Can strong structural encoding reduce the importance of Message Passing?

Floor Eijkelboom [1]   Erik Bekkers [1]   Michael Bronstein [2]   Francesco Di Giovanni [3]

Abstract

The most prevalent class of neural networks operating on graphs are message passing neural networks (MPNNs), in which the representation of a node is updated iteratively by aggregating information in the 1-hop neighborhood. Since this paradigm for computing node embeddings may prevent the model from learning coarse topological structures, the initial features are often augmented with structural information of the graph, typically in the form of Laplacian eigenvectors or Random Walk transition probabilities. In this work, we explore the contribution of message passing when strong structural encodings are provided. We introduce a novel way of modeling the interaction between feature and structural information based on their tensor product rather than the standard concatenation. The choice of interaction is compared in common scenarios and in settings where the capacity of the message-passing layer is severely reduced and ultimately the message-passing phase is removed altogether. Our results indicate that using tensor-based encodings is always at least on par with the concatenation-based encoding and that it makes the model much more robust when the message passing layers are removed, on some tasks incurring almost no drop in performance. This suggests that the importance of message passing is limited when the model can construct strong structural encodings.

1. Introduction

Graph Neural Networks (GNNs) (Gori et al., 2005; Scarselli et al., 2008) is now one of the most studied and adopted framework for machine learning on graphs. GNNs typically operate on attributed graphs, and learn functions that model complex interactions among the underlying topology and the features (Sperduti, 1993; Goller & Kuchler, 1996; Bruna et al., 2014; Defferrard et al., 2016). Since graphs are a flexible way of representing many data, GNNs are being developed in many scientific and industrial fields alike (Stokes et al., 2020; Mirhoseini et al., 2021; DeZoort et al., 2023; Bapst et al., 2020). Accordingly, understanding what makes a GNN work, when it is needed, and how to best leverage the topological information from the graph in the GNN pipeline are all essential problems of broad interest.

One of the most common classes of GNNs is message passing neural networks (MPNNs) (Gilmer et al., 2017), where the features associated with each node are iteratively updated based on the 1-hop neighborhood information in the graph. While this approach directly leverages the sparsity of the underlying graph by only exchanging messages among adjacent nodes, a consequence of this learning procedure is that nodes with similar local structures will obtain similar hidden representations, even when they actually play different roles from a coarser topological scale. These ambiguities, ultimately, impact the expressive power of MPNNs. In fact, Xu et al. (2019); Morris et al. (2019) showed that MPNNs are, at most, as powerful as the 1-WL color refinement algorithm (Weisfeiler & Leman, 1968) in distinguishing pairs of graphs without features. In tasks where the graph structure is fundamental - e.g. in molecular tasks - this limits the capabilities of the network severely.

A solution to the problem of limited expressive power of MPNNs that minimizes the impact on the efficiency by preserving the message-passing template, amounts to augmenting the initial node features with structural information (Bouritsas et al., 2022; Dwivedi et al., 2021), and is at the heart of applications of transformers to graphs (Ying et al., 2021; Kreuzer et al., 2021). This graph-aware feature augmentation is typically based on either the eigenvectors of the graph Laplacian or on the transition probabilities of random walks on the graph. Although the choice of the structural information to be provided is being investigated extensively (Rampasek et al., 2022), *how* we combine this information with the input features is a much less studied subject. In fact, the canonical choice in this regard is simply concatenating the node-features and the structural (positional) ones before feeding them into a message-passing layer (often after a

[1]University of Amsterdam [2]University of Oxford [3]University of Cambridge. Correspondence to: Floor Eijkelboom <eijkelboom-floor@gmail.com>.

suitable encoding). Moreover, while structural information is now known to improve MPNNs, especially on molecular tasks, it is unclear to what extent message-passing is needed, especially if we can extract structural information from the graph in a powerful way.

Contributions. In this paper, we explore structural encoding in GNNs and the role of the message-passing paradigm when using structural features. First, we introduce a novel way of modeling the interaction between feature and structural information based on their *tensor product* – rather than the conventional concatenation operation – and compare how these two approaches to structural encoding affect performance in different GNN models. Second, in order to assess the impact of the structural encoding compared to that of the actual message-passing paradigm, we study the effects of reducing the weights in the message-passing layer and number of message passing layers in the model, ultimately entirely removing the message-passing phase.

Our results indicate that using a tensor-product encoding is always on par with the concatenation-based encoding, often slightly outperforming the latter. Moreover, we provide empirical evidence that when excluding the message passing layer from our network, the tensor-based encoding is significantly more robust than its concatenation counterpart. In fact, on some tasks, when using a tensor encoding almost no drop in performance is observed, and a GNN consisting of encoder and decoder, *without any message passing*, performs on par with GNNs that include message passing even when the number of parameters used is significantly reduced. Furthermore, we observe that when the model is provided with strong structural encodings, there is no major improvement noticeable when using dense message passing layers compared to using very sparsely parametrized ones for both concatenation and tensor encodings. This suggests that *the role of message passing layers may often be limited when the model is able to encode structural information in a sufficiently powerful way.*

2. Preliminaries

Graphs and MPNNs. Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be an undirected graph with nodes $\mathcal{V}$ and edges $\mathcal{E}$ and let $i \sim j$ denote that nodes i and j are connected, i.e., $(i, j) \in \mathcal{E}$. The adjacency matrix $\mathbf{A}$ encodes the connectivity of the graph since $A_{ij} = 1$ if $i \sim j$ and $A_{ij} = 0$ otherwise, while the degree matrix $\mathbf{D} = \mathrm{diag}(d_1, \ldots, d_n)$, where $d_i = \sum_j A_{ij}$, represents the connectivity of each node.

