MESSAGE PASSING ALL THE WAY UP

Anonymous authors
Paper under double-blind review

ABSTRACT

The message passing framework is the foundation of the immense success enjoyed by graph neural networks (GNNs) in recent years. In spite of its elegance, there exist many problems it provably cannot solve over given input graphs. This has led to a surge of research on going *"beyond message passing"*, building GNNs which do not suffer from those limitations—a term which has become ubiquitous in regular discourse. However, have those methods truly moved beyond message passing? In this position paper, I argue about the dangers of using this term—especially when teaching graph representation learning to newcomers. I show that any function of interest we want to compute over graphs can, in all likelihood, be expressed using pairwise message passing – just over a potentially *modified* graph, and argue how most practical implementations subtly do this kind of trick anyway. Hoping to initiate a productive discussion, I propose replacing *"beyond message passing"* with a more tame term, **"augmented message passing"**.

1 INTRODUCTION

In the span of only five years, graph neural networks (GNNs) have ascended from a niche of representation learning to one of its most coveted methods—enabling industrial and scientific applications that were not possible before. The growing list of applications includes recommender systems (Ying et al., 2018; Hao et al., 2020), traffic prediction (Derrow-Pinion et al., 2021), chip design (Mirhoseini et al., 2021), virtual drug screening (Stokes et al., 2020) and advances in pure mathematics (Davies et al., 2021), especially representation theory (Blundell et al., 2021).

Most of these successes were propped up by the message passing framework (Gilmer et al., 2017), where pairs of nodes exchange vector-based messages with one another in order to update their representations. However, fundamental limitations of this framework have been identified (Xu et al., 2018; Morris et al., 2019)—and it is unable to detect even the *simplest* of substructures in graphs. For example, message passing neural networks provably cannot distinguish a 6-cycle from two triangles (Murphy et al., 2019; Sato et al., 2021), and they are vulnerable to effects like oversmoothing (Li et al., 2018) and oversquashing (Alon & Yahav, 2020). These limitations have led to a surge in methods that aim to make structural recognition easier for GNNs, which has been undoubtedly one of the most active areas of graph representation learning in the recent few years. See Maron et al. (2018); Murphy et al. (2019); Chen et al. (2019); Vignac et al. (2020); Morris et al. (2019); Chen et al. (2020); Li et al. (2020); de Haan et al. (2020) for just a handful of examples.

Collectively, these methods are known under many names, including *higher-order GNNs* (Morris et al., 2019), and *beyond message passing*. In particular, the latter has become popularised by influential researchers in year-end summaries (Bronstein, 2021; Bronstein & Veličković, 2022) and was the central topic of debate at flagship workshops in the area (Cheng et al., 2021, GTRL'21).

By writing this position paper, I intend to initiate a serious debate on this term, which I consider to be harmful in many cases, *especially* for newcomers entering the area. Namely, the very term "beyond message passing" apparently implies that the message passing primitive needs to be *replaced* in order to break away from the inherent limitations mentioned above. But, how many of the methods that improve the expressive power of GNNs *truly* moved beyond the message passing primitive?

I will make several arguments towards a **negative** answer. Firstly, in all likelihood, *any* function of interest we want to compute over a graph *can* be expressed using pairwise message passing—just

over a potentially *modified* graph structure. Beyond, many powerful GNNs are efficiently implementable *exactly* using message passing. Even when a powerful GNN is easier or semantically clearer to implement with other primitives, interpreting it from the lens of message passing is possible, and can draw meaningful connections to research directions that are currently underappreciated.

To be clear, I am absolutely in favour of research of expressive GNNs. However, teaching all of it as going *"beyond message passing"* is, at best, imprecise and, at worst, harmful—as these methods often can, and often do, make efficient use of the message passing primitive.

Regarding the related *higher-order GNN* term, I find its use to implicitly bind us to the Weisfeiler-Lehman hierarchy (Morris et al., 2021), which is not always a meaningful measure of expressive power—see Loukas (2019; 2020); Corso et al. (2020); Barceló et al. (2020) for examples.

Taken all together, I propose that we coin a new term to describe recent progress in expressive GNNs: one that acknowledges the important role the message passing primitive will continue to play, while also realising we are surpassing its naïve application. I propose **"augmented message passing"**, but am very much open to suggestions from the community.

