Crystal Hypergraph Neural Networks

A Universal Framework for Material System Machine Learning

Alex Heilman, Weiyi Gong, Qimin Yan

Northeastern University

October 13, 2025

Overview

- Crystal graphs , how to update graph representations , limitations of graph structures

- Crystal hypergraphs , how to update hypergraphs , examples of hyperedge types in crystals

- Comparative testing for different sets of hyperedge types

Machine Learning on Crystal Systems

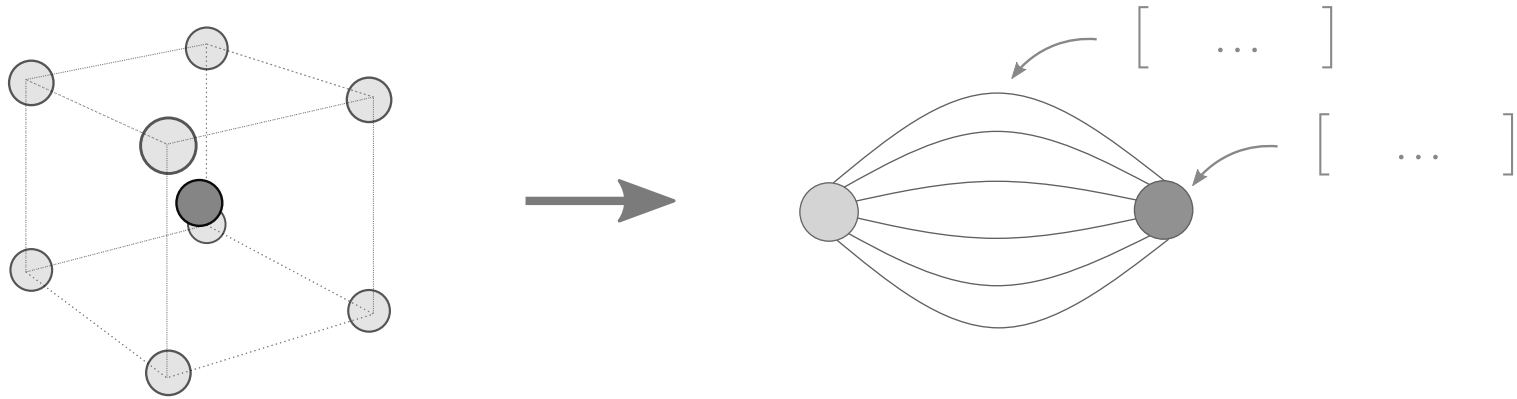
To perform predictive tasks for material systems, such as crystalline structures, we essentially need two things:

- A way to represent the material system mathematically
- A trainable predictive model or set of functions which takes, as input, the material system's representation

(Of course, we also need a large set of data)

Crystal Graph Construction

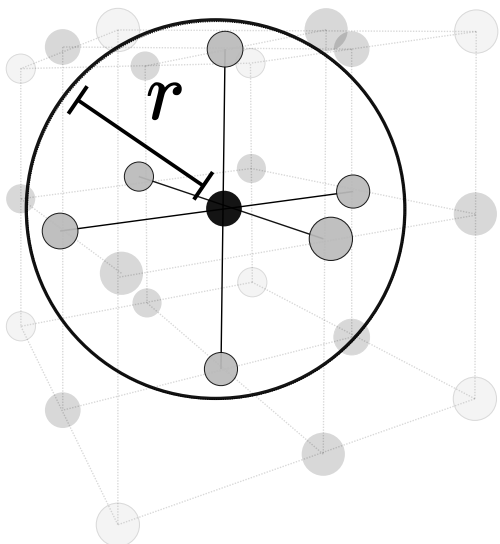
The usual technique is to represent crystalline systems as graphs (nodes and edges) [1]:



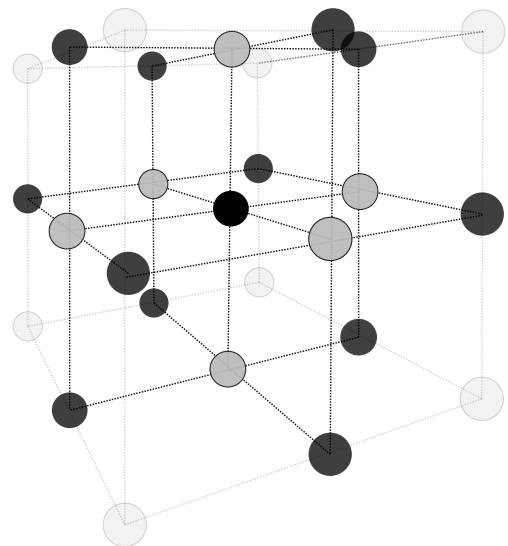
with atomic features associated with nodes and geometric features associated with edges

Crystal Graphs cont. I

Edges can be determined simply by distance cutoff (4 Ang.) and/or maximum number of neighbors for each node (12)



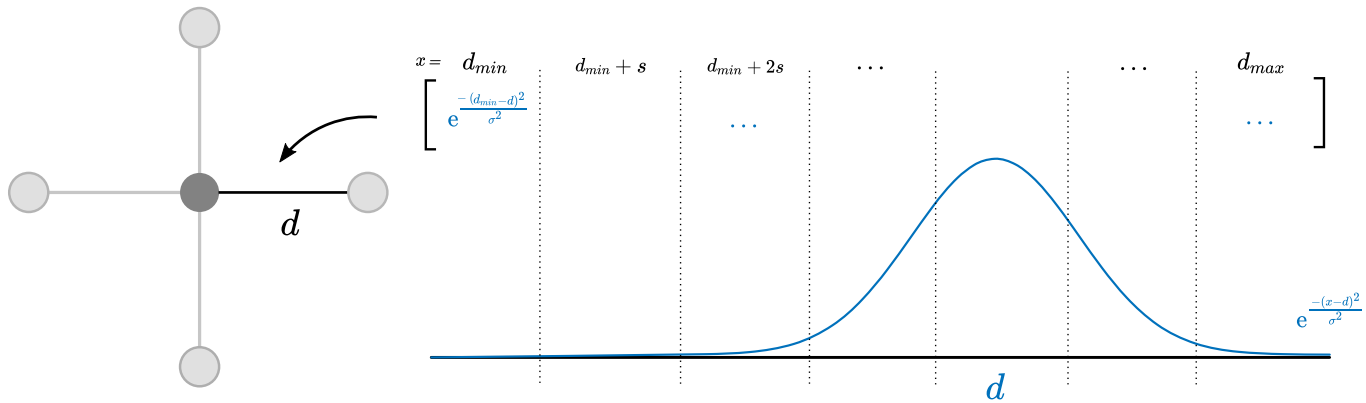
(within) Max radius r



N nearest neighbors

Crystal Graphs cont. II

Edge attributes then are constructed as a Gaussian expansion of interatomic distance:



Message Passing on Graphs

Now, we need to update these features through some trainable function that acts on graph representations.