In the deep learning context, the nodes of the graph are typically endowed with features $\{\mathbf{h}_i : i \in \mathcal{V}\} \subset \mathbb{R}^d$. Graphs lend themselves as useful descriptors for data that live on irregular domains, e.g. molecules or social networks. Graph neural networks (GNNs) are neural architectures that take as input graphs and process the graph structure in a permutation-equivariant way, e.g., to predict the chemical properties of a molecular graph.

One of the most common classes of GNNs are Message Passing neural networks (MPNNs) (Gilmer et al., 2017), where the features associated with each node are iteratively updated based on the 1-hop neighborhood information, i.e.

$$\mathbf{h}_i^{\ell+1} = \mathsf{MPNNLayer}(\mathbf{h}_i^{\ell}, \{\!\!\{\mathbf{h}_j^{\ell}\}\!\!\}_{i \sim j}), \qquad (1)$$

where $\mathbf{h}_i^{\ell}$ denotes the hidden state of node i in layer ℓ and $\{\!\!\{\mathbf{h}_j^{\ell}\}\!\!\}_{i \sim j}$ is the multiset of adjacent features. Note that the MPNN formalism includes several main instances of GNN architectures such as GCN (Kipf & Welling, 2017), GIN (Xu et al., 2019), or SAGEConv (Hamilton et al., 2017). The initial features $\{\mathbf{h}_i^{\mathrm{in}}\}$ are typically embedded using some multi-layer Perceptron (MLP),

$$\mathbf{h}_i^{\ell=0} = \mathsf{Encode}(\mathbf{h}_i^{\mathrm{in}}).$$

To get a hidden state representing the entire graph – which is needed for graph-level tasks – a permutation-invariant readout map is applied to all final hidden states of the nodes, that is for an MPNN of L layers we have

$$\mathbf{h}_{\mathcal{G}} = \mathsf{Agg}_{v \in \mathcal{V}} \mathbf{h}_v^L,$$

where Agg is a permutation invariant operator such as sum, mean, or max. This state is then passed through a final decoder (often an MLP) to derive the final prediction computed by the MPNN.

Augmented MPNNs. A now established paradigm to frame the expressive power of GNNs is the graph-isomorphism test. In fact, Xu et al. (2019); Morris et al. (2019) proved that MPNNs are, at most, as powerful as the 1-WL test (Weisfeiler & Leman, 1968) in distinguishing *unattributed* graphs. As a consequence of this analogy, MPNNs typically struggle to recognize (or count) substructures (Chen et al., 2020) and will, for instance, assign equal graph-level embeddings to the graphs in Figure 1 despite them being non-isomorphic due to the existence of cycles of different length.
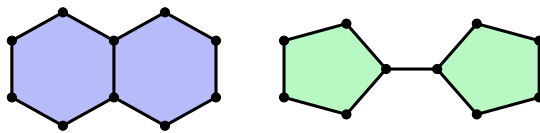

Figure 1. Non-isomorphic graphs not distinguished by 1-WL.

In tasks such as molecular prediction, identifying substructures and hence breaking ambiguities intrinsic to the 1-WL

color refinement, play an important role. To compensate for the shortcomings of classical MPNNs, several approaches have been proposed to design more powerful GNNs (Maron et al., 2018; Keriven & Peyré, 2019; Geerts & Reutter, 2022). However, most of these frameworks suffer from often impractical space-time complexity.

A different approach, amounts to augmenting the standard MPNN formulation in various ways by incorporating **topological information** from the underlying graph – in accordance with the conventional nomenclature, we use the terminology 'structural features' (or structural encodings). Such augmentation can take many forms, and we outline two broad strategies. One way considers the graphs' structure as explicit features (Bodnar et al., 2021). By considering e.g. rings as explicit features to be learned, we construct more complex neighborhood structures with the consequence of being able to distinguish strictly more graph isomorphisms (see also Morris et al. (2019)). Though effective, these methods tend to be computationally expensive and typically require more apriori domain-knowledge to identify beforehand the structures that are likely to be relevant for the task.

A second approach is to include structural information in the node features, meaning that the initial feature and structural information are altered as follows:

$$\mathbf{h}_i^{\ell=0} = \mathsf{Encode}(\mathbf{h}_i^{\mathrm{in}}, \mathbf{p}_i^{\mathrm{in}}),$$

where $\mathbf{p}_i^{\mathrm{in}}$ denotes the structural information of node i. Typically, the encodings are formed by first concatenating $\mathbf{h}_i^{\mathrm{in}}$ and $\mathbf{p}_i^{\mathrm{in}}$ and then passing the resulting state through an MLP. Besides augmenting initial features only, alternative approaches such as updating the structural information through the message passing layers separately, have also been explored (Dwivedi et al., 2021). Augmenting node features with graph-aware structural information can be particularly useful for graph transformers, where structural information is used as positional information when passing messages (Ying et al., 2021; Kreuzer et al., 2021; Rampasek et al., 2022).

