SHELLGATEDGNN: LONG-RANGE MESSAGE PASSING FOR 2D MOLECULAR ELECTRONIC PROPERTY PREDICTION

Anonymous authors
Paper under double-blind review

ABSTRACT

Message passing neural networks (MPNNs) provide a general framework for molecular graph modeling. To capture long-range structural information in a molecule, the standard approach is to stack many layers of 1-hop message passing; however, such propagation does not capture complex topology precisely and is prone to oversmoothing as it gets deeper. We propose SGGNN (ShellGatedGNN), a long-range message-passing architecture. It extends classical gated message passing from a single edge to the triple (sender, sender's environment, receiver), where the sender's environment is the structure around the sender. We divide the nodes into shells by their distance to the receiver, a common grouping strategy, so that the nodes in one shell are at the same distance. This shelling is permutation-equivariant, so SGGNN can use a concatenation instead of a sum in the node update, which keeps more information. On QM9 and QM40, SGGNN attains the best performance among the strong baselines on all six electronic properties. On the HOMO-LUMO gap regression task of the large-scale PCQM4Mv2 benchmark, it outperforms the message-passing baselines. Our code is available at the anonymous repository `https://anonymous.4open.science/r/SGGNN-114514A`.

1 INTRODUCTION

Molecule-level electronic property prediction is a graph-level regression problem and a common downstream use of graph neural networks (GNNs). Such predictors can substitute for first-principles (DFT) calculations, which are expensive and hard to scale over large chemical spaces. They are a central tool for high-throughput virtual screening in drug discovery and materials design. Molecular prediction methods that use 3D spatial conformations (Schütt et al., 2017; Gasteiger et al., 2020) take the conformer as input, and conformers are expensive to obtain over large chemical spaces. Methods that take the 2D molecular graph as input need no conformer, so they cost less and are better suited to large-scale screening.

Neural message passing has become a standard framework for molecular graph modeling. To capture long-range structural information in a molecule, existing message-passing methods fall into two groups. The first stacks 1-hop messages layer by layer, as in the classical GCN (Kipf & Welling, 2017) and GIN (Xu et al., 2019) or their strengthened variants GCN+ and GIN+ (Luo et al., 2025): every additional layer aggregates the local neighborhood again, and deep stacks are prone to oversmoothing. The second models long-range dependence directly, without stacking, for instance virtual nodes (Pham et al., 2017) or virtual edges that link distant nodes directly (Gilmer et al., 2017). What such shortcuts ignore is the actual topological connectivity between the two endpoints.

Addressing the long-range difficulty in molecular electronic-property prediction requires reexamining what a receptive field means on a graph. Existing methods assume by default a linearly expanding receptive field, that is, messages propagate along edges layer by layer. Real molecular topology, however, has branching and cyclic structure. A receptive field on a molecular graph is therefore a rooted, distance-layered subgraph whose connections within and between shells are not unique. Current mainstream GNN methods do not capture this branching, cyclic shell topology precisely, which limits the expressive power of the models.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

We propose a new long-range message passing: the receiver's multi-hop neighbors are divided into shells by breadth-first-search (BFS) distance, and the sender's environment is fused into both the message and the gate, so that gated aggregation produces the shell representations, which are then concatenated and fused to update the receiver's node representation. A single message-passing layer can therefore capture structure far beyond one hop. In summary, our contributions are as follows:

- **A long-range message-passing architecture.** We propose the long-range message-passing architecture SGGNN. In classical gated message passing the gate acts on a single edge; in SGGNN it acts on the messages of the node pairs in each shell, so its reach extends from the 1-hop neighborhood to multiple shells.
- **Modular sender-environment design.** The sender's environment enters the layer only through the environment vector, so its internal operators can be replaced without changing the rest of the layer; we give one concrete design consisting of two sum vectors that describe the sender's environment in the shells.
- **A node update that preserves more information.** SGGNN differs from a conventional MPNN in its node update: the shell representations obtained by gated aggregation are concatenated first and projected once, which keeps the per-depth information separate. This uses the permutation equivariance of the division into shells obtained from the BFS distance.
- **Best performance among models of the same type.** SGGNN attains the best performance on all six electronic properties of QM9 and QM40 using only two sum vectors for the sender's environment, and on the large-scale PCQM4Mv2 benchmark it also outperforms the baselines of the same kind.

2 Related Work

Molecular property prediction. The goal is to predict quantum-chemical, physical or pharmacological properties from molecular structure, and early methods relied mainly on molecular fingerprints, such as ECFP (Rogers & Hahn, 2010), together with classical machine-learning models such as random forests (Breiman, 2001) and gradient-boosted trees (Chen & Guestrin, 2016). In recent years, representation learning with graph neural networks has become a standard approach. Gilmer et al. (Gilmer et al., 2017) unified these models into the general *message passing* framework and demonstrated it on QM9; on molecules, Attentive FP (Xiong et al., 2020) couples attention with an attentive graph-level readout for drug discovery. D-MPNN / Chemprop (Yang et al., 2019; Heid et al., 2024) propagates messages along directed edges, so that the two directions of every edge are updated independently; a message is not sent back along the edge it arrived on, which avoids the repeated passing of messages over one edge (tottering). Some three-dimensional methods also reach very high performance: SchNet (Schütt et al., 2017) encodes interatomic *distances* through continuous-filter convolutions, and DimeNet (Gasteiger et al., 2020) additionally encodes bond angles. Our objective is to capture long-range and complex topology in molecules and thereby to improve molecular property prediction.

Long-range GNNs. MixHop (Abu-El-Haija et al., 2019) mixes several powers of the adjacency $\mathbf{A}^j$ in one layer. k-hop GNN (Nikolentzos et al., 2020) aggregates the whole k-hop neighborhood in one layer. PathNN (Michel et al., 2023) changes the *unit of computation* to path sets. AIMNet-X2D (Nandakumar, 2025) groups nodes into shells by BFS distance and convolves over the shells. DuAxMPNN (Cui & Sun, 2026) turns 2–4-hop node pairs into explicit multi-hop edges convolved per hop, but reuses the same node pair across several hops. A virtual node is connected to every node, and virtual edges join distant nodes directly (Pham et al., 2017; Gilmer et al., 2017). Nested GNN (Zhang & Li, 2021) runs a GNN on a subgraph around each node and takes its result as the root representation. Another class relies on global attention: graph transformers such as Graphormer (Ying et al., 2021) let pairs of nodes interact by attention, with the graph structure encoded into the model, and GraphGPS (Rampášek et al., 2022) combines such attention with local message passing. These methods likewise cover long range, and our design differs from them in two respects, the input of the gate and the sender's environment. The gate is computed from the receiver, the sender and the sender's own environment, and that environment describes the structure around the sender.

Gating in message passing. Sigmoid gating is a standard mechanism for controlling information flow (Hochreiter & Schmidhuber, 1997): LSTM introduces sigmoid gates that modulate information element-wise. In message passing, GGNN (Li et al., 2016) adopted GRU-style gating early, merging the aggregated neighbor state into the node state through a gate, so the gate decides how much of the neighborhood the node keeps. GatedGCN (Bresson & Laurent, 2017) moves the gate from the node to the individual edges. It weights each neighbor's message by a learnable per-edge sigmoid gate, computed from the receiver node, the sender node and the edge features, and it updates the edge representation at every layer. Edge gating makes effective use of edge features and fits edge-featured molecular graphs well. GNN+ (Luo et al., 2025) strengthens GatedGCN at the model level, including normalization (Ioffe & Szegedy, 2015), dropout (Srivastava et al., 2014) and residual connections (He et al., 2016); enhanced this way, GatedGCN+ attains very strong performance on graph-level tasks. These gated methods, however, remain inside the classical MPNN framework: one convolution layer aggregates only the 1-hop neighbors, and a message travels along a single edge. In our design, the division into shells and the sender's environment extend the reach of the gate to a long range.

3 METHOD

Notation. Let $G = (V, E)$ be an undirected graph whose node set is V and whose edge set is E, with $N = |V|$ nodes. Its adjacency matrix is $\mathbf{A} \in \{0,1\}^{N\times N}$, its node feature matrix is $\mathbf{X} \in \mathbb{R}^{N\times F}$, and its edge feature tensor is $\mathbf{E} \in \mathbb{R}^{N\times N\times C}$. The *BFS distance matrix* $\mathbf{D} \in \mathbb{R}^{N\times N}$ is obtained from G by breadth-first search; each of its entries $\mathbf{D}[a, b]$ is the number of edges on a shortest path between nodes a and b. For an ordered pair (i, j) we call i the *receiver* and j the *sender*, at distance $d = \mathbf{D}[i, j]$. A node p is a *parent* of the sender j if $\mathbf{A}[p, j] = 1$ and $\mathbf{D}[i, p] = d - 1$, and the edge $\{p, j\}$ is then its *parent edge*; a sender can have several parents. $\mathbf{x}_i^{(l)}$ is the hidden feature of node i at layer l ($0 \le l \le L$), with L stacked layers and hidden width H; $\|$ denotes concatenation.