This is most generally accomplished by a message passing network [2] applied to graph representations.

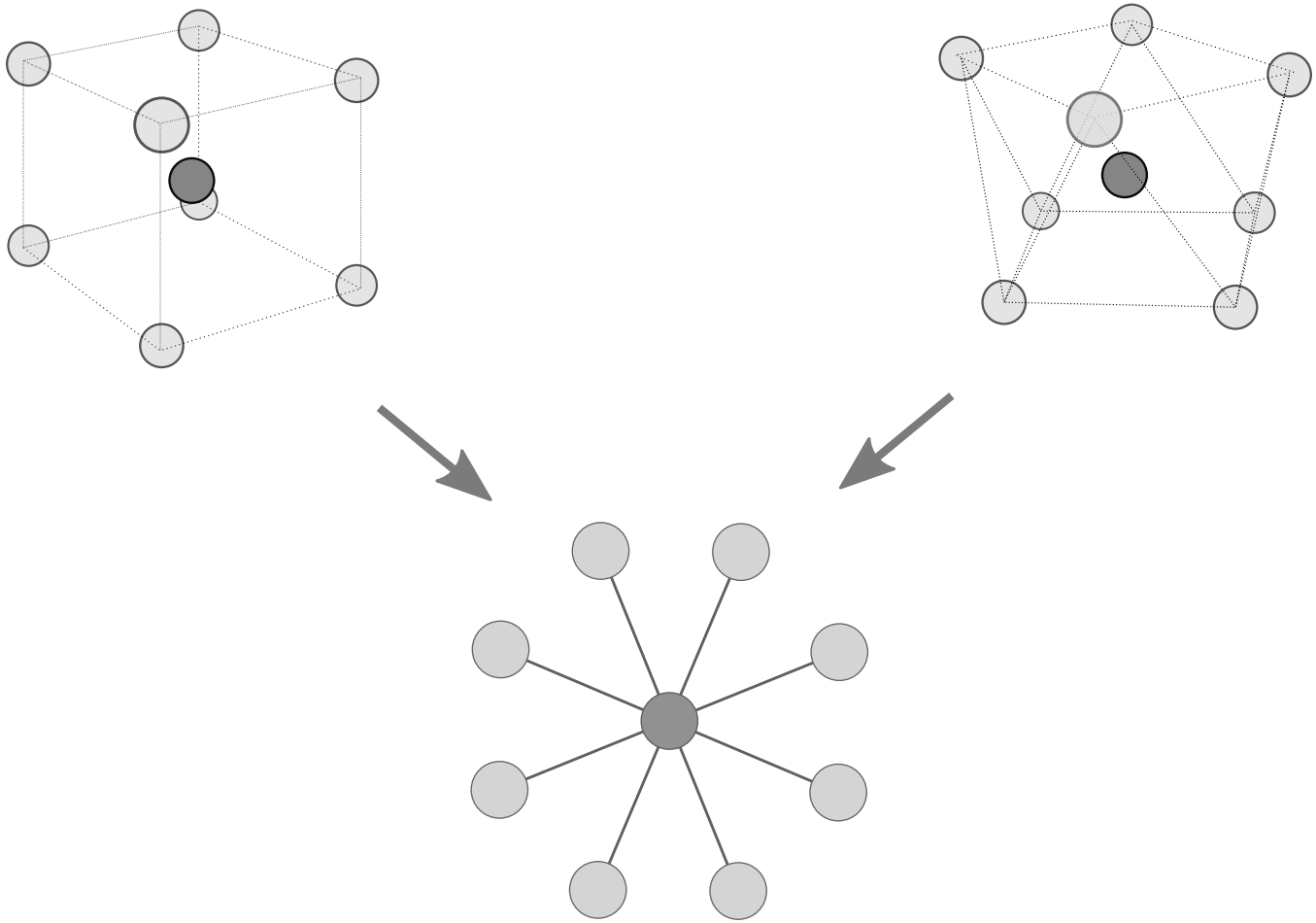
$$m_i^{t+1} = \sum_{n_j \in \mathcal{N}(i)} M_t(n_i^t, e_{ij}, n_j^t)$$

$$n_i^{t+1} = U_t(n_i^t, m_i^{t+1})$$

Here, each node n from layer t to $t+1$ is updated according to an update function U, which takes as input messages formed from each pair of nodes containing the node to be updated.

Graph Limitations

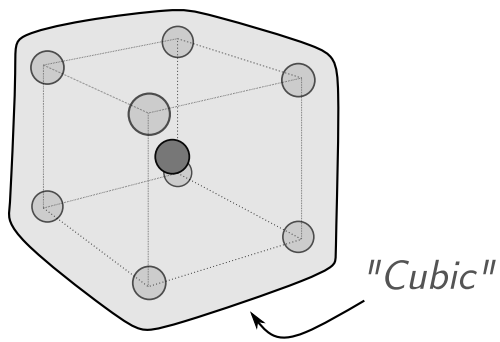
Problem: Our underlying representation encodes only distances between atoms!