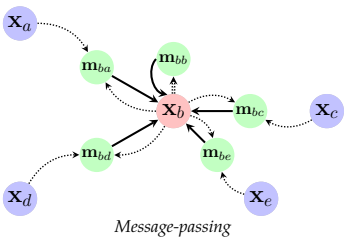
2 MESSAGE PASSING

I will adopt the definition of message passing given by Bronstein et al. (2021). Let a graph be a tuple of *nodes* and *edges*, $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, with one-hop neighbourhoods defined as $\mathcal{N}_u = \{v \in \mathcal{V} \mid (v, u) \in \mathcal{E}\}$. Further, a node feature matrix $\mathbf{X} \in \mathbb{R}^{|\mathcal{V}| \times k}$ gives the features of node u as $\mathbf{x}_u$; I omit edge- and graph-level features for clarity. A *message passing* GNN over this graph is then executed as:

$$\mathbf{h}_u = \phi\left(\mathbf{x}_u, \bigoplus_{v \in \mathcal{N}_u} \psi(\mathbf{x}_u, \mathbf{x}_v)\right) \tag{1}$$

where $\psi : \mathbb{R}^k \times \mathbb{R}^k \to \mathbb{R}^l$ is a *message function*, $\phi : \mathbb{R}^k \times \mathbb{R}^l \to \mathbb{R}^m$ is a *readout function*, and $\bigoplus$ is a permutation-invariant *aggregation function* (such as $\sum$ or $\max$). Both ψ and ϕ can be realised as MLPs, but many special cases exist, giving rise to, e.g., attentional GNNs (Veličković et al., 2017).

Message passing is a primitive that is propped up on *pairwise communication*. This locality makes it scalable, but can also be its weakness; it is known that Equation 1 is incapable of recognising even simple substructures in graphs (Xu et al., 2018), and provably cannot model some processes over collections of more than two nodes (Neuhäuser et al., 2021). Naturally, this spurred a plethora of research on improving upon Equation 1. I will unpack what that research actually did, and realise how it never truly escaped Equation 1, in terms of pairwise communication. The key is in allowing changes to the graph *structure* (e.g. $\mathcal{V}$ or $\mathcal{N}_u$).



Simple example: *Master nodes* To illustrate what is meant by changes to the graph structure, I will first discuss a simple case: that of handling *graph-level features* (e.g. as in Battaglia et al. (2018)). This is useful either when graph-level context is given and should condition the computation, when shortening the graph's diameter through a bottleneck is desirable, or simply when tasks require graph-level decision-making.

The standard way in which such a feature is implemented (Gilmer et al., 2017) is by incorporating a *master node*, μ, connecting it with all other nodes, and then performing message passing as usual. Mathematically, $\mathcal{V}' = \mathcal{V} \cup \{\mu\}$, $\mathcal{N}'_u = \mathcal{N}_u \cup \{\mu\}$, and $\mathcal{N}'_\mu = \mathcal{V}$.

It is now possible to do global-conditioned computation, and any two nodes are no more than two hops apart. Did we need to go *beyond* message passing to do that? No—we **augmented** message passing to perform over a graph that has one extra node, and $O(\mathcal{V})$ new edges. All the while, the message passing primitive itself is unchanged[1]. Such conclusions will repeat as we go on.

[1]It is worth noting that often, we need to tag the added nodes or edges (μ in this case) with a special one-hot *flag* feature, which would give the message function ψ sufficient context to treat them separately.

3 AUGMENTED MESSAGE PASSING

Broadly, augmented message passing techniques can be categorised into *six* distinct types.

Feature augmentation Since GNNs are unable to solve certain simple tasks on graphs, a very broad class of methods seek to (partially) precompute answers to those tasks, and feed the results as additional inputs. For example, it is well-known that even incorporating simple *identifying* information such as a one-hot encoding (Murphy et al., 2019) or a random feature (Sato et al., 2021) can assist GNNs in detecting patterns such as cycles. These can be further generalised to methods that literally *count* subgraphs of interest (Bouritsas et al., 2020), or provide the graph's Laplacian eigenvectors (Dwivedi & Bresson, 2020) as input. In all of the above, once the features are computed, the GNN computations proceed as before—hence, trivially expressible using message passing.