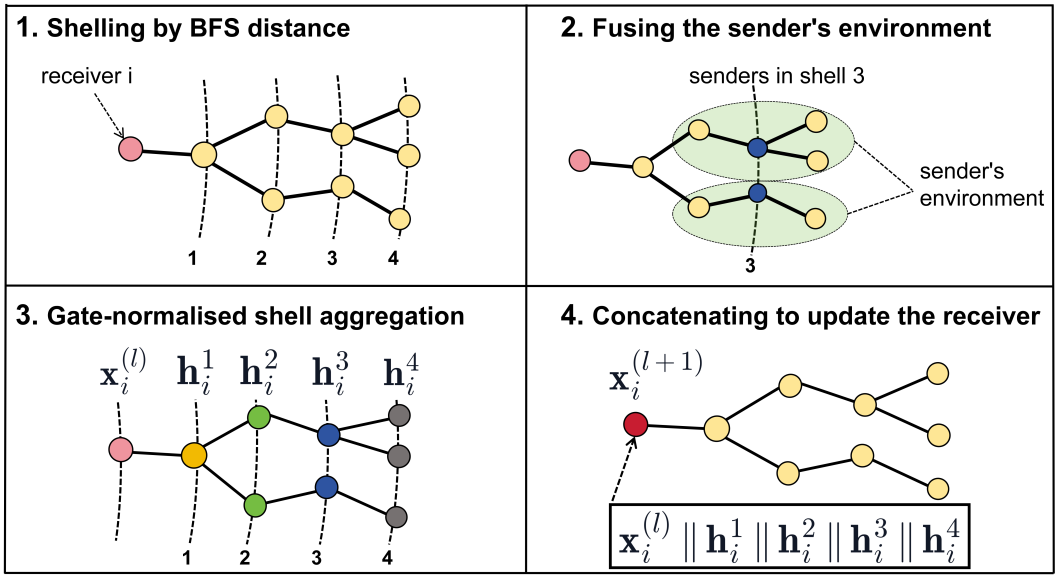


Figure 1: SGGNN message passing from the viewpoint of a receiver node i, in four steps. **1)** Shelling by BFS distance: the nodes are grouped into shells by their distance to i. **2)** Fusing the sender's environment into the message and the gate. **3)** Gate-normalized aggregation within a shell, giving $\mathbf{h}_i^d$. **4)** Concatenating the shell representations with the receiver's representation and fusing them to update $\mathbf{x}_i^{(l)}$ into $\mathbf{x}_i^{(l+1)}$.

Overview and motivation. Figure 1 shows the four steps of one layer for a receiver i. The division into shells produced in the first step is permutation-equivariant, which allows the shell representa-

tions to be concatenated and fused in the fourth step instead of summed, so that more information is preserved. The sender's environment in the second step uses exactly the well-defined relative position that shelling provides. The gate in the third step again uses exactly that environment to modulate the information flow with more discrimination, which in turn enhances the shell representations.

3.1 SHELLING BY THE BFS DISTANCE

The nodes at distance d from a receiver i form the d-th *shell* (Figure 1, step 1). $D_{\max}$ is a hyperparameter; each receiver covers shells up to distance $D_{\max}$. The node pairs (i, j) at distance d form the depth-d set

$$\mathcal{I}_d = \{(i, j) : \mathbf{D}[i, j] = d\}. \tag{1}$$

3.2 FUSING THE SENDER'S ENVIRONMENT

We design two sum vectors to describe the sender's environment (the design motivation is in Appendix B.3). The first is the pair of edge feature sums over the edges incident to the sender, $\mathbf{pes}^d_{ij}$ and $\mathbf{ses}^d_{ij}$, which describe the sender's local environment in the *receiver's* shell coordinates; the edge features are zero on pairs that are not connected, so only connected pairs contribute. The second is the parent ledger state $\mathbf{pls}^d_{ij}$, which reaches further by summing what the sender's parents have accumulated (Appendix D.1 characterizes which nodes it covers). All three are indexed by the node pair and are permutation-equivariant, $\mathbf{pes}'^d_{\pi(i)\pi(j)} = \mathbf{pes}^d_{ij}$ and likewise for $\mathbf{ses}$ and $\mathbf{pls}$ (Appendix B.3).

$$\mathbf{pes}^d_{ij} = \sum\nolimits_p \mathbf{1}\{\mathbf{D}[i, p] = d - 1\}\, \mathbf{E}[p, j], \tag{2}$$

$$\mathbf{ses}^d_{ij} = \sum\nolimits_k \mathbf{1}\{\mathbf{D}[i, k] = d\}\, \mathbf{E}[k, j], \tag{3}$$

$$\mathbf{e}^d_{ij} = \mathbf{W}^d_{\text{edge}}\big[\, \mathbf{pes}^d_{ij} \,\big\|\, \mathbf{ses}^d_{ij} \,\big]. \tag{4}$$

where $\mathbf{1}\{\cdot\}$ evaluates to 1 when the condition it encloses holds and to 0 otherwise, and $\mathbf{W}^d_{\text{edge}}$ is a depth-specific linear map that takes the two sums to the hidden width H.

The parent-edge feature sum. The parent-edge feature sum $\mathbf{pes}^d_{ij}$ adds up the edge features of the sender's *parents* and thus describes how the sender connects to the nodes of the previous shell.

The same-shell edge feature sum. The same-shell edge feature sum $\mathbf{ses}^d_{ij}$ adds up the edge features of the sender's neighbors in its own shell, and thus describes how the sender connects to the nodes of the same shell.

The parent ledger state. The message carries one more quantity besides the two edge sums: the *parent ledger state* $\mathbf{pls}^d_{ij}$, which accumulates, over the parents of the sender, what each of them has accumulated so far. To keep those contents available, each layer maintains a per-node *two-channel ledger* $\mathbf{Z}$, initialized as $\mathbf{Z}^0 = [\, \mathbf{H}^{(l)} \,\|\, \mathbf{0} \,]$, where $\mathbf{H}^{(l)}$ is the layer input, $\mathbf{H}^{(l)}[i] = \mathbf{x}^{(l)}_i$, and $\mathbf{0}$ is a zero block of the same width. The layer processes the shells $d{=}1, \ldots, D_{\max}$; after shell d is processed,

$$\mathbf{E}^d_{\text{acc}}[i] = \sum_{j\,:\,(i,j)\in\mathcal{I}_d} \mathbf{e}^d_{ij}, \qquad \mathbf{H}^d[i] = \mathbf{h}^d_i, \tag{5}$$

$$\mathbf{Z}^d = \mathbf{Z}^{d-1} + \big[\, \mathbf{H}^d \,\big\|\, \mathbf{E}^d_{\text{acc}} \,\big], \tag{6}$$

$$\mathbf{pls}^d_{ij} = \sum\nolimits_p \mathbf{1}\{\mathbf{D}[i, p] = d - 1\}\, \mathbf{1}\{\mathbf{A}[p, j] = 1\}\, \mathbf{Z}^{d-1}[p], \tag{7}$$

where $\mathbf{H}^d$ collects the shell representations, one entry per receiver, and the per-receiver entry $\mathbf{h}^d_i$ is defined below in Eq. (11). Both ledger updates are commutative scatter-adds, so they do not depend on the order in which the node pairs are processed, and the ledger is indexed by the node, so it is permutation-equivariant, $\mathbf{Z}'^d[\pi(i)] = \mathbf{Z}^d[i]$ (Appendix B.3). The ledger is advanced *serially, from the innermost shell outward*: when shell d is processed it holds $\mathbf{Z}^{d-1}$, already carrying the first $d{-}1$ shells, and the parents of the sender j on its shortest path to the receiver are exactly the nodes one

step closer to i, i.e. $\mathbf{D}[i,p]{=}d{-}1$ with $\mathbf{A}[p,j]{=}1$. At $d{=}1$ the sum reduces to the receiver's own ledger state on the pairs $\mathcal{I}_1$ where it is used. The reduction is exact because $\{\, p : \mathbf{D}[i,p]{=}0 \,\} = \{i\}$ within the molecule and $\mathbf{D}[i,j]{=}1$ implies $\mathbf{A}[i,j]{=}1$.

The environment vector.

$$\mathbf{env}_{ij}^{d} \;=\; \Big[\, \mathbf{e}_{ij}^{d} \,\Big\|\, \mathbf{pls}_{ij}^{d} \,\Big], \tag{8}$$

where $\mathbf{e}_{ij}^{d}$ is defined in Eq. (4) and $\mathbf{pls}_{ij}^{d}$ is defined in Eq. (7); the environment vector is their concatenation, so its width is set by this design alone.

3.3 Gate-normalized aggregation within a shell

The edge features are precomputed once, before the layer loop, and stay fixed across layers, so the parent-edge and same-shell edge feature sums $\mathbf{pes}_{ij}^{d}, \mathbf{ses}_{ij}^{d}$ of Eqs. (2)–(3) are the same at every layer. Both the message and the gate consume the environment vector of Eq. (8) as a single input block:

$$\mathbf{m}_{ij}^{d} = \mathbf{W}_{\text{msg}}^{d} \,[\, \mathbf{x}_{j}^{(l)} \,\|\, \mathbf{env}_{ij}^{d} \,], \tag{9}$$

$$\mathbf{g}_{ij}^{d} = \sigma\Big(\mathbf{W}_{\text{gate}}^{d} \,[\, \mathbf{x}_{i}^{(l)} \,\|\, \mathbf{x}_{j}^{(l)} \,\|\, \mathbf{env}_{ij}^{d} \,]\Big), \tag{10}$$

where $\mathbf{W}_{\text{msg}}^{d}$ and $\mathbf{W}_{\text{gate}}^{d}$ ($d{=}1,\dots,D_{\max}$) are linear maps that take their input to the hidden width H and σ is the elementwise logistic sigmoid. Every layer holds its own depth-indexed maps and its own depth-specific LayerNorm; for readability we drop the layer index from these symbols. The gate differs from the message input only by the receiver's features $\mathbf{x}_{i}^{(l)}$: the message carries *what* the sender is, and the gate modulates it by *who receives it*.