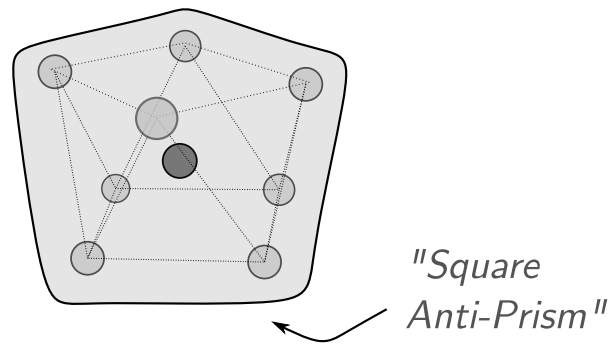


So, as a rough example, the above two crystalline structures would have the same representations!

Solution: Hypergraphs!

Hypergraphs allow us to have edges containing more than (or less than) two nodes.

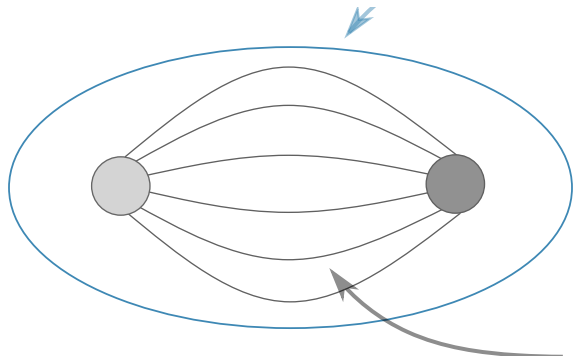




So, in a crystal hypergraph, we may define edges that explicitly describe this higher-order local geometry.

Crystal Hypergraphs

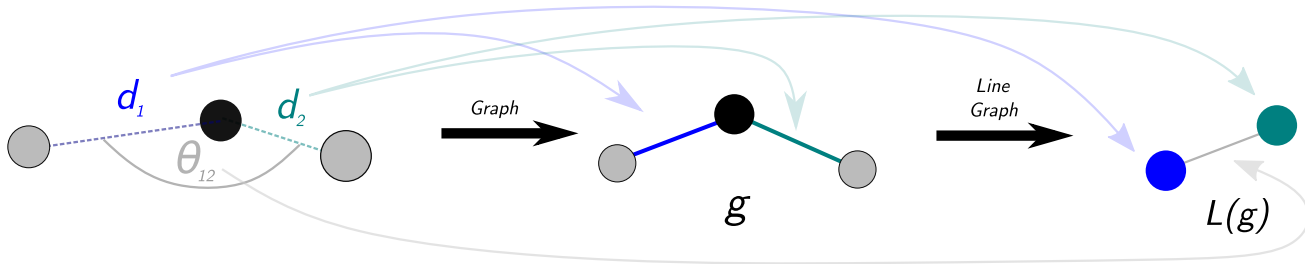
Hypergraphs give us a natural way to encode features with higher order physical structure, where hypergraph nodes still represent atoms of the underlying crystal structure.



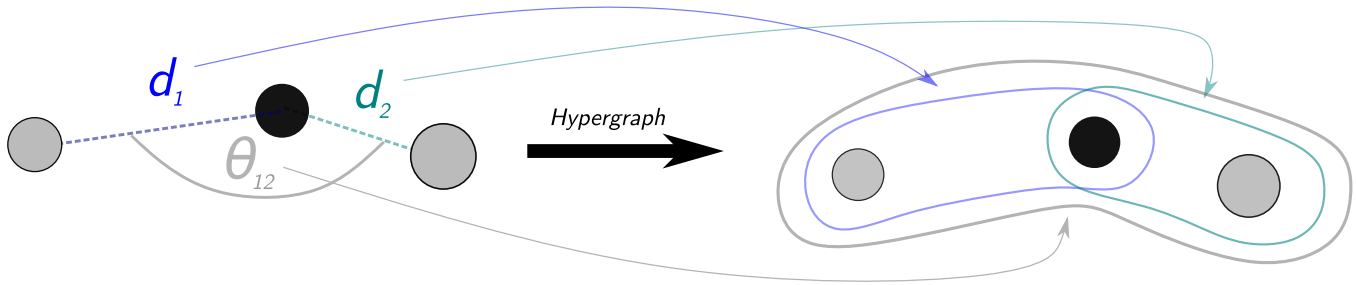
In a crystal hypegraph, we treat all different order structures (bonds, triplets, motifs, cells) in crystals on equal footing: each has a corresponding hyperedge type with a certain set of features (distance, angle, order parameters, point group).

Triplets as Hyperedges

Many modern models utilize 'triplet' information, specifically the angle between bonds, as the edge feature in a derived 'line-graph' in which nodes represent edges and edges represent overlapping bonds.



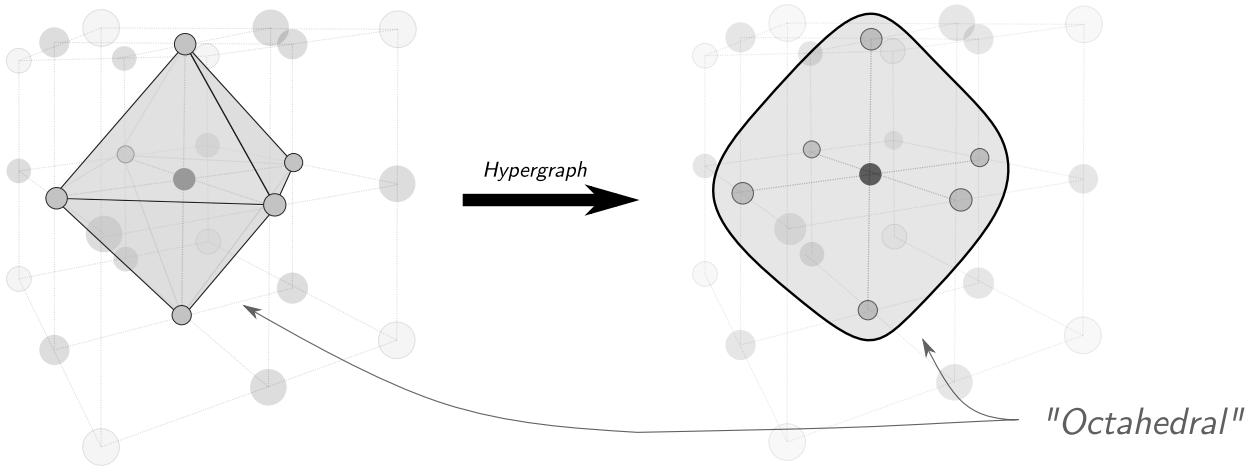
This may be thought of more simply in terms of hyperedges of order three, formed again from overlapping bonds. The hyperedge features then are the angle formed between the bonds.