Message passing modulation While adding additional features can and does boost expressive power, it is arguably *non-binding*: the model is not forced to use these features. Accordingly, the message function can be modulated to explicitly take into account these computations. One canonical example are directional graph networks (Beaini et al., 2021), where Laplacian eigenvectors are used to define "flows" on a graph, which explicitly guide how various incoming messages are scaled. More recent proposals like LSPE (Dwivedi et al., 2021) explicitly include a message passing mechanism over the computed positional features. In all cases, while the message function, ψ, is modified, the blueprint of Equation 1 remains, and this case is also trivially an instance of message passing.

Graph rewiring The previous categories did not require moving away from message passing over exactly the given graph—at most, constraints needed to be placed on ψ. However, as the input graph is often noisy, missing, or suboptimal for the task, many methods *modify* the edges of the input graph to compensate. Such *graph rewiring* methods leave $\mathcal{V}$ unchanged, but make direct changes to $\mathcal{N}_u$.

One common method in this space is to operate over a *fully connected graph*, i.e. setting $\mathcal{N}_u = \mathcal{V}$. This allows the model to rediscover the edges it needs, and has become quite popular in the context of *graph Transformers* (Ying et al., 2021; Kreuzer et al., 2021; Mialon et al., 2021), which allows for building GNNs that are able to "win the hardware lottery" (Hooker, 2021). Conveniently, the fully connected view also encompasses spectrally defined graph convolutions, such as the graph Fourier transform (Bruna et al., 2013). Nontrivial changes to $\mathcal{N}_u$, such as multi-hop layers (Defferrard et al., 2016), rewiring based on diffusion (Klicpera et al., 2019) or curvature (Topping et al., 2021), and subsampling (Hamilton et al., 2017) are also supported. Lastly, the methods which dynamically alter the adjacency in a learnable fashion (Kipf et al., 2018; Wang et al., 2019; Kazi et al., 2020; Veličković et al., 2020) can also be classified under this umbrella. The underlying message function setup of Equation 1 is unchanged, hence these methods are still expressible using message passing.

Subgraph aggregation An extension of graph rewiring methods which has seen considerable interest lately concerns itself with learning graph representations by aggregating multiple *subgraphs* at once—while being mindful of repeated nodes across those subgraphs; see Papp et al. (2021); Cotta et al. (2021); Zhao et al. (2021); Bevilacqua et al. (2021). Such computations can be realised in the message passing framework by *replicating* the nodes for every subgraph of interest, and *connecting the copies* of every node together. Mathematically, if we are learning over K subgraphs, let $\mathcal{V}' = \mathcal{V} \times \{1, 2, \ldots, K\}$ and, assuming that subgraph i's edges define neighbourhoods $\mathcal{N}_u^{(i)}$, we can redefine neigbourhoods as follows: $\mathcal{N}'_{u,i} = \{(v, i) \mid v \in \mathcal{N}_u^{(i)}\} \cup \{(u, j) \mid j \in \{1, 2, \ldots, k\}\}$.

Substructure based methods As briefly mentioned, naturally-occurring phenomena are sometimes best described by interactions of *groups* of entities—e.g., the *functional groups* in a molecule can often strongly influence its properties (Duvenaud et al., 2015). Accordingly, methods have been developed to support computing representations over junction trees (Fey et al., 2020), spectral signals (Stachenfeld et al., 2020), simplicial complexes (Bodnar et al., 2021b), cellular complexes (Bodnar et al., 2021a), or general k-tuples of nodes (Morris et al., 2019; 2020) and hypergraphs (Huang & Yang, 2021; Chien et al., 2021; Georgiev et al., 2022). Whenever the groups of interest are greater than two nodes, the expressivity of pairwise message passing can be brought into question. Indeed, over the *original* graph, it can be provably impossible to use pairwise messaging to simulate such interactions (Neuhäuser et al., 2021). But what if we *modify* the graph structure?

Indeed, we can; and very often, such methods will explicitly call out the use of message passing to achieve their objective (Fey et al., 2020; Stachenfeld et al., 2020). The trick is to create new nodes for every substructure that we want to model, and appropriately connecting them to their constituent nodes. How easy this is to do depends on whether the functions of interest are *permutation invariant*.