The shell representation. Each node pair's message is weighted by a per-pair sigmoid gate $\mathbf{g}_{ij}^{d}$ applied before any aggregation, accumulated within its shell and normalized by the sum of those gates, followed by a depth-specific LayerNorm, SiLU and dropout (Algorithm 1, SHELLGATEDLAYER):

$$\mathbf{h}_{i}^{d} = \mathrm{Dropout}\left(\mathrm{SiLU}\left(\mathrm{LN}\left(\frac{\sum_{j\,:\,(i,j)\in\mathcal{I}_d} \mathbf{g}_{ij}^{d} \odot \mathbf{m}_{ij}^{d}}{\left(\sum_{j\,:\,(i,j)\in\mathcal{I}_d} \mathbf{g}_{ij}^{d}\right) + \epsilon}\right)\right)\right), \tag{11}$$

where $\odot$ is the elementwise product, ϵ prevents division by zero for an empty shell, and both sums run over the shell-d pairs of receiver i, so the gate weights and normalizes each shell. This is the per-receiver definition of the shell representation $\mathbf{h}_{i}^{d}$.

3.4 Updating the node representation

A layer processes the shells one by one, each with its own parameters and its own LayerNorm, SiLU and dropout, so the node representation is updated only after the whole depth-wise sweep: the $D_{\max}$ shell representations are then concatenated with the receiver's feature and projected back to width H (Algorithm 1, SHELLGATEDLAYER):

$$\mathbf{x}_{i}^{(l+1)} = \mathrm{Dropout}\big(\mathrm{SiLU}\big(\mathrm{LN}\big(\mathbf{W}_{\text{comb}}^{(l)} \,[\, \mathbf{x}_{i}^{(l)} \,\|\, \mathbf{h}_{i}^{1} \,\|\, \cdots \,\|\, \mathbf{h}_{i}^{D_{\max}} \,]\big)\big)\big) + \mathbf{x}_{i}^{(l)} \tag{12}$$

Here $\mathbf{W}_{\text{comb}}^{(l)}$ projects the concatenation back to the hidden width H. The trailing $\mathbf{x}_{i}^{(l)}$ is a layer-wise residual. Concatenation keeps the per-depth shell outputs separate and lets $\mathbf{W}_{\text{comb}}^{(l)}$ combine them (Appendix B.5). The concatenation acts per node, and its permutation equivariance makes it a valid way to combine the shells (Proposition 3). The overall architecture is shown in Figure 2, and the settings of the readout and of the input embedding are listed in Table 5; the complexity analysis is in Appendix B.9.

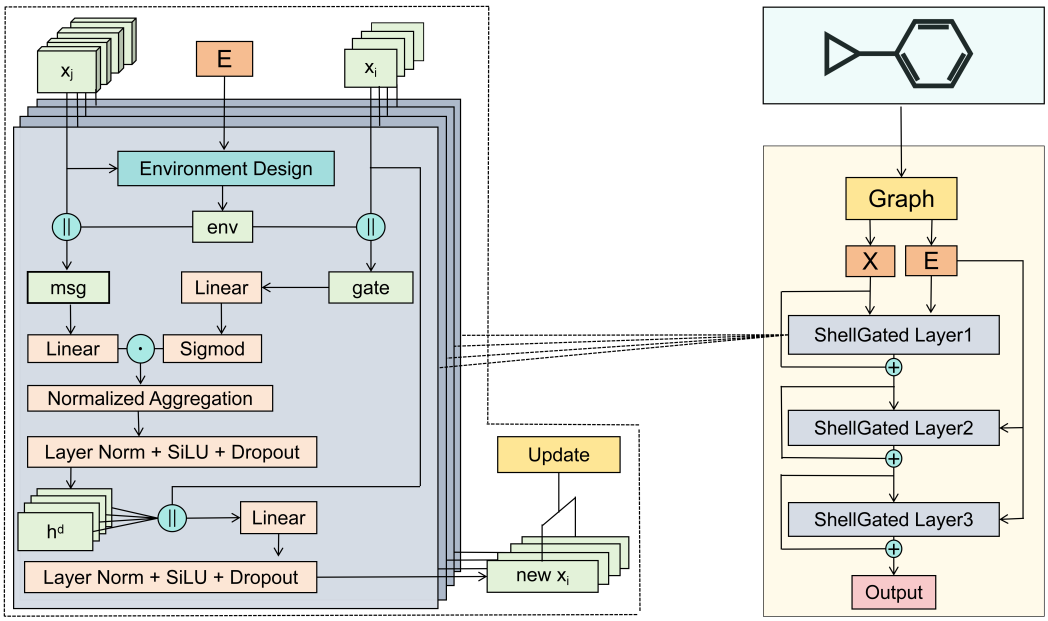


Figure 2: The SGGNN used in this work. **Right:** overall architecture: the input graph $(\mathbf{X}, \mathbf{E})$ is embedded by an MLP, passed through L residual SHELLGATEDLAYER layers and pooled by an MLP head. **Left:** one SHELLGATEDLAYER layer for a receiver i: the design of the sender's environment builds the environment vector $\mathbf{env}_{ij}^{d}$ (Eq. (8)), which feeds the message and the gate; the gated messages are aggregated within the shell into $\mathbf{h}_i^d$, and the per-depth outputs are concatenated for the update of $\mathbf{x}_i$ (Eq. (12)). The depths run serially; within one depth, the nodes are updated in parallel.

4 EXPERIMENTS

Our experiments answer the following three questions:

- **Performance.** How does SGGNN perform on molecular property prediction?
- **Range of one layer.** How does the performance change as $D_{\max}$ increases from 1 (approximately 1-hop message passing) to deeper shells?
- **Mechanism.** Do the design of the sender's environment and the concatenation of the shell representations improve performance and if so by how much?

4.1 EXPERIMENTAL SETUP

Datasets. QM9 (Ramakrishnan et al., 2014) is a standard quantum-chemical benchmark: **133,885** molecules of at most nine heavy atoms (C, N, O, F), of which we use **128,910** after cleaning. QM40 (Madushanka et al., 2024) extends the same protocol to a larger, drug-like space: **162,954** ZINC molecules (Sterling & Irwin, 2015) of up to 37 heavy atoms, adding sulfur and chlorine. Both compute every electronic property with DFT at the same level of theory, B3LYP/6-31G(2df,p). PCQM4Mv2 (Hu et al., 2021) is the largest public 2D-graph benchmark: 3,746,619 molecules in total, of which the official training split holds 3,378,606; the target is the DFT HOMO-LUMO gap in eV.

Baselines. We compare against two families of baselines. The first is message passing methods: the canonical representative is the plain GCN (Kipf & Welling, 2017) implemented with PyG; virtual-node message passing (VN-GCN, a GCN carrying the virtual node of Gilmer et al., 2017; Pham et al., 2017); GCN+, GIN+ and GatedGCN+ strengthened with classical message passing designs (Luo et al., 2025); directed-edge message passing Chemprop

(Yang et al., 2019; Heid et al., 2024); and the multi-hop methods MixHop (Abu-El-Haija et al., 2019), AIMNet-X2D (Nandakumar, 2025) and DuAxMPNN (Cui & Sun, 2026). The second is graph transformers and their hybrids, here GraphGPS (Rampášek et al., 2022), which combines message passing with global attention; we use it for a supplementary comparison across method classes. Each baseline is used in its official implementation. They are evaluated on QM9 and QM40; on PCQM4Mv2 we directly cite the baseline results reported on the official leaderboard.

Common configuration. On QM9 and QM40 all models share the same split and the same node and edge features, at a common hidden width of 256; PCQM4Mv2 uses the official split of that benchmark, with the input featurization of Appendix C.2. On QM9 and QM40, SGGNN is trained with the mean of six standardized L1 losses. $D_{\max}$ is selected on the validation split and equals 4 on both small datasets; PCQM4Mv2 keeps the same value. On QM9 and QM40 every model and every configuration is trained with three independent runs, reported as their mean with the sample standard deviation and with test values taken from the checkpoint of lowest validation std-MIX (defined in Eq. (13)).

Metric (QM9 and QM40). The evaluation metric *std-MIX* averages the per-property test mean absolute error (MAE) over the six properties, where each per-property MAE is normalized by the training-set standard deviation of that property:

$$\text{std-MIX} \;=\; \frac{1}{6}\sum_{p=1}^{6}\frac{\text{MAE}_p}{\sigma_p}, \qquad \text{MAE}_p \;=\; \frac{1}{|\mathcal{T}|}\sum_{i\in\mathcal{T}}\big|\hat{y}_i^{(p)} - y_i^{(p)}\big|, \tag{13}$$

where $\mathcal{T}$ is the test set, $\hat{y}_i^{(p)}$ is the model prediction of property p for molecule i in the raw unit of that property, and σ_p is the standard deviation of property p computed on the training set. Normalizing by σ_p makes the six properties comparable despite their very different scales. The values of σ_p are given in Table 9.

4.2 Performance results

Tables 1 and 2 report the per-property test MAE and the std-MIX (last column, Eq. (13)) for QM9 and QM40, respectively. The six target properties are electronic properties computed by DFT: the dipole moment μ; the isotropic static polarizability α; the frontier-orbital energies ($\varepsilon_{\text{HOMO}}$ and $\varepsilon_{\text{LUMO}}$), i.e. the highest occupied and lowest unoccupied molecular orbital energies, and their gap $\Delta\varepsilon$; and $\langle R^2\rangle$, the electronic spatial extent.

Table 1: QM9 main comparison: per-property test MAE and the std-MIX (last column); each entry is the mean over three independent runs with the sample std in parentheses (per-property MAE in each property's own unit, with $\varepsilon_{\text{HOMO}}/\varepsilon_{\text{LUMO}}/\Delta\varepsilon$ in meV; only the std-MIX column is unitless, lower is better, Eq. (13)). SGGNN uses $D_{\max}$=4. Rows are sorted by std-MIX, with SGGNN listed last. Within each column, blue/green/red bold mark the best, second-best and third-best entry (ties share a color).