Structural (topological) information. Although different strategies for augmenting features with graph-information have been proposed, random-walk transition probabilities and Laplacian-eigenvectors constitute the most adopted classes of structural encodings (Dwivedi et al., 2021; Rampasek et al., 2022). A random walk is a Markov chain supported on the vertices of the graph with transition matrix $\mathbf{R}$ defined by $\mathbf{R} = \mathbf{A}\mathbf{D}^{-1}$ – if we take any probability distribution over the graph to be a row vector. The matrix $\mathbf{R}$ is typically raised to various powers to find structural information for random walks of different lengths, where either all entries in $\mathbf{R}^k$ are used for pairwise nodes, or just the diagonal entries to only describe landing probabilities of a node to itself (see Li et al. (2020); Dwivedi et al. (2021)

respectively). Random walks are powerful descriptors of the graph-topology – since they capture the diffusion properties of the graph – and are intimately related to the spectrum of the Laplacian (Chung & Graham, 1997). In fact, recently Black et al. (2023); Di Giovanni et al. (2023) proved that the phenomenon of over-squashing is likely to occur among nodes that are hard to be visited by a random walk, i.e. with large commute time.

An alternative to using random-walk features is found in spectral graph theory, i.e. the study of properties of graphs using the eigenvectors and eigenvalues of matrices describing the graph. For example, the eigenvectors of the graph Laplacian are informative about the connectedness of the graph, and as such are commonly used as structural features in GNNs. Recall that the Laplacian $\mathbf{L}$ of a graph is computed by taking the difference between degree and adjacency matrices, i.e. $\mathbf{L} = \mathbf{D} - \mathbf{A}$. For a Laplacian $\mathbf{L}$ with eigendecomposition $\mathbf{L} = \mathbf{U}^\top \mathbf{\Lambda} \mathbf{U}$, one can define $\mathbf{p}_i^{\mathrm{lap}} := \begin{bmatrix} \tilde{\mathbf{U}}_{i1} & \cdots & \tilde{\mathbf{U}}_{ik} \end{bmatrix}$, where $\tilde{\mathbf{U}}$ is matrix $\mathbf{U}$ reordered such that the kth row in $\tilde{\mathbf{U}}$ has the kth highest eigenvalue.

3. Tensor-product structural encoding

In this section, we introduce an alternative approach to encode structural information in GNNs that leverages tensor products, rather than the typical concatenation operation. We take inspiration from physics, and in particular the description of quantum states, and design an encoding operation that instead of concatenating input features and structural features, takes the tensor product of the representation.

A novel structural encoding. Formally, the *tensor* (or *Kronecker*) *product* $\otimes$ between matrices $\mathbf{A}$ and $\mathbf{B}$ is the matrix formed by replacing each element in $\mathbf{A}$ with the product of that element with $\mathbf{B}$, i.e.

$$\mathbf{A} \otimes \mathbf{B} = \begin{bmatrix} a_{11}\mathbf{B} & \cdots & a_{1n}\mathbf{B} \\ \vdots & \ddots & \vdots \\ a_{m1}\mathbf{B} & \cdots & a_{mn}\mathbf{B} \end{bmatrix}.$$

Therefore, when considering $\mathbf{a} \otimes \mathbf{b}$ for vectors $\mathbf{a} \in \mathbb{R}^d, \mathbf{b} \in \mathbb{R}^q$, we find that the product $a_i \cdot b_j$ is the element in index $i \cdot q + j$, i.e. being the vector found by concatenating vectors $\mathbf{b}$ scaled for each element in $\mathbf{a}$.

Typically, when constructing an encoding for our input features and structural information, the respective vectors are first concatenated and then projected, i.e.

$$(\mathbf{h}_i, \mathbf{p}_i) \mapsto \mathbf{W} \begin{bmatrix} \mathbf{h}_i \\ \mathbf{p}_i \end{bmatrix},$$

for some weight matrix $\mathbf{W} \in \mathbb{R}^{d_{\mathrm{hidden}} \times (d+q)}$. To embed the input features using a tensor product, we simply mimic this

approach, i.e.

$$(\mathbf{h}_i, \mathbf{p}_i) \mapsto \mathbf{W}(\mathbf{h}_i \otimes \mathbf{p}_i),$$

for some weight matrix $\mathbf{W} \in \mathbb{R}^{d_{\text{hidden}} \times (d \cdot q)}$. The main rationale behind our choice of studying a tensor product representation, is that it is now possible to construct a much higher-dimensional (and hence more expressive) node representation that accounts for both features and graph topology, even by starting from lower-dimensional entities.

However, notice that the number of inputs to the projection has increased significantly, which is something we account for in the experiments by significantly reducing the input dimension. More generally though, below we explore alternative ways of reducing the number of parameters that are inspired by tensor-product and the phenomenon of entanglement. We note that from now on, in both the discussion and in the experiments, we restrict to the case where the feature and the structural coordinates share the same dimension so that $(\mathbf{h}_i, \mathbf{p}_i)$ is mapped to a vector in $\mathbb{R}^{d_{\text{hidden}} \times d_{\text{hidden}}}$.

Sparsifying message-passing through entanglement. It is common for a message-passing layer to update the feature of a state by using a linear projection of the current state combined with the aggregation of the messages from its neighbours, followed by a non-linear activation. Put differently, for many common instances of MPNNs, the MPNNLayer operation in (1) can be taken to be an MLP. Accordingly, the vast majority of the parameters used by a GNN enter the MPNN layer through the weight matrices of the MLP. Since the proposed tensor-product encoding might have a larger contribution to the size of the neural network, and we are generally interested in assessing the importance and impact of the message-passing step in the GNN, we introduce a novel way of controlling the expressivity of MPNNLayer and its parameter count.