If yes, all we have to do is establish bidirectional edges between the constituent nodes and the corresponding "substructure node". Mathematically, we assume that we have K substructures, $\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_K$ each operating in a permutation invariant way over its constituent nodes ($\mathcal{S}_i \subseteq \mathcal{V}$). Then, we augment the graph by creating new substructure nodes: $\mathcal{V}' = \mathcal{V} \cup \{\mu_1, \mu_2, \ldots, \mu_K\}$, and modifying the neighbourhoods to connect every constituent to its substructure(s): $\mathcal{N}'_u = \mathcal{N}_u \cup \{\mu_i \mid u \in \mathcal{S}_i\}$, and $\mathcal{N}'_{\mu_i} = \mathcal{S}_i$. Note that this is a more general case of the master node approach of Section 2, which can be seen as having just one substructure, $\mathcal{S}_1 = \mathcal{V}$.

To prove that *any* permutation invariant function over substructures can be represented in this way, it is sufficient to realise that, when computing substructure representations $\mathbf{h}_{\mu_i}$, Equation 1 implements a *Deep Sets* model over the nodes in $\mathcal{S}_i$ (Zaheer et al., 2017). Under certain conditions on the message function hyperparameters, it is known that *any* permutation invariant function over $\mathcal{S}_i$ *must* be expressible in this form (Wagstaff et al., 2019)—hence, this construction is universal.

When interactions within a substructure are permutation sensitive, a more intricate gadget is required which, due to space constraints, I fully derive in Appendix A. In a nutshell, we can either create $O(\mathcal{S}_i)$ new nodes to process nodes' features *one at a time* acccording to the permutation (not unlike a long short-term memory (Hochreiter & Schmidhuber, 1997)) or using carefully constructed message functions to materialise a *concatenation* of the inputs which respects the permutation.

In both cases, we have demonstrated an equivalent architecture using only the pairwise message passing primitive, which is often the way such methods are implemented anyway.

General equivariant GNNs While previous sections all sought to identify a *specific* computation or motif that can improve GNN expressivity, a converse approach is to characterise *all* possible linear permutation-equivariant layers over an input graph, and use them as a basis for building equivariant GNNs (Maron et al., 2018). A similar analysis over image data reveals that there is exactly one type of linear translation equivariant layer over images: the convolution (Bronstein et al., 2021).

Using this framework, Maron et al. (2018) discover a basis of two linear invariant and 15 linear equivariant layers for the setting with edge-valued inputs (for data defined over k-tuples of nodes, the dimension is defined by the k-th and $2k$-th Bell numbers). These 15 layers effectively allow for recombining information across *pairs of edges* while respecting the graph structure's symmetries. Therefore, they are represented as matrices in $\mathbb{R}^{\mathcal{V}^2 \times \mathcal{V}^2}$, which get multiplied with edge features.

Despite the clearly tensorial semantics of such an approach, it can still be represented in the language of pairwise message passing: any square matrix multiplication operation represents a convolutional GNN instance over the graph implied by that matrix's nonzero entries (Bronstein et al., 2021). The key difference is that in this case the messages are passed over *edges* rather than nodes, and therefore we need to invent new nodes corresponding to edges, and connect them accordingly!

Mathematically, assume we want to multiply with a basis matrix $\mathbf{B} \in \mathbb{R}^{\mathcal{V}^2 \times \mathcal{V}^2}$. Inventing new edge-based nodes corresponds to $\mathcal{V}' = \mathcal{V} \cup \{e_{uv} \mid u, v \in \mathcal{V}\}$. Then, we need to connect these edges to the nodes incident to them, and also to other edges, whenever the entries of $\mathbf{B}$ mandate it. Hence the neighbourhoods update as follows: $\mathcal{N}'_u = \mathcal{N}_u \cup \{e_{ab} \mid a = u \vee b = u\}$ for nodes, and $\mathcal{N}'_{e_{uv}} = \{u, v\} \cup \{e_{ab} \mid \mathbf{B}_{(a,b),(u,v)} \neq 0\}$ for edges. Similar gadgets (potentially "tensored up" to k-tuple nodes) would also apply for follow-up work, including but not limited to Maron et al. (2019); Keriven & Peyré (2019); Albooyeh et al. (2019); Azizian & Lelarge (2020).

In this particular case, the semantics of message passing do not "line up" very nicely with the tensorial approach. Still, expressing the edges as nodes reveals connections to several proposals that *do* directly augment message passing, especially over the related concept of a *line graph* (Monti et al., 2018; Cai et al., 2021). Hence, even when the conversion to message passing is not the most practical, it can still be a useful exercise that reveals surprising connections between seemingly unrelated proposals, and stimulates future research.