Local Environments as Hyperedges

Returning to our previous problem, how may we more simply incorporate the missing local geometrical information?

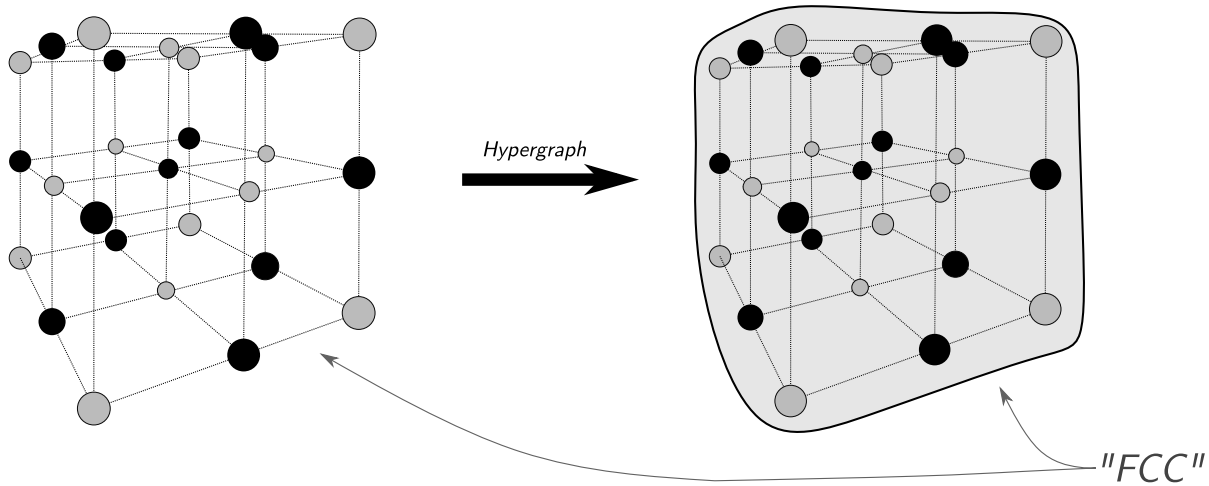
We encode this lost geometrical information in larger hyperedges:



Where these hyperedges (of variable order, but generally > 1) contain the entire first shell of neighbors. The relevant geometrical information can then be encoded quantitatively with continuous symmetry measures, local structure order parameters, etc.

Cell Hyperedges

Another order of hyperedge we may consider is that describing the entire unit cell of some crystalline structure.



These 'unit' cell hyperedges allow for the explicit inclusion of global crystalline structure properties, i.e. point group information.

This unit cell feature also may be learned through convolution and potentially used as a dynamic state vector.

Extending Message Passing to Hypergraphs

Now, we need a suitable convolutional structure that applies to hypergraphs...

In the case of a hypergraph, the neighborhood of nodes relevant to each message are now a set (instead of the single neighboring node feature of classic MPNNs)

$$m_i^{t+1} = \sum_{h_j \ni x_i} M_t(n_i^t, \underbrace{h_j^t, \{n_w^t | n_w \in h_j\}}_{e_{ij}, n_j})$$

$$n_i^{t+1} = U_t(n_i^t, m_i^{t+1})$$

Furthermore, we may want to update nodes AND hyperedge features with different convolutional functions for different hyperedge types.

Three Approaches to Hypergraph Convolution

Three different approaches to message forming for hypergraphs were tested, each applicable to the updating of both node features and hyperedge features.

1. Naive Relatives Graph Convolution: This approach constructs a graph from the hypergraph

$$n_i' = n_i + f(n_i \oplus m_j)$$

2. Interorder Hypergraph Convolution: This approach constructed a message for each node in each hyperedge (per origin node). It results in a generalization of message passing that exactly reproduces the latter in the case of only edges (of order 2, bonds)

$$n_i' = n_i + \sum_{n_k \in m_j} f(n_i \oplus m_j \oplus n_k)$$

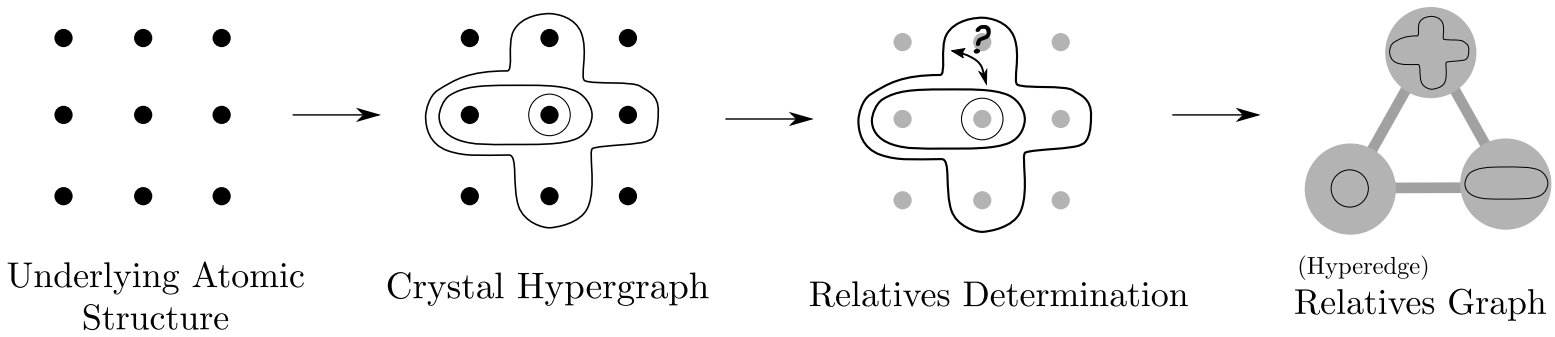
Three Approaches cont.