Suppose that $\mathbf{h}_i \in \mathbb{R}^{d^2}$ for some d – as in the cases of d-dimensional features augmented with d-dimensional structural encoding using a tensor product. We can then 'reshape' $\mathbf{h}_i$ to be a $d \times d$ matrix, which we denote by $\mathsf{Mat}(\mathbf{h}_i)$. We then propose to approximate the linear layer-projection in MPNNLayer with a series of small matrices $\{(\mathbf{W}_k, \mathbf{Q}_k)\}_{k=1}^K$ with $\mathbf{W}_k, \mathbf{Q}_k \in \mathbb{R}^{d \times d}$, by left and right multiplying the node-state $\mathsf{Mat}(\mathbf{h}_i)$, i.e.

$$\mathbf{h}_i \mapsto \sum_{k=1}^K \mathbf{W}_k \mathsf{Mat}(\mathbf{h}_i) \mathbf{Q}_k. \tag{2}$$

The update is then used on the current state and the aggregation over the 1-hop as per (1) before the action of a pointwise non-linear activation, e.g. for GIN with a one-layer MLP the state $\mathsf{Mat}(\mathbf{h}_i)$ would be updated as follows:

$$\mathsf{ReLU}(\sum_{k=1}^K \mathbf{W}_k((1+\epsilon) \cdot \mathsf{Mat}(\mathbf{h}_i) + \sum_{j \sim i} \mathsf{Mat}(\mathbf{h}_j))\mathbf{Q}_k).$$

Note that when using the sparse linear layers, the expressivity of the message passing procedure is constrained, and using a larger value of K increases its expressivity. We remark that the proposed sparsification reduces the number of parameters in each MPNN-layer from d^4 to $2K \times d^2$ parameters, i.e. a reduction of a factor of $\frac{d^2}{2K}$ parameters in general.

We note that if the node-state satisfies $\mathbf{h}_i' = \mathbf{h}_i \otimes \mathbf{p}_i$, then reshaping the tensor-product state into a matrix, simply corresponds to writing it as the outer product $\mathbf{h}_i \mathbf{p}_i^\top$, meaning that the sparse layer-update in (2) takes the form

$$\mathbf{h}_i' \mapsto \sum_{k=1}^K (\mathbf{W}_k \mathbf{h}_i)(\mathbf{Q}_k \mathbf{p}_i)^\top = \sum_{k=1}^K (\mathbf{W}_k \otimes \mathbf{Q}_k)(\mathbf{h}_i \otimes \mathbf{p}_i)$$
$$= \sum_{k=1}^K (\mathbf{W}_k \otimes \mathbf{Q}_k)\mathbf{h}_i'.$$

Accordingly, the number K determines the level of 'entanglement' between the states $\mathbf{h}_i$ and $\mathbf{p}_i$ at node i. In fact, each linear matrix on the tensor (outer) product space can be decomposed as a sum of tensor products of smaller matrices; since any matrix of shape $(d \times d) \times (d \times d)$ can be represented using a linear combination of at most $K = d^4$ of such tensor product matrices, the larger K, the more entanglement we can model and hence the more degrees of freedom we can model in the tensor product space. *This is a more geometric approach to controlling the size of the weights and hence the number of parameters in the message-passing layer.*

We finally stress that even for states that are not in a tensor-product form, the sparsification introduced in (2) still makes sense, and will in fact be used for the concatenation case as well in our experiments.

4. Empirical analysis

The aim of the experimental evaluation is twofold. First, we evaluate the difference in performance when using tensor encoding versus concatenation encoding by comparing their performances for multiple MPNN layers and datasets. Second, to better understand to what extent the graph structure can be captured without (much) message passing by both types of encodings, a comparison is done in which a low parameter regime for the message passing is considered. The capacity of the message passing is constrained by reducing either the number of weights or the number of layers – ultimately removing the message passing altogether.

To ensure the models considered can embed the structural information well, the input features and structural information are first embedded separately before either concatenation or a tensor product is applied, i.e.

$$\mathbf{h}_i^{\ell=0} = \mathsf{Encode}(\mathbf{W}_h\mathbf{h}_i, \mathbf{W}_p\mathbf{p}_i),$$

where Encode is either the concatenation or tensor product encoding and $\mathbf{W}_h, \mathbf{W}_p$ denote linear projections of the input features and structural information respectively. Moreover, the readout is done by summing over the final hidden states and passing the resulting state through a two-layer MLP for the final prediction.

To compensate for the number of features in the tensor encodings compared to the case of concatenation, smaller dimensions are provided to the weight matrices $\mathbf{W}_h$ and $\mathbf{W}_p$ when a tensor product is used. Namely, when using a resulting hidden dimension in the layers of d^2 features, for tensor the weight matrices embed the inputs to d-dimensional vectors whereas in concatenation the matrices embed the inputs to $\frac{d^2}{2}$ features. As such, this comparison is restricted to using hidden dimensionalities of d^2 features for some even number d and ensures that both approaches share similar parameters.

The message passing layers considered in this work are GCN, GIN, and SAGEConv. The effect of the structural encodings is compared when 1) not reducing the number of parameters in the linear layers, referred to as 'full' message passing, 2) using the sparse linear layers as seen in (2) for two different levels of entanglement, referred to as 'sparse' message passing, and 3) when removing the message passing layers altogether. Note that when doing either sparse or no message passing, we do not compensate for the reduction in parameters in any way, implying that the sparse and no message passing models have significantly fewer weights. Moreover, when no message passing is done, the model is reduced to only the encoding and decoding phases, meaning that all the graph structure must be learned through the structural encodings *only*. Last, an ablation study is conducted for GCN to compare the effect of reducing both the number of layers and the number of parameters in the models.