REFERENCES

Marjan Albooyeh, Daniele Bertolini, and Siamak Ravanbakhsh. Incidence networks for geometric deep learning. *arXiv preprint arXiv:1905.11460*, 2019.

Uri Alon and Eran Yahav. On the bottleneck of graph neural networks and its practical implications. *arXiv preprint arXiv:2006.05205*, 2020.

Waïss Azizian and Marc Lelarge. Characterizing the expressive power of invariant and equivariant graph neural networks. 2020.

Pablo Barceló, Egor Kostylev, Mikael Monet, Jorge Pérez, Juan Reutter, and Juan-Pablo Silva. The logical expressiveness of graph neural networks. In *8th International Conference on Learning Representations (ICLR 2020)*, 2020.

Peter W Battaglia, Jessica B Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, et al. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*, 2018.

Dominique Beaini, Saro Passaro, Vincent Létourneau, Will Hamilton, Gabriele Corso, and Pietro Liò. Directional graph networks. In *International Conference on Machine Learning*, pp. 748–758. PMLR, 2021.

Beatrice Bevilacqua, Fabrizio Frasca, Derek Lim, Balasubramaniam Srinivasan, Chen Cai, Gopinath Balamurugan, Michael M Bronstein, and Haggai Maron. Equivariant subgraph aggregation networks. *arXiv preprint arXiv:2110.02910*, 2021.

Charles Blundell, Lars Buesing, Alex Davies, Petar Veličković, and Geordie Williamson. Towards combinatorial invariance for kazhdan-lusztig polynomials. *arXiv preprint arXiv:2111.15161*, 2021.

Cristian Bodnar, Fabrizio Frasca, Nina Otter, Yu Guang Wang, Pietro Liò, Guido F Montufar, and Michael Bronstein. Weisfeiler and lehman go cellular: Cw networks. *Advances in Neural Information Processing Systems*, 34, 2021a.

Cristian Bodnar, Fabrizio Frasca, Yuguang Wang, Nina Otter, Guido F Montufar, Pietro Lio, and Michael Bronstein. Weisfeiler and lehman go topological: Message passing simplicial networks. In *International Conference on Machine Learning*, pp. 1026–1037. PMLR, 2021b.

Giorgos Bouritsas, Fabrizio Frasca, Stefanos Zafeiriou, and Michael M Bronstein. Improving graph neural network expressivity via subgraph isomorphism counting. *arXiv preprint arXiv:2006.09252*, 2020.

Michael Bronstein. Predictions and hopes for graph ml in 2021, Jan 2021. URL `https://towardsdatascience.com/predictions-and-hopes-for-graph-ml-in-2021-6af2121c3e3d`.

Michael Bronstein and Petar Veličković. Predictions and hopes for geometric & graph ml in 2022, Jan 2022. URL `https://towardsdatascience.com/predictions-and-hopes-for-geometric-graph-ml-in-2022-aa3b8b79f5cc`.

Michael M Bronstein, Joan Bruna, Taco Cohen, and Petar Veličković. Geometric deep learning: Grids, groups, graphs, geodesics, and gauges. *arXiv preprint arXiv:2104.13478*, 2021.

Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. Spectral networks and locally connected networks on graphs. *arXiv preprint arXiv:1312.6203*, 2013.

Lei Cai, Jundong Li, Jie Wang, and Shuiwang Ji. Line graph neural networks for link prediction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

Zhengdao Chen, Soledad Villar, Lei Chen, and Joan Bruna. On the equivalence between graph isomorphism testing and function approximation with gnns. *Advances in neural information processing systems*, 32, 2019.

Zhengdao Chen, Lei Chen, Soledad Villar, and Joan Bruna. Can graph neural networks count substructures? *Advances in neural information processing systems*, 33:10383–10395, 2020.

Xiuyuan Cheng, Smita Krishnaswamy, Jure Leskovec, Bastian Rieck, Soledad Villar, and Guy Wolf. Iclr 2021 workshop on geometrical and topological representation learning, May 2021. URL `https://gt-rl.github.io/2021/`.