3. Hyperedge Aggregating Convolution: This approach constructs one message for each hyperedge (per origin node)

$$n_i' = n_i + f\left(n_i \oplus m_j \oplus \mathrm{AGG}\{n_k \in m_j\}\right)$$

1. Relatives Graph Convolution

The first approach, and perhaps the simplest, converts the initial hypergraph into a heterogeneous graph in which nodes represent hyperedges.



This is essentially a line graph for hyperedges.
Regular graph convolution may then be applied to this hyperedge 'relatives' graph, via it's *edge index* (representing connections between hyperedge 'nodes'), resulting in node updates of the form below:

$$x_i \rightarrow x_i + f_m(x_i \oplus m_j)$$

1. Relatives Graph Convolution cont.

This approach has the benefit of working out-of-the-box with most current convolutional structures, but suffers in performance from a lack of neighbor features in it's form of messages.

Note also that the number of messages here scales as $\mathcal{O}(nm)$, with n the number of hyperedges and m the average number of nodes in hyperedges. This is similar to the scaling of a typical MPNN.

2. Interorder Hypergraph Convolution

The second approach utilizes just the hypergraph, defined by way of *hyperedge-relations indices*:

[[origin-node-index, ...],
[connecting-hyperedge-index, ...],
[destination-node-index, ...]]

From these indices, messages may be read off in complete analogy to the usual graph case:

$$x_i \to x_i + \sum_{x_k \in m_j} f_m(x_i \oplus m_j \oplus x_k)$$

This approach is nice in that it completely generalizing the usual message passing framework, but the number of messages scales as $\mathcal{O}(nm^2)$.

3. Hyperedge Aggregation

To deal with the variable sized neighborhoods of features, we may also simply aggregate the neighborhood features in the message passing phase . Then, we essentially have the following form of convolution:

$$x_i \to x_i + f_m(x_i \oplus m_j \oplus \text{AGG}(\{n_w | n_w \in m_j\}))$$

This requires only that we define a *hyperedge index* of dimension 2:

$$[[\text{node-index}, ...],$$
$$[\text{hyperedge-index}, ...]]$$

This also has the advantage that messages only scale as $\mathcal{O}(nm)$; as well as that, after aggregation, most existing graph convolutional structures may be applied.

Comparative Performance Testing

The hyperedge aggregation method seems to be the best balance in terms of performance and computational cost.

As a proof of concept, we implement a basic convolutional structure amenable to the hyperedge aggregation scheme (**3**) above, and compare performance between different sets of hyperedges.

Here, we focus on performance between models with bond hyperedges and motif hyperedges.

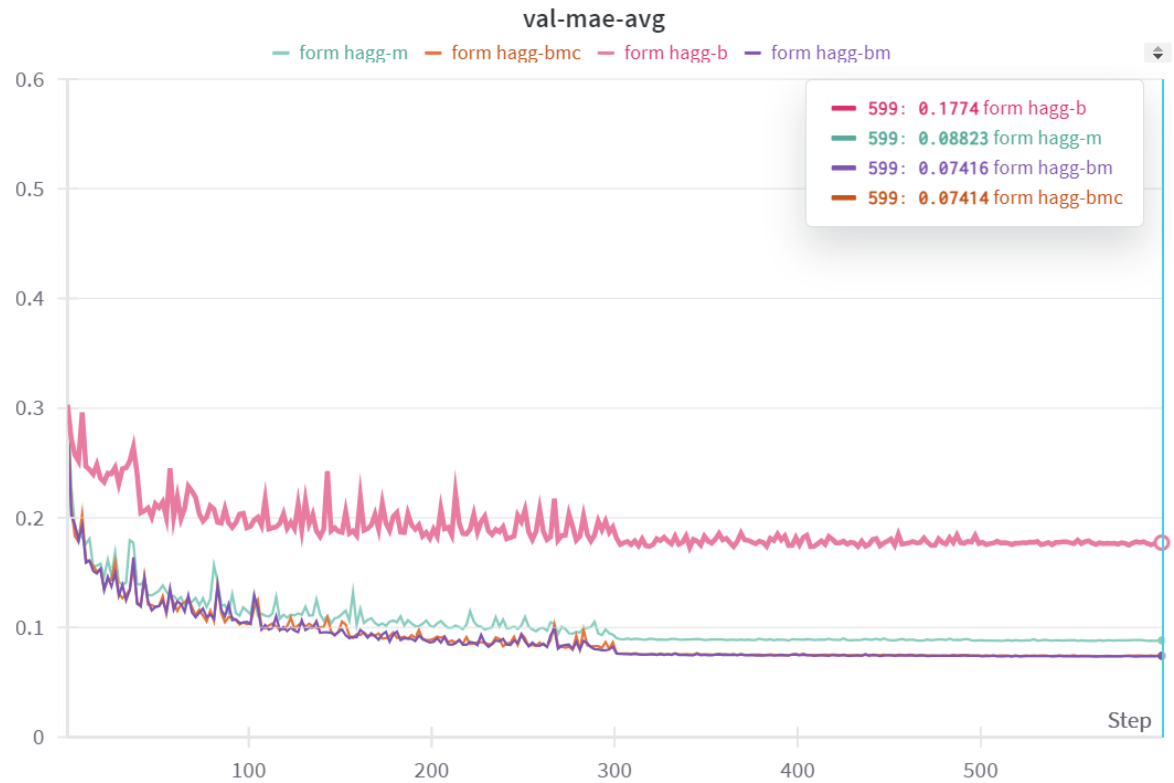
Hyperparameters and Model Architecture

For each convolutional structure, testing was done for a model with 3 convolutional layers, an initial learning rate of 0.01, hidden node features of dimension 64, and a hidden output layer of dimension 128 (Similar to CGCNN's architecture). The loss function utilized is MSE.

Datasets are split 80% for training and 20% for validation tests.