For the main experiments, a 4-layer MPNN architecture with approximately 500K parameters using full message passing is used as the point of reference for each of the MPNN layer types. The corresponding hidden dimension is then also used when doing sparse message passing and no message passing in which cases the number of parameters is severely reduced. We report the final training and test performance averaged over 3 runs. We also denote the relative improvement of tensor-encodings ('gain'). In all experiments, we use the structural encoding

$$\mathbf{p}_i := \begin{bmatrix} \mathbf{R}_{ii} & \mathbf{R}_{ii}^2 & \cdots \mathbf{R}_{ii}^k \end{bmatrix},$$

i.e. the diagonal entries of the random walk matrix $\mathbf{R}$ with random walks up to length 20.

All models are optimized using Adam with a learning rate of 10^{-3} using a scheduler halving the learning rate whenever the model does not improve for 25 epochs. Training is stopped when the learning rate drops below 10^{-5}. The regression tasks are optimized based on mean average error (MAE) which is the metric reported, whereas the classification is optimized using cross entropy and the reported metric is the average precision (AP). For all main results, we provide standard deviations over the multiple runs in Appendix A, though these are omitted in the main text for visual clarity.

4.1. Results

Peptides. The Long Range Graph Benchmark is a collection of five graph-based datasets with tasks that are based on long-range dependencies in graphs. Two of these datasets - Peptides-struct and Peptides-func - are multi-class regression and classification in the domain of chemistry. Results are reported as train/test pairs and are provided in Table 1. These datasets have been introduced as benchmarks for tasks that may exhibit long-range interactions, and have in fact been used by graph-transformers or GNN-models that are designed to reduce over-squashing (Rampasek et al., 2022; Gutteridge et al., 2023).

When considering full weight matrices, a tensor encoding performs at least as well as concatenation, never leading to a significant decrease in test performance. Furthermore, we observe that using sparse linear layers in the MPNNs does not lead to stark decreases in performance in general. For the Peptides-struct task, we observe that both the final train MAE and test MAE are quite similar to the performance when using the approximate layers, suggesting that indeed the *role of MPNN layers in this task might not be as crucial – or at least that over-parameterization of the message-passing layers is redundant* – something which is emphasized by the fact that even on the train data the model does not learn more when using full weight matrices compared to the approximate layers. On the Peptides-func task, we also notice that the difference between the test performances of the full weights versus the approximate layers is very minimal.

On the Peptides-struct task, we observe that not using MPNN layers at all leads to a stark decrease in performance compared to using the full weights in each layer when the encodings are formed through concatenation, whereas when using a tensor encoding such a decrease in performance is not present. A similar effect is seen in the Peptides-func task, where even though both encoding types perform worse when no MPNN layers are added, this decrease in performance is much stronger when concatenation is used. This

Table 1. Results on Peptides-Struct (MAE) and Peptides-Func (AP). All results are averaged over 3 runs with different seeds.

	Peptides-Struct (MAE $\downarrow$)			
State Type	full	$K = 10$	$K = 1$	no MP
GCN RW-Concat	0.212 / 0.255	0.21 / 0.255	0.226 / 0.251	0.64 / 0.643
RW-Tensor	0.212 / 0.252	0.206 / 0.254	0.223 / 0.249	0.249 / 0.268
Gain	1.0 / 1.012	1.019 / 1.004	1.013 / 1.008	2.57 / 2.399
GIN RW-Concat	0.5 / 0.29	0.64 / 0.555	0.225 / 0.252	0.64 / 0.643
RW-Tensor	0.239 / 0.257	0.498 / 0.309	0.22 / 0.249	0.25 / 0.266
Gain	2.092 / 1.128	1.285 / 1.796	1.023 / 1.012	2.56 / 2.417
SAGE RW-Concat	0.21 / 0.254	0.218 / 0.256	0.227 / 0.251	0.64 / 0.643
RW-Tensor	0.198 / 0.253	0.219 / 0.253	0.228 / 0.25	0.254 / 0.267
Gain	1.061 / 1.004	0.995 / 1.012	0.996 / 1.004	2.52 / 2.408
	Peptides-Func (AP $\uparrow$)			
State Type	full	$K = 10$	$K = 1$	no MP
GCN RW-Concat	0.906 / 0.613	0.877 / 0.598	0.786 / 0.606	0.509 / 0.468
RW-Tensor	0.875 / 0.607	0.846 / 0.595	0.784 / 0.593	0.766 / 0.586
Gain	0.966 / 0.99	0.965 / 0.995	0.997 / 0.979	1.505 / 1.252
GIN RW-Concat	0.672 / 0.577	0.165 / 0.328	0.847 / 0.596	0.548 / 0.49
RW-Tensor	0.921 / 0.611	0.923 / 0.601	0.854 / 0.606	0.736 / 0.582
Gain	1.371 / 1.059	5.594 / 1.832	1.008 / 1.017	1.343 / 1.188
SAGE RW-Concat	0.907 / 0.607	0.844 / 0.594	0.773 / 0.596	0.555 / 0.501
RW-Tensor	0.929 / 0.616	0.842 / 0.607	0.776 / 0.601	0.724 / 0.579
Gain	1.024 / 1.015	0.998 / 1.022	1.004 / 1.008	1.305 / 1.156

suggests that tensor encodings are able to capture more complex information in their encodings, supporting our intuition for using tensor encodings in MPNNs.