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

Gabriele Corso, Luca Cavalleri, Dominique Beaini, Pietro Liò, and Petar Veličković. Principal neighbourhood aggregation for graph nets. *Advances in Neural Information Processing Systems*, 33:13260–13271, 2020.

Leonardo Cotta, Christopher Morris, and Bruno Ribeiro. Reconstruction for powerful graph representations. *Advances in Neural Information Processing Systems*, 34, 2021.

Alex Davies, Petar Veličković, Lars Buesing, Sam Blackwell, Daniel Zheng, Nenad Tomašev, Richard Tanburn, Peter Battaglia, Charles Blundell, András Juhász, et al. Advancing mathematics by guiding human intuition with ai. *Nature*, 600(7887):70–74, 2021.

Pim de Haan, Taco S Cohen, and Max Welling. Natural graph networks. *Advances in Neural Information Processing Systems*, 33:3636–3646, 2020.

Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. *Advances in neural information processing systems*, 29, 2016.

Austin Derrow-Pinion, Jennifer She, David Wong, Oliver Lange, Todd Hester, Luis Perez, Marc Nunkesser, Seongjae Lee, Xueying Guo, Brett Wiltshire, et al. Eta prediction with graph neural networks in google maps. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, pp. 3767–3776, 2021.

David K Duvenaud, Dougal Maclaurin, Jorge Iparraguirre, Rafael Bombarell, Timothy Hirzel, Alán Aspuru-Guzik, and Ryan P Adams. Convolutional networks on graphs for learning molecular fingerprints. *Advances in neural information processing systems*, 28, 2015.

Vijay Prakash Dwivedi and Xavier Bresson. A generalization of transformer networks to graphs. *arXiv preprint arXiv:2012.09699*, 2020.

Vijay Prakash Dwivedi, Anh Tuan Luu, Thomas Laurent, Yoshua Bengio, and Xavier Bresson. Graph neural networks with learnable structural and positional representations. *arXiv preprint arXiv:2110.07875*, 2021.

Matthias Fey, Jan-Gin Yuen, and Frank Weichert. Hierarchical inter-message passing for learning on molecular graphs. *arXiv preprint arXiv:2006.12179*, 2020.

Dobrik Georgiev, Marc Brockschmidt, and Miltiadis Allamanis. Heat: Hyperedge attention networks. *arXiv preprint arXiv:2201.12113*, 2022.

Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *International conference on machine learning*, pp. 1263–1272. PMLR, 2017.

Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30, 2017.

Junheng Hao, Tong Zhao, Jin Li, Xin Luna Dong, Christos Faloutsos, Yizhou Sun, and Wei Wang. P-companion: A principled framework for diversified complementary product recommendation. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, pp. 2517–2524, 2020.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8): 1735–1780, 1997.

Sara Hooker. The hardware lottery. *Communications of the ACM*, 64(12):58–65, 2021.

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

Anees Kazi, Luca Cosmo, Nassir Navab, and Michael Bronstein. Differentiable graph module (dgm) for graph convolutional networks. *arXiv preprint arXiv:2002.04999*, 2020.

Nicolas Keriven and Gabriel Peyré. Universal invariant and equivariant graph neural networks. *Advances in Neural Information Processing Systems*, 32, 2019.

Thomas Kipf, Ethan Fetaya, Kuan-Chieh Wang, Max Welling, and Richard Zemel. Neural relational inference for interacting systems. In *International Conference on Machine Learning*, pp. 2688–2697. PMLR, 2018.

Johannes Klicpera, Stefan Weißenberger, and Stephan Günnemann. Diffusion improves graph learning. *Advances in Neural Information Processing Systems*, 32, 2019.

Devin Kreuzer, Dominique Beaini, Will Hamilton, Vincent Létourneau, and Prudencio Tossou. Rethinking graph transformers with spectral attention. *Advances in Neural Information Processing Systems*, 34, 2021.

Pan Li, Yanbang Wang, Hongwei Wang, and Jure Leskovec. Distance encoding: Design provably more powerful neural networks for graph representation learning. *Advances in Neural Information Processing Systems*, 33:4465–4478, 2020.

Qimai Li, Zhichao Han, and Xiao-Ming Wu. Deeper insights into graph convolutional networks for semi-supervised learning. In *Thirty-Second AAAI conference on artificial intelligence*, 2018.

Andreas Loukas. What graph neural networks cannot learn: depth vs width. *arXiv preprint arXiv:1907.03199*, 2019.