Method	Year	μ MAE ↓	α MAE ↓	$\varepsilon_{\text{HOMO}}$ MAE ↓	$\varepsilon_{\text{LUMO}}$ MAE ↓	$\Delta\varepsilon$ MAE ↓	$\langle R^2\rangle$ MAE ↓	std-MIX MAE ↓
GCN	2017	0.4813(0.0019)	1.2950(0.0168)	106.73(0.18)	114.18(0.25)	149.65(0.60)	48.0873(0.5125)	0.1723(0.0010)
VN-GCN	2017	0.4648(0.0009)	0.4945(0.0024)	107.05(0.57)	109.83(0.53)	145.67(0.22)	31.8491(0.2877)	0.1436(0.0004)
GCN+	2025	0.4436(0.0011)	0.7825(0.0196)	95.03(0.63)	102.62(0.13)	131.97(0.41)	33.7393(0.4789)	0.1422(0.0006)
MixHop	2019	0.4350(0.0017)	0.7396(0.0184)	94.66(0.67)	101.83(0.13)	131.37(0.49)	32.1699(0.3642)	0.1391(0.0009)
Chemprop	2024	0.4232(0.0038)	0.5865(0.0052)	98.56(0.19)	114.14(1.49)	141.19(1.19)	31.2128(0.3566)	0.1381(0.0006)
AIMNet-X2D	2025	0.4178(0.0066)	0.4831(0.0175)	88.72(4.05)	103.85(2.43)	126.83(2.27)	25.7824(2.0703)	0.1263(0.0035)
GIN+	2025	0.3733(0.0010)	0.5406(0.0052)	77.61(0.31)	84.77(0.64)	107.47(0.09)	25.1704(0.0840)	0.1140(0.0002)
GraphGPS	2022	0.3549(0.0007)	0.4272(0.0109)	75.03(0.34)	81.20(0.58)	102.46(0.40)	22.6760(0.2682)	0.1064(0.0004)
GatedGCN+	2025	**0.3514(0.0011)**	**0.4008(0.0014)**	**74.21(0.23)**	**78.97(0.21)**	**101.11(0.40)**	**20.9589(0.0435)**	**0.1037(0.0002)**
DuAxMPNN	2026	**0.3439(0.0022)**	**0.2618(0.0020)**	**66.09(0.05)**	**70.77(1.02)**	**90.21(0.50)**	**17.5737(0.0662)**	**0.0933(0.0005)**
SGGNN	Ours	**0.3016(0.0012)**	**0.2434(0.0017)**	**57.52(0.31)**	**60.97(0.24)**	**77.51(0.39)**	**15.5938(0.1420)**	**0.0817(0.0004)**

Table 2: QM40 main comparison; conventions as in Table 1 ($\varepsilon_{\mathrm{HOMO}}$/$\varepsilon_{\mathrm{LUMO}}$/$\Delta\varepsilon$ in meV, the other properties in their raw unit). SGGNN uses $D_{\max}$=4.

Method	Year	μ MAE ↓	α MAE ↓	$\varepsilon_{\mathrm{HOMO}}$ MAE ↓	$\varepsilon_{\mathrm{LUMO}}$ MAE ↓	$\Delta\varepsilon$ MAE ↓	$\langle R^2 \rangle$ MAE ↓	std-MIX MAE ↓
VN-GCN	2017	1.2779(0.0026)	**2.9972(0.0456)**	145.06(0.51)	142.70(0.16)	188.42(0.92)	1285.9548(4.8510)	0.2354(0.0005)
GCN	2017	1.2067(0.0006)	7.1139(0.0177)	125.53(0.17)	120.59(0.24)	157.64(0.32)	1418.8732(1.3065)	0.2265(0.0001)
Chemprop	2024	1.2050(0.0043)	3.9126(0.1394)	126.84(0.35)	125.79(1.19)	162.78(0.62)	1213.5831(12.8197)	0.2171(0.0008)
AIMNet-X2D	2025	1.2092(0.0095)	4.2566(0.5374)	124.92(2.92)	123.00(2.90)	156.85(2.51)	1166.1403(15.3549)	0.2150(0.0034)
MixHop	2019	1.1894(0.0022)	5.3064(0.0206)	120.13(0.30)	114.70(0.25)	149.15(0.47)	1257.9940(3.2972)	0.2134(0.0001)
GCN+	2025	1.1882(0.0009)	4.7494(0.0207)	120.26(0.26)	114.20(0.45)	148.71(0.88)	1238.9414(2.1977)	0.2115(0.0002)
DuAxMPNN	2026	1.2163(0.0109)	**3.2622(0.4992)**	123.17(4.35)	116.84(5.51)	152.67(7.04)	**1145.9204(47.7835)**	0.2104(0.0061)
GraphGPS	2022	1.2047(0.0049)	4.2283(0.0571)	116.95(0.18)	**107.97(0.54)**	**141.11(0.66)**	1183.6055(7.3041)	0.2069(0.0005)
GIN+	2025	**1.1800(0.0011)**	3.8067(0.0042)	**116.26(0.37)**	108.66(0.33)	141.91(0.24)	1154.1230(2.7295)	**0.2035(0.0001)**
GatedGCN+	2025	**1.1774(0.0016)**	3.5199(0.0134)	**115.15(0.36)**	**106.88(0.39)**	**140.15(0.29)**	**1130.4186(3.1925)**	**0.2012(0.0004)**
SGGNN	Ours	**1.1615(0.0011)**	**2.4590(0.0071)**	**111.59(0.18)**	**102.57(0.21)**	**134.52(0.13)**	**1063.5413(1.3831)**	**0.1933(0.00003)**

Large-scale comparison on PCQM4Mv2. We also train SGGNN on PCQM4Mv2, with the same $D_{\max}$=4, L=3 and H=256 as our other runs and 50 training epochs; the full hyper-parameters are in Appendix C (Table 5).

Table 3 compares SGGNN with methods of the same kind; the baseline numbers are the validation column of the public leaderboard.

Table 3: PCQM4Mv2 validation MAE (eV, lower is better).

Method	MAE ↓
GCN (Kipf & Welling, 2017)	0.1379
GIN (Xu et al., 2019)	0.1195
GCN-virtual (Hu et al., 2021)	0.1153
GIN-virtual (Hu et al., 2021)	0.1083
SGGNN	**0.0957**

4.3 Range sweep

The depth $D_{\max}$ controls how many shells are aggregated. With $D_{\max}$=1 only the first-hop shell is used and all multi-hop shells are inactive; increasing $D_{\max}$ gradually restores deeper shells. In this limit SGGNN approximates a standard 1-hop baseline (Appendix B.8). Figure 3 (QM9 left, QM40 right) and the depth tables in Appendix D.2 report the per-property test MAE (same units as Tables 1 and 2, with $\varepsilon_{\mathrm{HOMO}}$/$\varepsilon_{\mathrm{LUMO}}$/$\Delta\varepsilon$ in meV) and the std-MIX.

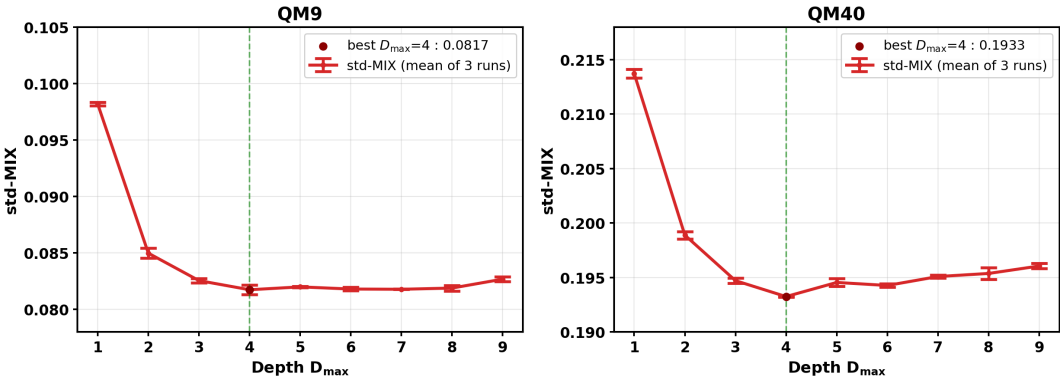


Figure 3: Range sweep $D_{\max}$=1–9 on QM9 (left) and QM40 (right): test std-MIX, mean over three runs, with bars showing the run-to-run sample std.

On both datasets the error drops sharply as $D_{\max}$ grows, reaches its minimum near $D_{\max}$=4 and then flattens (Fig. 3, Tables 7 and 8; per-property curves in Figs. 10 and 11). A further observation from the same sweep is that the properties respond to depth very differently, which may reflect how much each of them depends on long-range structure; the per-property curves are discussed in Appendix D.2.

4.4 ABLATION STUDY

The environment vector $\mathbf{env}_{ij}^{d}$ of a node pair (i, j) describes the sender's local environment in the receiver's shell coordinates (Eq. (8)); it is the concatenation of $\mathbf{e}_{ij}^{d}$ and $\mathbf{pls}_{ij}^{d}$. An ablation zero-fills the entries of the target term instead of deleting it.

The sender's environment. With both terms zeroed the message no longer carries the two edge feature sums or the ledger state, and only the gate-normalized aggregation within the shells and the receiver's own representation $\mathbf{x}_i^{(l)}$ remain. When the terms are added back one at a time, $\mathbf{pls}_{ij}^{d}$ gives the larger reduction and $\mathbf{e}_{ij}^{d}$ the smaller one; the two separate reductions add up to more than the one of the full model, so the two terms overlap. The gain of $\mathbf{e}_{ij}^{d}$ is notable in itself: it is the simpler of the two, a plain sum of the sender's edge features with no accumulation across shells, and it still brings a substantial reduction.

The node update. Replacing the concatenation by a sum before the projection costs performance as well (last two rows of Table 4; per-property values in Table 6).