Last, we observe that even without message passing layers we are competitive with graph transformers on the Peptides-struct task. This begs the question of to what extent this dataset contains long-range interactions, and more specifically to what extent such interactions can also be captured by strong structural encodings.

ZINC. The dataset ZINC is a molecular dataset with molecules up to 38 heavy atoms. The task is to predict the penalized logP score, used for training molecular generation models. A subset of 12,000 molecules is used following Dwivedi et al. (2020). Results are again reported as train/test pairs and are provided in Table 2.

Similarly to the results observed in the Long Range Graph Benchmark, tensor encodings are again mostly on par with concatenation in all scenarios. Moreover, we again observe that sparse MPNN layers are able to perform on par with full weight matrices, suggesting that also in this task there is no need to over-parametrize the message passing layers. Last, we again observe a significant gap in the performance be-

tween tensor encodings and concatenation encodings when no MPNN layers are used, again solidifying our intuition that tensor encodings are strictly more expressive encodings for topological information.

Ablation. The ablation study aims to reduce the message passing layers even further by decreasing both the hidden dimension and the number of layers in the message passing phase. The performance is compared again for full message passing, sparse message passing with an entanglement of $K = 1$, and no message passing. Moreover, the joint encoder is removed, meaning that the hidden representations in the first layer are simply found by either concatenating or taking the tensor product of the individual encodings. The hidden dimensions considered are $\{16, 36, 64, 328\}$, and compared using one layer ($L = 1$) and four layers ($L = 4$). The results reported are for the Peptides-struct task using a GCN, and are again reported as train/test pairs in Table 3.

In the case of full message passing, we observe that we can reduce the number of layers without noticing a significant drop in performance when tensor encodings are used, whereas such a drop is observed in the concatenation case. We also observe that we can reduce the hidden dimension

Table 2. Results on ZINC (MAE). All results are averaged over 3 runs with different seeds.

	State Type	full	$K = 10$	$K = 1$	no MP
		ZINC (MAE $\downarrow$)			
GCN	RW-Concat	0.02 / 0.207	0.043 / 0.233	0.135 / 0.24	0.466 / 0.607
	RW-Tensor	0.043 / 0.209	0.052 / 0.22	0.12 / 0.219	0.324 / 0.527
	Gain	0.465 / 0.99	0.827 / 1.059	1.125 / 1.096	1.438 / 1.152
GIN	RW-Concat	0.076 / 0.263	0.04 / 0.281	0.116 / 0.279	0.46 / 0.609
	RW-Tensor	0.075 / 0.242	0.025 / 0.249	0.127 / 0.252	0.361 / 0.538
	Gain	1.013 / 1.087	1.6 / 1.129	0.913 / 1.107	1.274 / 1.132
SAGE	RW-Concat	0.021 / 0.192	0.03 / 0.176	0.104 / 0.193	0.457 / 0.614
	RW-Tensor	0.023 / 0.201	0.04 / 0.177	0.121 / 0.205	0.367 / 0.552
	Gain	0.913 / 0.955	0.75 / 0.994	0.86 / 0.941	1.245 / 1.112

Table 3. Ablation on Peptides-Struct (MAE). All results are averaged over 3 runs with different seeds.

		Peptides-Struct (MAE $\downarrow$)			
		Full message passing			
$L = 1$	Parameters	689	2557	7169	162109
	RW-Concat	0.64 / 0.643	0.64 / 0.643	0.395 / 0.397	0.254 / 0.261
	RW-Tensor	0.283 / 0.282	0.269 / 0.269	0.255 / 0.26	0.226 / 0.255
$L = 4$	Parameters	1457	6445	19457	477037
	RW-Concat	0.279 / 0.281	0.264 / 0.267	0.248 / 0.259	0.219 / 0.253
	RW-Tensor	0.281 / 0.279	0.258 / 0.261	0.247 / 0.257	0.213 / 0.252
		Sparse message passing ($K = 1$)			
$L = 1$	Parameters	465	1333	3201	57781
	RW-Concat	0.321 / 0.321	0.398 / 0.399	0.271 / 0.271	0.242 / 0.257
	RW-Tensor	0.296 / 0.294	0.276 / 0.276	0.27 / 0.27	0.242 / 0.256
$L = 4$	Parameters	561	1549	3585	59725
	RW-Concat	0.294 / 0.291	0.275 / 0.276	0.264 / 0.268	0.229 / 0.252
	RW-Tensor	0.3 / 0.299	0.272 / 0.272	0.268 / 0.269	0.228 / 0.251
		No message passing			
$L = 0$	Parameters	433	1261	3073	57133
	RW-Concat	0.64 / 0.643	0.64 / 0.642	0.527 / 0.528	0.352 / 0.351
	RW-Tensor	0.302 / 0.299	0.401 / 0.401	0.272 / 0.274	0.248 / 0.26

significantly when using tensor encodings without losing much performance, e.g. multiple models with only a few thousand parameters are able to perform almost on par with the 500K parameter model. This supports the claim that, for this task, structural encoding could be sufficient to leverage the graph structure, and that *tensor encodings are significantly more able to capture this information well in sparse and shallow regimes*.