Andreas Loukas. How hard is to distinguish graphs with graph neural networks? *Advances in neural information processing systems*, 33:3465–3476, 2020.

Haggai Maron, Heli Ben-Hamu, Nadav Shamir, and Yaron Lipman. Invariant and equivariant graph networks. *arXiv preprint arXiv:1812.09902*, 2018.

Haggai Maron, Heli Ben-Hamu, Hadar Serviansky, and Yaron Lipman. Provably powerful graph networks. *Advances in neural information processing systems*, 32, 2019.

Grégoire Mialon, Dexiong Chen, Margot Selosse, and Julien Mairal. Graphit: Encoding graph structure in transformers. *arXiv preprint arXiv:2106.05667*, 2021.

Azalia Mirhoseini, Anna Goldie, Mustafa Yazgan, Joe Wenjie Jiang, Ebrahim Songhori, Shen Wang, Young-Joon Lee, Eric Johnson, Omkar Pathak, Azade Nazi, et al. A graph placement methodology for fast chip design. *Nature*, 594(7862):207–212, 2021.

Federico Monti, Oleksandr Shchur, Aleksandar Bojchevski, Or Litany, Stephan Günnemann, and Michael M Bronstein. Dual-primal graph convolutional networks. *arXiv preprint arXiv:1806.00770*, 2018.

Christopher Morris, Martin Ritzert, Matthias Fey, William L Hamilton, Jan Eric Lenssen, Gaurav Rattan, and Martin Grohe. Weisfeiler and leman go neural: Higher-order graph neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pp. 4602–4609, 2019.

Christopher Morris, Gaurav Rattan, and Petra Mutzel. Weisfeiler and leman go sparse: Towards scalable higher-order graph embeddings. *Advances in Neural Information Processing Systems*, 33:21824–21840, 2020.

Christopher Morris, Yaron Lipman, Haggai Maron, Bastian Rieck, Nils M Kriege, Martin Grohe, Matthias Fey, and Karsten Borgwardt. Weisfeiler and leman go machine learning: The story so far. *arXiv preprint arXiv:2112.09992*, 2021.

Ryan Murphy, Balasubramaniam Srinivasan, Vinayak Rao, and Bruno Ribeiro. Relational pooling for graph representations. In *International Conference on Machine Learning*, pp. 4663–4673. PMLR, 2019.

Leonie Neuhäuser, Michael T Schaub, Andrew Mellor, and Renaud Lambiotte. Opinion dynamics with multi-body interactions. In *International Conference on Network Games, Control and Optimization*, pp. 261–271. Springer, 2021.

Pál András Papp, Karolis Martinkus, Lukas Faber, and Roger Wattenhofer. Dropgnn: random dropouts increase the expressiveness of graph neural networks. *Advances in Neural Information Processing Systems*, 34, 2021.

Ryoma Sato, Makoto Yamada, and Hisashi Kashima. Random features strengthen graph neural networks. In *Proceedings of the 2021 SIAM International Conference on Data Mining (SDM)*, pp. 333–341. SIAM, 2021.

Kimberly Stachenfeld, Jonathan Godwin, and Peter Battaglia. Graph networks with spectral message passing. *arXiv preprint arXiv:2101.00079*, 2020.

Jonathan M Stokes, Kevin Yang, Kyle Swanson, Wengong Jin, Andres Cubillos-Ruiz, Nina M Donghia, Craig R MacNair, Shawn French, Lindsey A Carfrae, Zohar Bloom-Ackermann, et al. A deep learning approach to antibiotic discovery. *Cell*, 180(4):688–702, 2020.

Jake Topping, Francesco Di Giovanni, Benjamin Paul Chamberlain, Xiaowen Dong, and Michael M Bronstein. Understanding over-squashing and bottlenecks on graphs via curvature. *arXiv preprint arXiv:2111.14522*, 2021.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

Petar Veličković, Lars Buesing, Matthew Overlan, Razvan Pascanu, Oriol Vinyals, and Charles Blundell. Pointer graph networks. *Advances in Neural Information Processing Systems*, 33: 2232–2244, 2020.

Clement Vignac, Andreas Loukas, and Pascal Frossard. Building powerful and equivariant graph neural networks with structural message-passing. *Advances in Neural Information Processing Systems*, 33:14143–14155, 2020.

