning rate that also starts $2e^{-4}$ and decays into half at epoch 2.

[4] `https://github.com/Thinklab-SJTU/ThinkMatch`

Table 8: Ablation on k-edge identification (5 runs each).

	Global interaction	Order info.	MLP realization	Seen k	Unseen k Interpolation	Unseen k Extrapolation
AllDeepSets [13]	×	×	◯	76.99±0.98	79.6±0.86	79.01±2.82
EHNN-MLP (ablated)						
• w/o global and order	×	×	◯	78.94±1.18	78.6±1.66	77.97±1.83
• w/o global interaction	×	◯	◯	80.34±2.87	77.86±2.38	79.56±3.03
• w/o order embedding	◯	×	◯	84.05±1.77	81.30±3.56	80.17±3.34
EHNN (Section 3.3)						
• Lookup table for $\mathcal{W}, \mathcal{B}$	◯	◯	×	87.09±2.49	84.09±1.29	80.20±2.21
• Hypernetwork for $\mathcal{W}, \mathcal{B}$	◯	◯	×	83.74±2.89	83.19±1.57	79.93±2.61
EHNN-MLP (Section 3.4)	◯	◯	◯	**98.02±0.73**	**90.70±2.90**	**85.65±2.89**

Table 9: Runtime and memory cost (20 runs each).

	Forward (ms)	Backward (ms)	Peak mem. (MB)
AllDeepSets [13]	5.538±0.689	3.470±0.071	4.608±0.000
AllSetTransformers [13]	6.883±0.657	5.037±0.681	5.077±0.000
EHNN (Section 3.3)			
• Lookup table for $\mathcal{W}, \mathcal{B}$	28.68±0.535	71.71±3.597	14.43±0.000
• Hypernetwork for $\mathcal{W}, \mathcal{B}$	31.56±1.110	76.78±6.044	17.40±0.000
EHNN-MLP (Section 3.4)	7.517±0.865	7.179±0.548	7.361±0.000
EHNN-Transformer (Section 3.4)	14.51±1.146	13.41±0.0962	11.98±0.000

A.4 Additional Experiments

We report additional experimental results that could not be included in the main text due to space restriction.

Comparative and Ablation Experiments We perform a comprehensive ablation experiment on k-edge identification (Section 4.1) by gradually ablating each component of EHNN-MLP (Section 3.4) until it reduces to message passing (≈AllDeepSet [13]). We also compare it against EHNN (Section 3.3) which is maximally expressive but, unlike EHNN-MLP, is not realized as 3 elementwise MLPs. The results are in Table 8. We find ablating any component of EHNN-MLP degrades performance until similar to AllDeepSet.

Time and Memory Cost Analysis We perform runtime and memory cost analysis on an A100 GPU using random hypergraphs with 1024 nodes and 128 hyperedges with orders$\sim \mathcal{U}(2, 10)$. Results are in Table 9. While naïve EHNN (Section 3.3) suffers from high cost, EHNN-MLP/Transformer (Section 3.4) are significantly more efficient, improving time and memory cost from $5-20\times$ to $2-3\times$ w.r.t. highly optimized message passing while still maximally expressive.