Table 4: Ablation study at $D_{\max}=4$ on QM9: test std-MIX (mean over three runs, sample std in parentheses; lower is better, Eq. (13)), its absolute and relative reduction vs. the first row (both environment terms zeroed, per-depth parts summed); abs. Δ is the drop in std-MIX and rel. Δ the same drop in %, larger is better. ✓ denotes that the column's term is present and $\times$ that it is zero-filled; "cat"/"sum" denote the per-depth combination before the projection (Eq. (12)); the first column is $\mathbf{e}_{ij}^{d}$ (Eq. (4)) and the second is $\mathbf{pls}_{ij}^{d}$ (Eq. (7)), i.e. the two terms of Eq. (8). Receiver/sender features $\mathbf{x}_i^{(l)}, \mathbf{x}_j^{(l)}$ are always kept.

Dataset	$\mathbf{e}_{ij}^{d}$	$\mathbf{pls}_{ij}^{d}$	**comb.**	**std-MIX**	**reduction vs. row 1 (sum)**	
					abs. Δ	**rel. Δ (%)**
QM9	$\times$	$\times$	sum	$0.0939(0.0002)$	–	–
QM9	$\times$	$\times$	cat	$0.0899(0.0003)$	$0.0040(0.0005)$	$4.2(0.6)$
QM9	$\times$	✓	cat	$0.0824(0.0003)$	$0.0115(0.0001)$	$12.2(0.1)$
QM9	✓	$\times$	cat	$0.0862(0.0003)$	$0.0077(0.0005)$	$8.2(0.5)$
QM9	✓	✓	sum	$0.0854(0.0002)$	$0.0085(0.0004)$	$9.1(0.4)$
QM9	✓	✓	cat	$0.0816(0.0001)$	$0.0123(0.0003)$	$13.1(0.3)$

Setting a term to zero does not remove it from the network: the block stays and only its output is zero, so the parameter count equals that of the full model and the rows differ only in whether that information is available. Both Δ columns are computed from the unrounded per-run values, so an entry can differ from the difference of the two displayed means by one unit in the last digit. The full-model row comes from a run set independent of the performance experiments.

5 CONCLUSION

In this paper, we introduce SGGNN, a long-range message-passing architecture that extends gated message passing from a single edge to a triple of the sender, the sender's environment and the receiver. The architecture divides the receiver's neighbors into shells and lets the gate act on each shell, so one layer covers several hops. It decouples the reach of a layer from the number of layers, and it thereby turns long-range modeling into a question of how each shell is described and aggregated. This view keeps the design of the sender's environment open: that environment enters only through the environment vector, so other descriptions can replace the two sum vectors used here without changing the rest of the layer. Empirically, SGGNN attains the best performance among the strong baselines on QM9 and QM40, and it outperforms the baselines of the same kind on PCQM4Mv2 without pretraining, attention or an ensemble. Limitations of this work and directions for further research are discussed in Appendix A.

REPRODUCIBILITY STATEMENT

The anonymous code repository given in the abstract contains the training scripts of all the runs, with the hyperparameters and the settings they use, together with the scripts that build the node and edge features and the labels from the raw CSV files. The datasets, their splits and the label construction are described in Appendix C.1, the input features and the preprocessing in Appendix C.2, the hyperparameters and the training protocol in Appendix C.4 and the computing environment in Appendix C.3. The complete forward pass at the level of individual operations is given in Algorithm 1.

ETHICS STATEMENT

The authors have read and commit to adhering to the Code of Ethics of ICLR. This work studies molecular property prediction on three public datasets (QM9, QM40 and PCQM4Mv2), and it involves no human subjects, no personally identifiable information and no sensitive data. The predictions of the models studied here approximate quantum-chemical calculations, and a model of this kind should not replace first-principles results in decisions with safety or regulatory consequences. Beyond this, we foresee no direct ethical concerns.

REFERENCES

Sami Abu-El-Haija, Bryan Perozzi, Amol Kapoor, Nazanin Alipourfard, Kristina Lerman, Hrayr Harutyunyan, Greg Ver Steeg, and Aram Galstyan. MixHop: Higher-order graph convolutional architectures via sparsified neighborhood mixing. In *International Conference on Machine Learning (ICML)*, pp. 21–29, 2019.

Uri Alon and Eran Yahav. On the bottleneck of graph neural networks and its practical implications. In *International Conference on Learning Representations (ICLR)*, 2021.

Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001. doi: 10.1023/A:1010933404324.

Xavier Bresson and Thomas Laurent. Residual Gated Graph ConvNets. arXiv preprint arXiv:1711.07553, 2017.

Tianqi Chen and Carlos Guestrin. XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pp. 785–794, 2016. doi: 10.1145/2939672.2939785.

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

Xuefeng Cui and Shiwei Sun. Dual-axis message passing on molecular graphs: Layerwise multi-hop convolution meets depthwise dense mixing. In *Bioinformatics Research and Applications (ISBRA 2026), Lecture Notes in Computer Science*, pp. 54–68. Springer, 2026. doi: 10.1007/978-981-92-3716-6_5. Released implementation: `https://github.com/xfcui/DuAxMPNN`.

Johannes Gasteiger, Janek Groß, and Stephan Günnemann. Directional message passing for molecular graphs. In *International Conference on Learning Representations (ICLR)*, 2020.

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 (ICML)*, pp. 1263–1272, 2017.

William L. Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 1025–1035, 2017.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. doi: 10.1109/CVPR.2016.90.

Esther Heid, Kevin P. Greenman, Yunsie Chung, Shih-Cheng Li, David E. Graff, Florence H. Vermeire, Haoyang Wu, William H. Green, and Charles J. McGill. Chemprop: A machine learning package for chemical property prediction. *Journal of Chemical Information and Modeling*, 64(1): 9–17, 2024. doi: 10.1021/acs.jcim.3c01250.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8): 1735–1780, 1997. doi: 10.1162/neco.1997.9.8.1735.

Weihua Hu, Matthias Fey, Hongyu Ren, Maho Nakata, Yuxiao Dong, and Jure Leskovec. OGB-LSC: A large-scale challenge for machine learning on graphs. In *NeurIPS Datasets and Benchmarks Track*, 2021.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning (ICML)*, pp. 448–456, 2015.

Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations (ICLR)*, 2017.

Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard Zemel. Gated graph sequence neural networks. In *International Conference on Learning Representations (ICLR)*, pp. 146–157, 2016.

Yuankai Luo, Lei Shi, and Xiao-Ming Wu. Can classic gnns be strong baselines for graph-level tasks? simple architectures meet excellence. In *International Conference on Machine Learning (ICML)*, pp. 41290–41310, 2025.

Ayesh Madushanka, Renaldo T. Moura, and Elfi Kraka. QM40, realistic quantum mechanical dataset for machine learning in molecular science. *Scientific Data*, 11:1376, 2024. doi: 10.1038/s41597-024-04206-y.

Gaspard Michel, Giannis Nikolentzos, Johannes F. Lutzeyer, and Michalis Vazirgiannis. Path neural networks: Expressive and accurate graph neural networks. In *International Conference on Machine Learning (ICML)*, pp. 24737–24755, 2023.

Rohit Nandakumar. AIMNET-X2D: Universal and scalable framework for chemical property transfer learning. Thesis, Carnegie Mellon University, KiltHub, 2025. doi: `https://doi.org/10.1184/R1/29120381`; Code: `https://github.com/isayevlab/aimnet-x2d` (v1.0.0).

Giannis Nikolentzos, George Dasoulas, and Michalis Vazirgiannis. k-hop graph neural networks. *Neural Networks*, 130:195–205, 2020. doi: 10.1016/j.neunet.2020.07.008.

Trang Pham, Truyen Tran, Hoa Dam, and Svetha Venkatesh. Graph Classification via Deep Learning with Virtual Nodes. arXiv preprint arXiv:1708.04357, 2017.

Raghunathan Ramakrishnan, Pavlo O. Dral, Matthias Rupp, and O. Anatole von Lilienfeld. Quantum chemistry structures and properties of 134 kilo molecules. *Scientific Data*, 1:140022, 2014. doi: 10.1038/sdata.2014.22.

Ladislav Rampášek, Michael Galkin, Vijay Prakash Dwivedi, Anh Tuan Luu, Guy Wolf, and Dominique Beaini. Recipe for a general, powerful, scalable graph transformer. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

David Rogers and Mathew Hahn. Extended-connectivity fingerprints. *Journal of Chemical Information and Modeling*, 50(5):742–754, 2010. doi: 10.1021/ci100050t.

Kristof T. Schütt, Pieter-Jan Kindermans, Huziel E. Sauceda, Stefan Chmiela, Alexandre Tkatchenko, and Klaus-Robert Müller. SchNet: A continuous-filter convolutional neural network for modeling quantum interactions. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 30, pp. 992–1002, 2017.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014.

Teague Sterling and John J. Irwin. ZINC 15 – ligand discovery for everyone. *Journal of Chemical Information and Modeling*, 55(11):2324–2337, 2015. doi: 10.1021/acs.jcim.5b00559.

Zhaoping Xiong, Dingyan Wang, Xiaohong Liu, Feisheng Zhong, Xiaozhe Wan, Xutong Li, Zhaojun Li, Xiaomin Luo, Kaixian Chen, Hualiang Jiang, and Mingyue Zheng. Pushing the boundaries of molecular representation for drug discovery with the graph attention mechanism. *Journal of Medicinal Chemistry*, 63(16):8749–8760, 2020. doi: 10.1021/acs.jmedchem.9b00959.

Keyulu Xu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken-ichi Kawarabayashi, and Stefanie Jegelka. Representation learning on graphs with jumping knowledge networks. In *International Conference on Machine Learning (ICML)*, pp. 5449–5458, 2018.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *International Conference on Learning Representations (ICLR)*, 2019.