(Materials-Project) Formation Energy

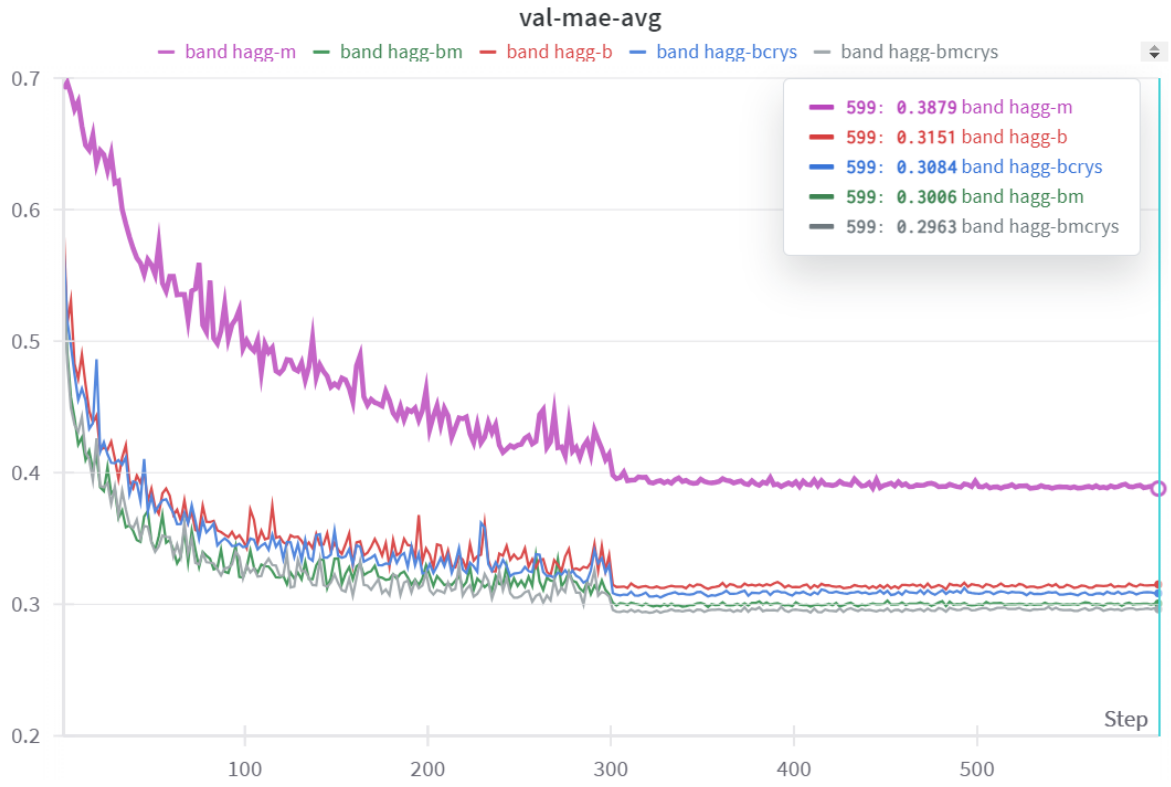
Below, validation MAE is shown through training for formation energy. The total dataset includes 152,605 materials.



Model	Best MAE (eV/Atom)
Bond-only	0.177
Motif-only	0.088
Bond & Motif	0.074

(Materials-Project) Band Gap

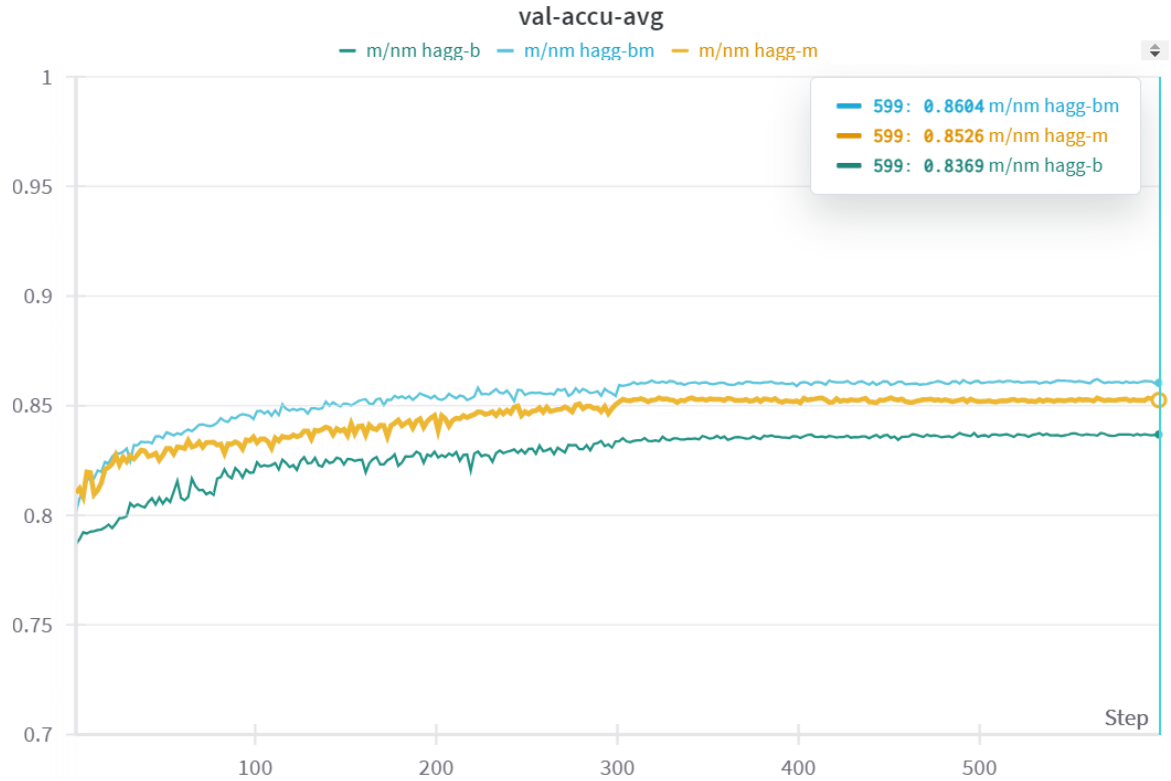
Below are results for band gap training, based on a dataset including 152,605 materials.