Similarly, when doing sparse message passing, we observe that the effect of reducing the number of layers or hidden di-

mensions has a much more significant negative effect when the encodings are formed through concatenation than when they are formed through tensor encodings. Finally, again when no message passing is used at all, the tensor encodings are much more robust in the low-parameter regimes.

Is the choice of geometric encoding important? The results obtained on ZINC as well as those on Peptides would highlight that when using the full message-passing layer, it seems that there is no significant difference in terms of

performance between either of the two, despite the two approaches being geometrically very different. This brings up the question of what the message-passing is actually learning based on the structural information, and to what extent the message-passing layer is actually needed. Our investigation supports the idea that the tensor-product encoding is significantly more resilient to the reduction of the number of parameters in the MPNN, or in fact the removal of the MPNN all together, in contrast to the concatenation approach where instead eliminating the message-passing leads to serious performance degradation. We speculate that one reason for the robustness of the tensor product to the 'size' (or presence) of the message-passing as opposed to the concatenation operation, resides in its ability to model more complex and high-dimensional entangled representation which can then be leveraged by an MLP without the usage of the graph.

5. Discussion

Conclusion and future research. In this work, the effect of the topologically informed node encodings in MPNNs and the role of message passing have been explored. For this, we have proposed a tensor based encoding and compared this against the standard encoding done through concatenation. Moreover, we have looked at the effect of reducing the weights in the MPNN layers or removing the message-passing step all-together, investigating such approaches across different MPNN architectures.

Our results indicate that when no MPNN layers are included in the model, tensor encodings are able to better learn on graphs in general, and even able to perform on par with models with MPNN layers in some tasks. Moreover, since tensor encodings are almost always at least on par with using standard concatenation, we conclude that tensor encodings form a promising new approach for learning on graphs with structural information. For future research, a natural question is to see if tensor encodings can also improve graph-agnostic architectures such as transformers.

Moreover, we observe that in the cases where excluding all MPNN layers led to a significant reduction in performance, much of that performance could also be obtained using very sparse weight matrices. This suggests that the role of MPNN layers in these structural tasks is minimal. An interesting avenue to explore relating to this result is to study what MPNNs learn, and whether it is possible to reduce the parameters in MPNN layer.

References

Bapst, V., Keck, T., Grabska-Barwińska, A., Donner, C., Cubuk, E. D., Schoenholz, S. S., Obika, A., Nelson, A. W., Back, T., Hassabis, D., et al. Unveiling the predictive power of static structure in glassy systems. *Nature Physics*, 16(4):448–454, 2020.

Black, M., Nayyeri, A., Wan, Z., and Wang, Y. Understanding oversquashing in gnns through the lens of effective resistance. *arXiv preprint arXiv:2302.06835*, 2023.

Bodnar, C., Frasca, F., Wang, Y., Otter, N., Montúfar, G. F., Lió, P., and Bronstein, M. M. Weisfeiler and lehman go topological: Message passing simplicial networks. In *International Conference on Machine Learning*, pp. 1026–1037, 2021.

Bouritsas, G., Frasca, F., Zafeiriou, S., and Bronstein, M. Improving graph neural network expressivity via subgraph isomorphism counting. 2022.

Bruna, J., Zaremba, W., Szlam, A., and LeCun, Y. Spectral networks and locally connected networks on graphs. In *International Conference on Learning Representations*, 2014.

Chen, Z., Chen, L., Villar, S., and Bruna, J. Can graph neural networks count substructures? *Advances in neural information processing systems*, 33:10383–10395, 2020.

Chung, F. R. and Graham, F. C. *Spectral graph theory*. American Mathematical Soc., 1997.

Defferrard, M., Bresson, X., and Vandergheynst, P. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in neural information processing systems*, volume 29, 2016.

DeZoort, G., Battaglia, P. W., Biscarat, C., and Vlimant, J.-R. Graph neural networks at the large hadron collider. *Nature Reviews Physics*, pp. 1–23, 2023.

Di Giovanni, F., Giusti, L., Barbero, F., Luise, G., Lio, P., and Bronstein, M. On over-squashing in message passing neural networks: The impact of width, depth, and topology. In *International Conference on Machine Learning*, 2023.

Dwivedi, V. P., Joshi, C. K., Laurent, T., Bengio, Y., and Bresson, X. Benchmarking graph neural networks. abs/2003.00982, 2020.

Dwivedi, V. P., Luu, A. T., Laurent, T., Bengio, Y., and Bresson, X. Graph neural networks with learnable structural and positional representations. 2021.

Geerts, F. and Reutter, J. L. Expressiveness and approximation properties of graph neural networks. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=wIzUeM3TAU.

Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., and Dahl, G. E. Neural message passing for quantum chemistry. In *International Conference on Machine Learning*, pp. 1263–1272. PMLR, 2017.

Goller, C. and Kuchler, A. Learning task-dependent distributed representations by backpropagation through structure. In *Proceedings of International Conference on Neural Networks (ICNN'96)*, volume 1, pp. 347–352. IEEE, 1996.

Gori, M., Monfardini, G., and Scarselli, F. A new model for learning in graph domains. In *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, volume 2, pp. 729–734. IEEE, 2005.

Gutteridge, B., Dong, X., Bronstein, M., and Di Giovanni, F. Drew: Dynamically rewired message passing with delay. *arXiv preprint arXiv:2305.08018*, 2023.

Hamilton, W. L., Ying, R., and Leskovec, J. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*, pp. 1025–1035, 2017.

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

Kipf, T. N. and Welling, M. Semi-Supervised Classification with Graph Convolutional Networks. In *International Conference on Learning Representations*, 2017.