Kevin Yang, Kyle Swanson, Wengong Jin, Connor Coley, Philipp Eiden, Hua Gao, Angel Guzman-Perez, Timothy Hopper, Brian Kelley, Miriam Mathea, Andrew Palmer, Volker Settels, Tommi Jaakkola, Klavs Jensen, and Regina Barzilay. Analyzing learned molecular representations for property prediction. *Journal of Chemical Information and Modeling*, 59(8):3370–3388, 2019. doi: 10.1021/acs.jcim.9b00237.

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

Muhan Zhang and Pan Li. Nested graph neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

APPENDIX

A DISCUSSION

A.1 LIMITATIONS

Validated on molecular graph-level regression only. SGGNN is defined for any connected undirected graph with node and edge features, not only for molecular graphs. In this work we validate it only on graph-level regression of molecular electronic properties, on QM9, QM40 and PCQM4Mv2. Node-level tasks such as node classification, edge-level tasks such as link prediction, other graph families and larger scales are left for future evaluation.

The depth budget is fixed in advance. $D_{\max}$ is a model hyper-parameter, selected on the validation split and set to 4 for all three datasets (Section 4.3); the layer does not adapt how far it reaches to the molecule at hand.

Information is compressed into a fixed width. One layer aggregates the messages from all shells into the receiver's representation of width H, so whatever arrives from a large neighbourhood is compressed into a fixed number of channels, the bottleneck known as over-squashing (Alon & Yahav, 2021). On both datasets the aggregate std-MIX of the depth sweep stops improving beyond $D_{\max}{=}4$ (Section 4.3); whether this bottleneck contributes to that is not measured here.

Topological distances only. The shells count edges, so the model carries no conformation and no spatial geometry; properties that depend on the actual geometry, on non-bonded contacts or on a conformational ensemble lie outside its inputs.

The layer components are the standard ones. The normalization, the activation and the dropout inside the layer follow the standard configuration of the baselines (Appendix C.4), and no ablation here separates them from the shell structure they act on.

The two edge sums are computed densely. Both are implemented densely: each of them is contracted over all N candidates for every node pair, so the two sums alone cost $O(D_{\max}N^3C)$ (Appendix B.9).

A.2 FUTURE DIRECTIONS

The first item concerns the setting the model applies to; the others take three parts of the model in turn, the division into shells, the design of the sender's environment and the gate-normalized aggregation within a shell, and then its implementation.

- **Settings beyond molecular graphs.** The model applies to a wider setting than molecules: it requires only an undirected connected graph whose nodes and edges carry features. Our experiments cover discrete features only and we have not validated the model on continuous features. Molecular graphs are one instance, with atom types on the nodes and bond types on the edges; materials, proteins and much of physics give the same description. The main work in moving to such a task is to describe the sender's neighbourhood in the same generic way and to choose a readout suited to it.
- **The division into shells.** The depth budget is fixed before training (Section 4.3), so a layer does not adapt its reach to the graph at hand. Letting the depth adapt, or sharing parameters across depths, is a natural direction.
- **The design of the sender's environment.** The inputs that condition a message are currently permutation-invariant statistics over the sender's neighbourhood. Other permutation-invariant operators, or additional information such as a three-dimensional geometry, could take their place without changing the rest of the layer.
- **Aggregation and normalization within a shell.** The aggregation and the normalization inside a layer follow common practice, for instance pooling (Hamilton et al., 2017) or several aggregations with degree scaling (Corso et al., 2020); whether some other choice

suits this layer better has not been tested here. A related question is the fixed number of channels a shell is aggregated into (Appendix A.1): letting that number grow with the shell, or attending over the nodes of a shell instead of averaging them, are two natural directions.

- **Sparse evaluation of the two sums.** Both sums are evaluated densely at present: each of them is contracted over all N candidates for every node pair. Contracting only over the edges that contribute would reduce this cost from cubic in N to linear in the number of contributing edge terms.

B Methodological Analysis

This appendix gives a methodological analysis of our model SGGNN, one layer step by step and then the permutation equivariance of the whole message-passing step, and closes with a comparison of what the input of one layer sees with the 1-WL test, the boundary case $D_{\max}=1$, the complexity analysis and the complete forward pass.

B.1 Distinction from other model families

This subsection compares our model with two kinds of related methods, three-dimensional models, and graph transformers together with their hybrids.

Three-dimensional models. SchNet (Schütt et al., 2017) and DimeNet (Gasteiger et al., 2020) enter through the geometry: they take the interatomic distances, and DimeNet the bond angles as well. SGGNN enters through a two-dimensional graph, so the same molecules are described without any spatial coordinate.

Graph transformers and their hybrids. Graph transformers flatten the graph into a set of tokens and let pairs of tokens interact by attention, with the graph structure encoded into the model (Ying et al., 2021). A hybrid such as GraphGPS (Rampášek et al., 2022) combines message passing with a graph transformer. Our design still belongs to traditional message passing: the main framework is unchanged, namely generating a message, aggregating the messages and updating the node representation.

The design has three ingredients. **Long-range message-passing gating**: one layer processes every shell, from the nearest to the farthest, so the reach of a layer is set by $D_{\max}$ and not by the number of layers L. **The environment**: the message and the gate both carry the sender's environment, namely the sender's connections to the nodes of the previous shell and of its own shell together with what its parents have accumulated; the topology between the receiver and the sender is therefore part of the input. **The node update**: the per-depth representations are concatenated rather than summed, which keeps more information.

B.2 Shelling by the BFS distance

This subsection states the shelling and proves that it is permutation-equivariant.

The method derives the BFS distance matrix $\mathbf{D}$ from the edges, so $\mathbf{D}$ is a function of the edges alone; in our data the edge features are non-negative, symmetric and zero on pairs that are not connected, the diagonal included, so Eq. (15) returns the adjacency matrix of G; throughout the appendix G is connected, so every distance is finite. For a depth $d \geq 1$ the method operates on the depth-d pairs $\mathcal{I}_d = \{(i,j) : \mathbf{D}[i,j] = d\}$; the nodes j of these pairs form the receiver's d-th shell.

Permutation equivariance of the shelling. Renumbering the nodes only renames the adjacency and the distance matrix, and the depth-d pairs are renumbered along with the nodes.

Proposition 1 (Permutation equivariance of the shelling)**.** *Let π renumber the nodes of G into $G' = \pi(G)$. Then*

$$\mathbf{D}'[\pi(i),\pi(j)] = \mathbf{D}[i,j], \qquad \mathbf{A}'[\pi(i),\pi(j)] = \mathbf{A}[i,j], \qquad \mathcal{I}'_d = \{(\pi(i),\pi(j)) : (i,j) \in \mathcal{I}_d\}. \tag{14}$$

Proof. The adjacency is computed from the edge features by the rule

$$\mathbf{A}[i,j] = \mathbf{1}\Big\{ \sum_{c} \mathbf{E}[i,j,c] > 0 \Big\} \cdot \mathbf{1}\{i \neq j\}, \tag{15}$$

which singles out no index. The renumbering carries the edge features along, $\mathbf{E}'[\pi(i),\pi(j)] = \mathbf{E}[i,j]$, so the same rule gives $\mathbf{A}'[\pi(i),\pi(j)] = \mathbf{A}[i,j]$. Breadth-first search derives the distances from $\mathbf{A}$ and singles out no index either, so $\mathbf{D}'[\pi(i),\pi(j)] = \mathbf{D}[i,j]$. The depth-$d$ pairs are exactly the pairs at distance d, hence $\mathcal{I}'_d = \{(\pi(i),\pi(j)) : (i,j) \in \mathcal{I}_d\}$. ■

B.3 The design of the sender's environment

This subsection gives the motivation for the sender's environment and the proof that its three sums, which are indexed by the node pair, are permutation-equivariant.

The relative position the shells give. The shelling gives every node pair (i,j) a relative position: the sender j falls in the d-th shell of the receiver i, with $d{=}\mathbf{D}[i,j]$, so the shells of one receiver become coordinates that can be told apart. The environment vector describes the sender's local environment in these coordinates, so the shelling makes that description more discriminative: a sender at distance d_1 from the receiver and a sender at distance d_2 ($d_1 \neq d_2$) get different environment descriptions, and nodes within one shell are distinguished from nodes in another. The gate conditions on the environment vector (its input differs from the message only by the receiver's feature $\mathbf{x}_i$), so the weighting it produces is set by the shell coordinates as well.

The two edge feature sums. Here we state which edges each of the two sums collects and which ring lengths the two of them separate; the edge features are zero on pairs that are not connected, so only the sender's edges contribute to either sum. An edge of the sender can only point to shell $d-1$, to its own shell d, or to shell $d+1$. The parent edge of the first kind enters $\mathbf{pes}^d_{ij}$, the same-shell edge of the second kind enters $\mathbf{ses}^d_{ij}$, and the edge of the third kind is collected one depth later, by the parent-edge feature sum of the next shell. The two sums therefore collect each of the sender's edges at exactly one depth within the depth budget, and they separate ring lengths: a neighbor of the sender in its own shell closes an odd cycle through the receiver, which $\mathbf{ses}^d_{ij}$ records; two parents in shell $d-1$ give two shortest paths instead of one, which $\mathbf{pes}^d_{ij}$ records, and the two paths form an even cycle when they share no node (Figure 4).