Model	Best MAE (eV)	w/ Crystal Feature
Bond-only	0.315	0.3084
Motif-only	0.387	
Bond & Motif	0.301	0.2963

(Materials-Project) Metal/Non-metal

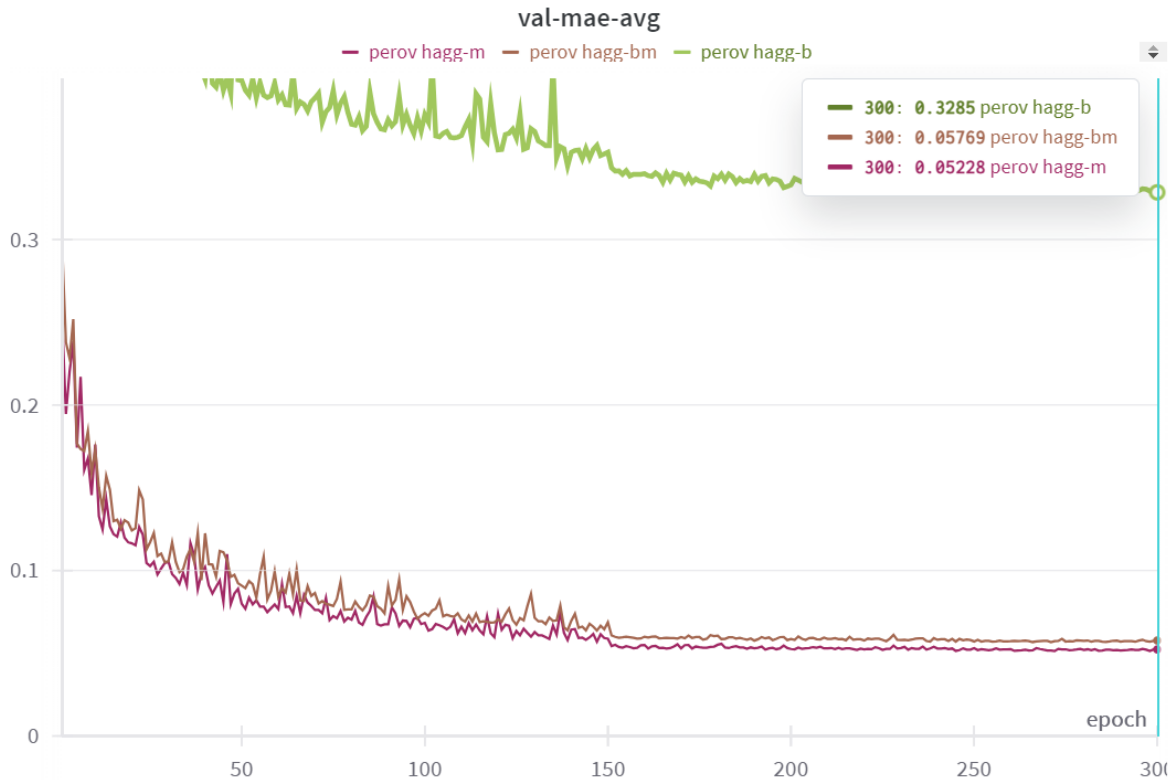
Below, we test metal/non-metal classification for 152,605 materials.



Model	Best Accuracy
Bond-only	.837
Motif-only	.853
Bond & Motif	.860

(MatBench) Perovskites

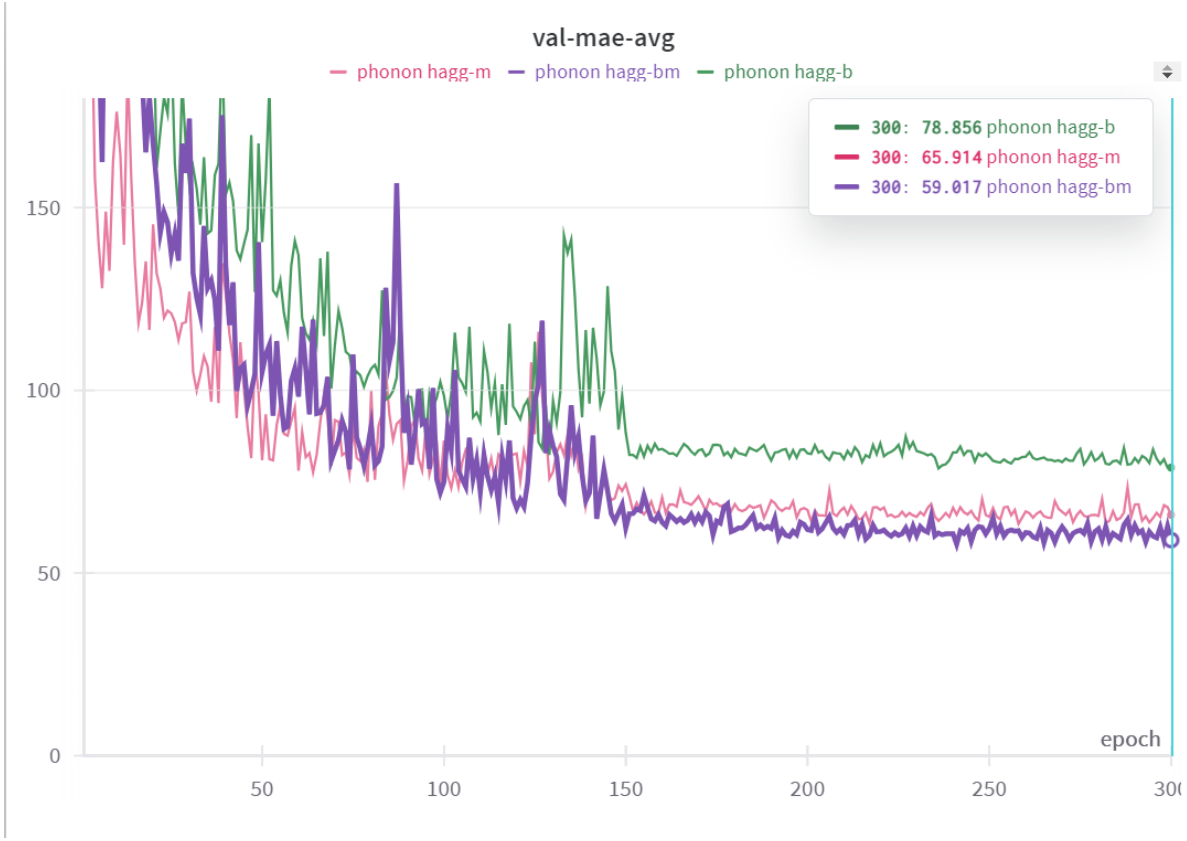
Below, we test on 18,928 calculated formation energies for Perovskites.