Kreuzer, D., Beaini, D., Hamilton, W., Létourneau, V., and Tossou, P. Rethinking graph transformers with spectral attention. In *Advances in Neural Information Processing Systems*, volume 34, pp. 21618–21629, 2021.

Li, P., Wang, Y., Wang, H., and Leskovec, J. Distance encoding: Design provably more powerful neural networks for graph representation learning. *Advances in Neural Information Processing Systems*, 33:4465–4478, 2020.

Maron, H., Ben-Hamu, H., Shamir, N., and Lipman, Y. Invariant and equivariant graph networks. 2018.

Mirhoseini, A., Goldie, A., Yazgan, M., Jiang, J. W., Songhori, E., Wang, S., Lee, Y.-J., Johnson, E., Pathak, O., Nazi, A., et al. A graph placement methodology for fast chip design. *Nature*, 594(7862):207–212, 2021.

Morris, C., Ritzert, M., Fey, M., Hamilton, W. L., Lenssen, J. E., Rattan, G., and Grohe, M. Weisfeiler and leman go neural: Higher-order graph neural networks. In *AAAI Conference on Artificial Intelligence*, pp. 4602–4609. AAAI Press, 2019.

Rampasek, L., Galkin, M., Dwivedi, V. P., Luu, A. T., Wolf, G., and Beaini, D. Recipe for a general, powerful, scalable graph transformer. In *Advances in Neural Information Processing Systems*, 2022.

Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., and Monfardini, G. The graph neural network model. *IEEE transactions on neural networks*, 20(1):61–80, 2008.

Sperduti, A. Encoding labeled graphs by labeling raam. In *Advances in Neural Information Processing Systems*, volume 6, 1993.

Stokes, J. M., Yang, K., Swanson, K., Jin, W., Cubillos-Ruiz, A., Donghia, N. M., MacNair, C. R., French, S., Carfrae, L. A., Bloom-Ackermann, Z., et al. A deep learning approach to antibiotic discovery. *Cell*, 180(4): 688–702, 2020.

Weisfeiler, B. and Leman, A. The reduction of a graph to canonical form and the algebra which appears therein. *nti, Series*, 2(9):12–16, 1968.

Xu, K., Hu, W., Leskovec, J., and Jegelka, S. How powerful are graph neural networks? In *International Conference on Learning Representations*, 2019.

Ying, C., Cai, T., Luo, S., Zheng, S., Ke, G., He, D., Shen, Y., and Liu, T.-Y. Do transformers really perform badly for graph representation? In *Advances in Neural Information Processing Systems*, volume 34, pp. 28877–28888, 2021.

A. Detailed results

Table 4. Test results of main experiments reported with standard deviation. All results are averaged over 3 runs with different seeds.

	State Type	Layer	full	$K = 10$	$K = 1$	no MP
			Test performance with standard deviation			
Peptides–Struct	RW-Concat	gcn	0.255 ± 0.007	0.255 ± 0.001	0.251 ± 0.006	0.643 ± 0.000
	RW-Tensor	gcn	0.252 ± 0.001	0.254 ± 0.001	0.249 ± 0.001	0.268 ± 0.002
	RW-Concat	gin	0.290 ± 0.244	0.555 ± 0.000	0.252 ± 0.006	0.643 ± 0.000
	RW-Tensor	gin	0.257 ± 0.005	0.309 ± 0.045	0.249 ± 0.001	0.266 ± 0.001
	RW-Concat	sage	0.254 ± 0.004	0.256 ± 0.010	0.251 ± 0.003	0.643 ± 0.000
	RW-Tensor	sage	0.253 ± 0.002	0.253 ± 0.003	0.250 ± 0.001	0.267 ± 0.002
Peptides-Func	RW-Concat	gcn	0.613 ± 0.001	0.598 ± 0.014	0.606 ± 0.009	0.468 ± 0.028
	RW-Tensor	gcn	0.607 ± 0.013	0.595 ± 0.010	0.593 ± 0.006	0.586 ± 0.008
	RW-Concat	gin	0.577 ± 0.44	0.328 ± 0.000	0.596 ± 0.016	0.490 ± 0.033
	RW-Tensor	gin	0.611 ± 0.008	0.601 ± 0.003	0.606 ± 0.014	0.582 ± 0.006
	RW-Concat	sage	0.607 ± 0.010	0.594 ± 0.013	0.596 ± 0.009	0.501 ± 0.022
	RW-Tensor	sage	0.616 ± 0.014	0.607 ± 0.005	0.601 ± 0.007	0.579 ± 0.007
Zinc	RW-Concat	gcn	0.207 ± 0.010	0.233 ± 0.008	0.24 ± 0.001	0.607 ± 0.003
	RW-Tensor	gcn	0.209 ± 0.002	0.220 ± 0.001	0.219 ± 0.010	0.527 ± 0.003
	RW-Concat	gin	0.263 ± 0.010	0.281 ± 0.005	0.279 ± 0.008	0.609 ± 0.014
	RW-Tensor	gin	0.242 ± 0.008	0.249 ± 0.008	0.252 ± 0.005	0.538 ± 0.008
	RW-Concat	sage	0.192 ± 0.003	0.176 ± 0.014	0.193 ± 0.004	0.614 ± 0.005
	RW-Tensor	sage	0.201 ± 0.003	0.177 ± 0.007	0.205 ± 0.002	0.552 ± 0.006