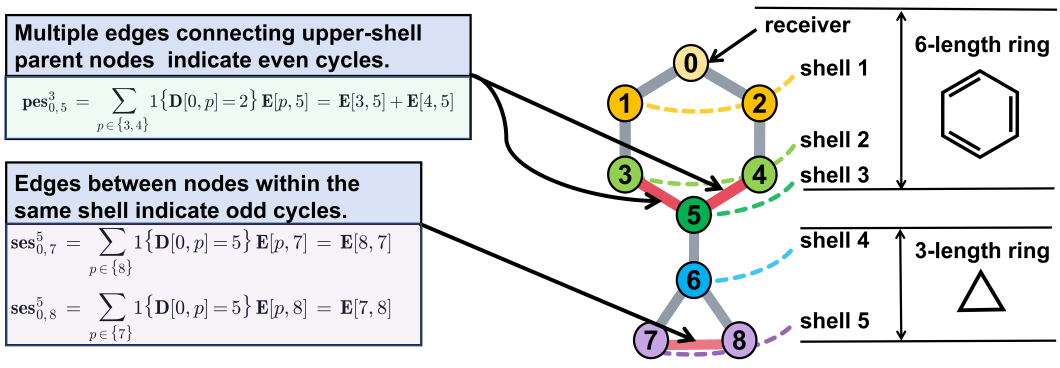


Figure 4: The parent-edge feature sum $\mathbf{pes}^d_{ij}$ and the same-shell edge feature sum $\mathbf{ses}^d_{ij}$ on a concrete BFS-shell example: the receiver is node 0, and the two shortest paths shown are internally disjoint.

The ledger and the parent ledger state. Here we give the two channels of the ledger and the coverage of the parent ledger state along the shortest paths. After shell d is processed, its shell representation and its edge-embedding sum are added to the ledger, one into the node channel and one into the edge channel:

$$\mathbf{Z}^d \;=\; \mathbf{Z}^0 + \sum_{d'=1}^{d} \big[\, \mathbf{H}^{d'} \,\big\|\, \mathbf{E}^{d'}_{\mathrm{acc}} \,\big].$$

Nothing is overwritten, so a pair (i, j) in an outer shell uses what the inner shells produced; the ledger is indexed by the node and inherits the permutation equivariance of the step (Proposition 5), $\mathbf{Z}'^{d}[\pi(i)] = \mathbf{Z}^{d}[i]$. The parent ledger state at depth d is the sum of the ledger states written by its parents, and each parent wrote its state under the same definition while it was itself a receiver. The state is therefore accumulated shell by shell along the shortest paths: write $F(d')$ for the largest number of hops from its own center that the ledger state at depth d' carries. It obeys

$$F(1) = 1, \qquad F(d') = (d' - 1) + F(d' - 1), \qquad F(d') = 1 + \frac{d'(d' - 1)}{2},$$

the first hop being the parent's own feature entering that shell. A depth-d message therefore covers $(d-1)+F(d-1) = F(d)$ hops, i.e. seven hops at $D_{\max}$=4, and the bound is tight (Appendix D.1). The environment therefore covers not one edge but the neighborhood the sender sits in. The two sums enter the environment through the edge embedding and the parent ledger state enters it directly (Eq. (8)).

Permutation equivariance of the sums and the ledger. The sums $\mathbf{pes}^{d}_{ij}$, $\mathbf{ses}^{d}_{ij}$ and $\mathbf{pls}^{d}_{ij}$ are indexed by the node pair and the ledger $\mathbf{Z}^{d}$ by the node. Relabeling the summation indices maps every term of the three sums to its counterpart, so they are permutation-equivariant:

$$\mathbf{pes}'^{d}_{\pi(i)\pi(j)} = \mathbf{pes}^{d}_{ij}, \qquad \mathbf{ses}'^{d}_{\pi(i)\pi(j)} = \mathbf{ses}^{d}_{ij}, \qquad \mathbf{pls}'^{d}_{\pi(i)\pi(j)} = \mathbf{pls}^{d}_{ij}, \qquad \mathbf{Z}'^{d}[\pi(i)] = \mathbf{Z}^{d}[i].$$

Each of the three sums in Eqs. (2), (3) and (7) runs over index sets, so relabeling the summation indices maps every term to the corresponding term of the sum at $(\pi(i), \pi(j))$: the nodes at distance $d-1$ (or d) from $\pi(i)$ are the images of those at distance $d-1$ (or d) from i, and the ones adjacent to $\pi(j)$ are the images of the ones adjacent to j. At d=1 the depth indicator selects only $p = i$, and the remaining indices have depth indicator 0 and contribute $\mathbf{0}$; it gives $\mathbf{E}[i, j]$ in $\mathbf{pes}^{1}_{ij}$ and $\mathbf{1}\{\mathbf{A}[i, j]{=}1\}\mathbf{Z}^{0}[i]$ in $\mathbf{pls}^{1}_{ij}$, and both match their counterparts at $p' = \pi(i)$ by the base case.

B.4 Gate-normalized aggregation within a shell

This subsection states the aggregation inside a shell and proves that it is permutation-equivariant, its output $\mathbf{h}^{d}_{i}$ being indexed by the receiver.

Gate-normalized aggregation within a shell. Each depth ends with the aggregation of Eq. (11): a per-pair sigmoid gate weights each message, the weighted messages are accumulated within their shell, and the sum is divided by the sum of the same gates. The gate weights and normalizes the shell.

Proposition 2 (Permutation equivariance of the aggregation)**.** *Let π carry G into $G' = \pi(G)$, with the features on the nodes and on the pairs carried along, and let both runs share the parameters, which are indexed by the layer and the depth alone. If the inputs of the message and of the gate correspond under the renumbering, then so do the message, the gate and the aggregation:*

$$\mathbf{m}'^{d}_{\pi(i)\pi(j)} = \mathbf{m}^{d}_{ij}, \qquad \mathbf{g}'^{d}_{\pi(i)\pi(j)} = \mathbf{g}^{d}_{ij}, \qquad \mathbf{h}'^{d}_{\pi(i)} = \mathbf{h}^{d}_{i}.$$

The equality holds exactly for the evaluation-time map; with dropout active a single mask is applied to corresponding entries in the two runs.

Proof. The message and the gate are produced by per-pair maps with shared parameters, so corresponding inputs give corresponding outputs. The shell $\mathcal{I}_d$ is renumbered with the nodes (Proposition 1), so the numerator of Eq. (11) at i and at $\pi(i)$ runs over corresponding pairs, and its terms match one by one; the denominator is the sum of the same gates over the same shell and matches in the same way, ϵ being a constant. LayerNorm and SiLU act per node with shared parameters and preserve the correspondence; if the shell of i at depth d is empty, then so is the shell of $\pi(i)$, and both sides give the same constant, so the equality still holds. ■

B.5 The node update

This subsection states the node update, and proves that it is permutation-equivariant and that the concatenation retains more information than a sum.

Concatenating the depths. Concatenation keeps each depth in its own block of channels and defers the combination to the learned $\mathbf{W}_{\text{comb}}^{(l)}$, which allocates one coefficient per depth; a sum before the projection would put the depths into one vector and leave the projection no way to weight them independently, which the ablation of Table 4 confirms. The block order is fixed by the depth index rather than by the node index, so Eq. (12) is still permutation-equivariant. The permutation equivariance of the message-passing step is in Appendix B.6. JK networks (Xu et al., 2018) also merge several representations rather than one, but along the layer axis: the blocks they merge are the outputs of successive propagation layers, which lets a node draw on a range of its accumulated 1-hop neighborhoods. The concatenation of Eq. (12) is along the depth axis inside one layer, where each block is one shell of the receiver.

Proposition 3 (Permutation equivariance of the node update)**.** *Let π carry G into $G' = \pi(G)$, with the features carried along, and let both runs share the parameters, which are indexed by the layer and the depth alone. If the layer input and the per-depth shell representations correspond under the renumbering, $\mathbf{x}'^{(l)}_{\pi(i)} = \mathbf{x}_i^{(l)}$ and $\mathbf{h}'^{d}_{\pi(i)} = \mathbf{h}_i^d$ for $d \le D_{\max}$, then*

$$\mathbf{x}'^{(l+1)}_{\pi(i)} = \mathbf{x}_i^{(l+1)}.$$

The equality holds exactly for the evaluation-time map; with dropout active a single mask is applied to corresponding entries in the two runs.

Proof. Each depth keeps its own block of channels, and the order of the blocks is fixed by the depth index rather than by the node index, so the concatenation at $\pi(i)$ holds the same blocks in the same order as the concatenation at i. The projection $\mathbf{W}_{\text{comb}}^{(l)}$ is shared across nodes and acts per node, the residual $\mathbf{x}_i^{(l)}$ corresponds by the hypothesis, and LayerNorm and SiLU act per node with shared parameters; the concatenation itself is per node, so all of them preserve the correspondence. ■

The concatenation is permutation-equivariant; it also keeps more information than a sum. This is direct, and we give the proof nonetheless.

Proposition 4 (Concatenation retains more than a sum)**.** *With $D_{\max} \ge 2$ and $\mathbf{x}, \mathbf{h}^1, \dots, \mathbf{h}^{D_{\max}} \in \mathbb{R}^H$, define*

$$\mathbf{c} = [\,\mathbf{x} \,\|\, \mathbf{h}^1 \,\|\, \cdots \,\|\, \mathbf{h}^{D_{\max}}\,], \qquad \mathbf{s} = \mathbf{x} + \sum_{d=1}^{D_{\max}} \mathbf{h}^d.$$

Then $\mathbf{s}$ is a fixed linear function of $\mathbf{c}$, but $\mathbf{c}$ is not a function of $\mathbf{s}$.

Proof. The first claim holds because the sum takes the coefficient 1 on $\mathbf{x}$ and on every $\mathbf{h}^d$ inside $\mathbf{c}$. For the second, take in one tuple

$$\mathbf{h}^1 = \mathbf{u}, \qquad \mathbf{h}^2 = \mathbf{u}',$$

and in the other

$$\tilde{\mathbf{h}}^1 = \mathbf{u} + \mathbf{w}, \qquad \tilde{\mathbf{h}}^2 = \mathbf{u}' - \mathbf{w} \qquad (\mathbf{w} \neq \mathbf{0}),$$

the remaining depths equal. Both give the same $\mathbf{s}$ but different $\mathbf{c}$, so no map from $\mathbf{s}$ recovers the per-depth attribution, and any map applied after the sum takes the same value on both tuples. ■

B.6 Permutation equivariance of the message-passing step

This subsection composes the four steps above into the whole message-passing step and proves that the composition is permutation-equivariant.