Model	Best MAE (eV/Atom)
Bond-only	0.329
Motif-only	0.052
Bond & Motif	0.058

(MatBench) Phonons

Below, we test on a dataset of 1,265 materials with the target being the highest calculated frequency optical phonon mode peak.



Model	Best MAE (cm^{-1})
Bond-only	78.9
Motif-only	65.9
Bond & Motif	59.0

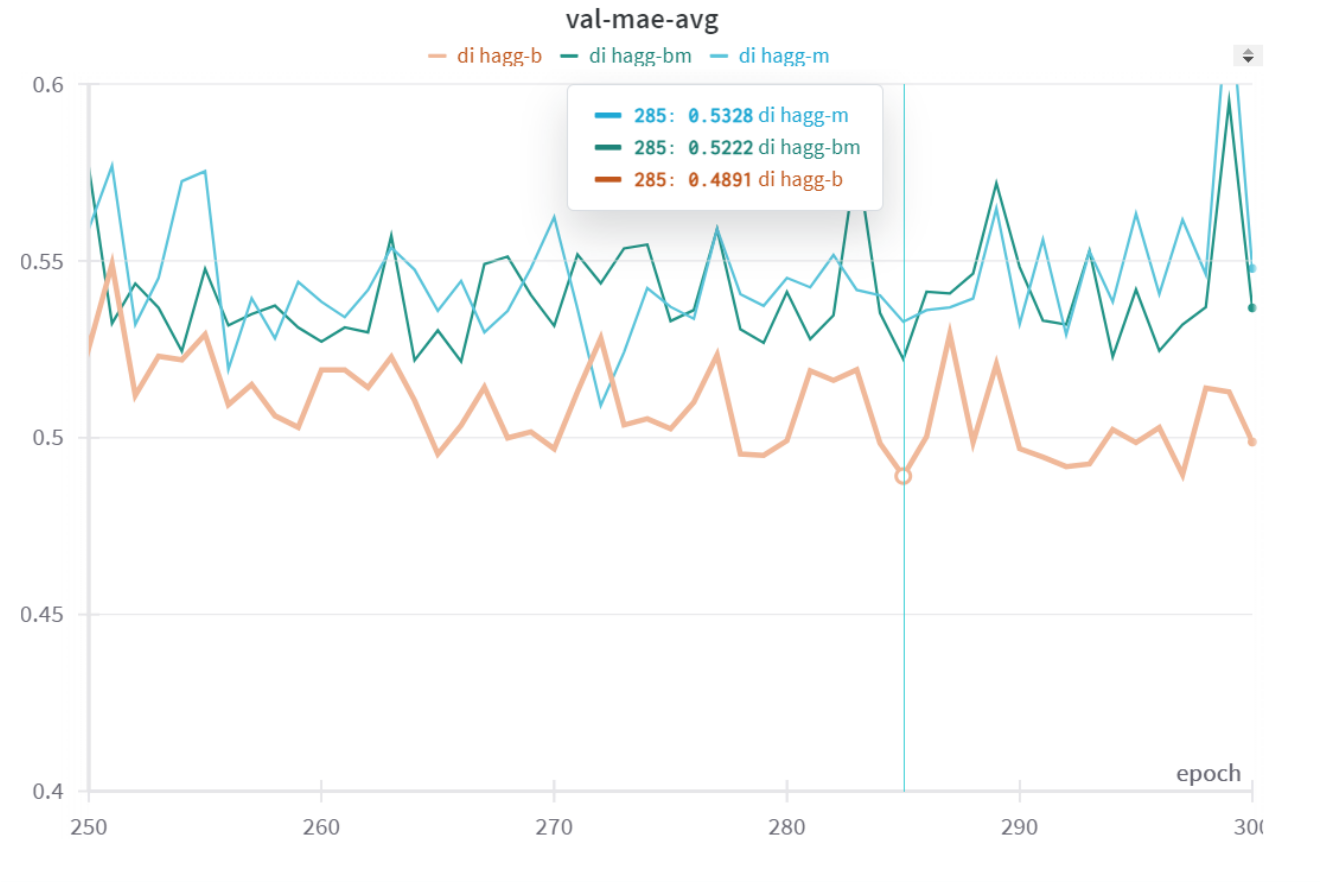
(MatBench) Shear Moduli $\log_{10}(G_{vrh})$

Below, we test on the calculated shear moduli of 10,987 materials

Model	Best MAE (eV)
Bond-only	
Motif-only	
Bond & Motif	

(MatBench) Dielectrics

Below, we test on the calculated refractive index of 4,764 materials



Model	Best MAE
Bond-only	.4891
Motif-only	.5328
Bond & Motif	.5222

Overview of Experimental Results

• Electronic tasks (band gap, dielectric targets) seem to benefit much less, or be negatively impacted, by this additional information. It seems pair-wise correlators are, in general, a better descriptor for such tasks.

• Formation energy tasks benefit greatly from included motif (i.e. higher order geometrical) descriptors

• In the case of Perovskites (a harder set of formation energy targets), this geometrical information seems even more impactful . Though, it also may just help more for the smaller dataset available for training.

• Phonons (also heavily dependent on geometrical information) seem to also benefit greatly from motif-level features

GUESS

Elastic properties (bulk, shear moduli) should also benefit more from motif information than bond information, since these are also related to overall structure.

Conclusion

- Graphs are limited in their expression of higher-order (above pair-wise) features of crystal structures

- Crystal hypergraphs give us a natural way to encode different types of features in one representation