Edward Wagstaff, Fabian Fuchs, Martin Engelcke, Ingmar Posner, and Michael A Osborne. On the limitations of representing functions on sets. In *International Conference on Machine Learning*, pp. 6487–6494. PMLR, 2019.

Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein, and Justin M Solomon. Dynamic graph cnn for learning on point clouds. *Acm Transactions On Graphics (tog)*, 38(5): 1–12, 2019.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*, 2018.

Chengxuan Ying, Tianle Cai, Shengjie Luo, Shuxin Zheng, Guolin Ke, Di He, Yanming Shen, and Tie-Yan Liu. Do transformers really perform badly for graph representation? *Advances in Neural Information Processing Systems*, 34, 2021.

Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L Hamilton, and Jure Leskovec. Graph convolutional neural networks for web-scale recommender systems. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, pp. 974–983, 2018.

Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Russ R Salakhutdinov, and Alexander J Smola. Deep sets. *Advances in neural information processing systems*, 30, 2017.

Lingxiao Zhao, Wei Jin, Leman Akoglu, and Neil Shah. From stars to subgraphs: Uplifting any gnn with local structure awareness. *arXiv preprint arXiv:2110.03753*, 2021.

A GRAPH MODULATION FOR PERMUTATION-SENSITIVE SUBSTRUCTURES

Let's assume that we want to model a function $f(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_k)$ over k-tuples of nodes, and that it is permutation sensitive in the first n arguments, and permutation invariant in the remaining m (s.t. $n + m = k$). Just like before, we can fit the m invariant parameters in the same way as in the main paper (inventing a new "master node", μ, that exchanges messages with those m nodes). This gives us a modified function, re-using Equation 1:

$$f(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_k) = g\left(\mathbf{x}_1, \ldots, \mathbf{x}_n, \phi_1\left(\mathbf{x}_\mu, \bigoplus_{l=1}^{m} \psi_1(\mathbf{x}_\mu, \mathbf{x}_{n+l})\right)\right) \tag{2}$$

where we can set the master node's features, $\mathbf{x}_\mu$ as a zero-vector initially, if none are available.

Now, we need to process all of these $n+1$ inputs to g in a way that is order-dependent. The simplest way to do so is to create a new node which would use "concat-aggregation", but it is unclear whether this is explicitly supported by Equation 1. There do exist ways to materialise this concatenation by preparing message functions that copy each of the $\mathbf{x}_i$ vectors into a separate "slot" of the result vector, then applying the sum aggregation. Then the concatenated node can send its own features as a message to a node storing the result.

If we want to avoid explicitly materialising concatenations, it is also possible to use Turing-complete recurrent models such as the LSTM (Hochreiter & Schmidhuber, 1997) to gradually process the inputs to g one at a time, storing intermediate results $\boldsymbol{\lambda}_i$ as we go along ($i \in \{1, 2, \ldots, n+1\}$).

Initialising $\boldsymbol{\lambda}_i = \mathbf{0}$ (or making it learnable), we then update each of them as follows:

$$\boldsymbol{\lambda}'_i = \begin{cases} \phi_2\left(\boldsymbol{\lambda}_i, \psi_2(\boldsymbol{\lambda}_i, \boldsymbol{\lambda}_{i-1}) + \psi_2(\boldsymbol{\lambda}_i, \mathbf{x}_i)\right) & i \leq n \\ \phi_2\left(\boldsymbol{\lambda}_i, \psi_2(\boldsymbol{\lambda}_i, \boldsymbol{\lambda}_{i-1}) + \psi_2(\boldsymbol{\lambda}_i, \mathbf{x}_\mu)\right) & i = n+1 \end{cases} \tag{3}$$

which aligns with the message passing framework of Equation 1 (setting $\mathcal{N}_{\lambda_i} = \{\lambda_{i-1}, i\}$, and specially $\mathcal{N}_{\lambda_{n+1}} = \{\lambda_n, \mu\}$). Further, these equations align with recurrent neural networks (e.g. we can recover LSTMs for a special choice of ϕ_2 and ψ_2). Once $n+1$ steps of such a model have been performed, sufficient time has passed for all the features to propagate, and we may use $\boldsymbol{\lambda}_{n+1}$ as the final representation of our target function f, potentially feeding its outputs back to relevant nodes.