Renumbering the nodes renames its results along with them, and L stacked layers with the readout give the same graph-level output. Let π renumber the nodes of G into $G' = \pi(G)$, carrying the features along ($\mathbf{X}'[\pi(i)] = \mathbf{X}[i]$, $\mathbf{E}'[\pi(i), \pi(j)] = \mathbf{E}[i, j]$). Every parameter is indexed by the layer and the depth alone, so both runs use the same parameters, and the division into shells is renumbered with the nodes (Appendix B.2). The sums $\mathbf{pes}$, $\mathbf{ses}$, $\mathbf{pls}$ are indexed by the node pair and the ledger $\mathbf{Z}$ by the node, and all of them are permutation-equivariant (Appendix B.3); the concatenation is per node.

Proposition 5 (Permutation equivariance of the message-passing step)**.** *With the same parameters and the same $D_{\max}$ in both runs, with the features carried along by the renumbering, and with the layer input produced by a per-node map shared across nodes, L stacked layers and a readout head shared across graphs give*

$$\mathbf{h}'^{d}_{\pi(i)} = \mathbf{h}^{d}_{i}, \qquad \mathbf{x}'^{(l+1)}_{\pi(i)} = \mathbf{x}^{(l+1)}_{i} \qquad (d \le D_{\max}),$$

and the mean-pooling readout gives the same graph-level output for G and G'. The equalities hold exactly for the evaluation-time map; with dropout active the two runs can be coupled draw by draw, using a single mask on corresponding entries in the two runs.

Proof. By Proposition 1 the division into shells is renumbered with the nodes, and by the proof in Appendix B.3 the three sums $\mathbf{pes}^{d}_{ij}$, $\mathbf{ses}^{d}_{ij}$ and $\mathbf{pls}^{d}_{ij}$ are permutation-equivariant (the $\mathbf{pls}$ term using the correspondence of $\mathbf{Z}^{d-1}$ supplied by the induction below), so the environment built from them, which is indexed by the node pair, satisfies

$$\mathbf{env}'^{d}_{\pi(i)\pi(j)} = \mathbf{env}^{d}_{ij}.$$

Induct on the depth $d \le D_{\max}$; the induction is well founded because a depth-d access uses $\mathbf{Z}^{d-1}$ and $\mathbf{Z}^{d}$ is updated only afterwards. The base case is $\mathbf{Z}'^{0}[\pi(i)] = \mathbf{Z}^{0}[i]$, which follows from $\mathbf{H}'^{(l)} = \pi \cdot \mathbf{H}^{(l)}$. Assume $\mathbf{Z}'^{d-1}[\pi(p)] = \mathbf{Z}^{d-1}[p]$ for every p. The concatenated blocks correspond, the layer input being permutation-equivariant by induction on l, and a single dropout mask is applied to corresponding entries in the two runs, so the two runs couple depth by depth and layer by layer. By Proposition 2 the messages and the gates, which are indexed by the node pair, and the aggregation, whose output is indexed by the node, are permutation-equivariant, so $\mathbf{h}'^{d}_{\pi(i)} = \mathbf{h}^{d}_{i}$. The same reindexing gives

$$\mathbf{E}'^{d}_{\mathrm{acc}}[\pi(i)] = \mathbf{E}^{d}_{\mathrm{acc}}[i],$$

and adding both to the ledger gives

$$\mathbf{Z}'^{d}[\pi(i)] = \mathbf{Z}^{d}[i],$$

which closes the induction. By Proposition 3 the concatenated update gives $\mathbf{x}'^{(l+1)}_{\pi(i)} = \mathbf{x}^{(l+1)}_{i}$, and induction on l extends this to the L-layer stack, the base case being $\mathbf{x}'^{(0)}_{\pi(i)} = \mathbf{x}^{(0)}_{i}$, which the hypothesis on the layer input gives. Mean pooling sums over the nodes, which the renumbering permutes among themselves, and the head acts per graph, so the graph-level output is the same for G and G'. ■

Numerical check of the readout invariance. On real QM40 molecules under random node permutations, the graph-level output deviates by at most $\sim 10^{-6}$ in float32 (rounding noise from the non-associativity of accumulation) and by $\sim 10^{-15}$ in float64.

B.7 Expressive power strictly beyond the 1-WL test

This subsection relates the input of one layer to the node coloring of the 1-WL test. Two things in the input provide the separation: the distances the shells give, that is which shell a pair lies in ($\mathcal{I}_d$), and the two edge feature sums of the environment, where $\mathbf{pes}^{d}_{ij}$ lists the edges between two successive shells and $\mathbf{ses}^{d}_{ij}$ is the term that separates the two graphs below. The parent ledger state $\mathbf{pls}^{d}_{ij}$ and the concatenation of Eq. (12) take no part in the separation below: Proposition 6 idealizes the aggregations of the two edge feature sums to injective ones on multisets, and requires in addition that the update, including the aggregation within a shell, be injective.

The shells around the receiver determine the 1-WL color. The inputs of a layer are indexed by the BFS distance to the receiver, so the nodes and edges they cover are exactly those within distance $D_{\max}$ of the receiver, and that range is what this paragraph compares against the 1-WL color. Fix a receiver i and let $B_k(i)$ be the ball of radius k around i: the nodes at distance at most k from i, each labelled with that distance and with its node feature, and the edges among them with their edge features. After k rounds, the 1-WL test colors i through the tree of walks of length at most k that start at i; that tree carries the input features of the graph but not the distances to i. The ball

determines that tree: a walk is a sequence of nodes in which consecutive nodes are joined by an edge, and the two endpoints of an edge are at distances that differ by at most one, so a walk can be enumerated from the ball together with the distances. Two receivers whose balls are isomorphic therefore receive the same color from 1-WL, and if k rounds of 1-WL give two receivers different colors, their balls are not isomorphic.

The quantities that a layer accumulates for a node pair (i, j) are the two edge sums $\mathbf{pes}_{ij}^d$, $\mathbf{ses}_{ij}^d$ and the ledger state $\mathbf{pls}_{ij}^d$ of Eqs. (2), (3) and (7). A sum keeps the multiset of the values it adds but not which parent or neighbour supplied each one, so sums alone are weaker than the ball: take a node whose six neighbours have, among themselves, either the edges of a six-cycle or the edges of two triangles, with the same edge features, and both cases give the same value to every sum. Once the aggregations of Eqs. (2) and (3) keep the node pair that contributed each term, they determine the ball together with the node features that the layer receives.

Proposition 6 (Expressive power)**.** *Let $D_{\max} \geq 1$ and let the aggregations of Eqs. (2) and (3) keep, for each term, both the contributing node pair and the term itself, and aggregate the resulting multisets injectively. Then the quantities they keep, together with the node features that the layer receives, determine the ball $B_{D_{\max}}(i)$ for every receiver i; with an injective update, including the aggregation within a shell, and an injective graph-level readout, the resulting model separates the nodes and the graphs that $D_{\max}$ rounds of the 1-WL test separate, and it also separates some that those rounds do not: a hub joined to the six nodes of a six-cycle and a hub joined to the six nodes of two triangles obtain the same colors at every round, while the layer receives different inputs for them at $D_{\max} \geq 1$.*

Proof. For the first part, an aggregation of $\mathbf{pes}_{ij}^d$ that keeps the contributing parent p records

$$\mathbf{D}[i, p]{=}d{-}1, \qquad \mathbf{A}[p, j]{=}1,$$

and one of $\mathbf{ses}_{ij}^d$ that keeps the contributing neighbour k records

$$\mathbf{D}[i, k]{=}d, \qquad \mathbf{A}[k, j]{=}1.$$

Together with the pair sets $\mathcal{I}_d$, these quantities list the nodes of every shell, the edges inside a shell and the edges between two successive shells; together with the node features that the layer receives, that is the ball. Since the ball determines the colors that $D_{\max}$ rounds of 1-WL assign, the layer separates the nodes that those $D_{\max}$ rounds separate.

For the second part, consider two graphs that the 1-WL test cannot tell apart. In both, a hub of degree six is joined to six nodes of degree three, so the color refinement gives one color to the hub and one to a neighbour of the hub at every round, and the two color histograms agree. Standard 1-hop message passing has no same-shell term, so it separates them no better. Their balls of radius 1 differ. Take a receiver among the six nodes and its two neighbours other than the hub. In the graph of two triangles those two neighbours are joined by an edge, the third edge of the triangle through the receiver, and that edge enters $\mathbf{ses}_{ij}^1$ for the pair formed by the receiver and one of them. In the six-cycle they are joined by no edge, so no such term enters it. Therefore, with $D_{\max} \geq 1$ the layer receives different inputs for the two graphs. The six nodes other than the hub are interchangeable within each graph, so each graph gives them one representation; since the inputs of those nodes differ, an injective update gives the two graphs different values for it, the two multisets of node representations differ, and an injective readout separates the two graphs. ■

If $D_{\max}$ is at least the diameter of G, the ball is the whole graph. By the first part of the proof, the input of the layer then determines the whole graph together with the distances to the receiver. The color that the 1-WL test gives a node once its colors have stabilized depends on that graph and that node alone. Therefore the layer separates every pair of nodes and every pair of graphs that the 1-WL test separates, and with the two graphs above a layer whose $D_{\max}$ is at least the diameter is strictly more expressive than the 1-WL test.

Two remarks are needed for the proposition to hold. First, in the example the separating term is the same-shell edge feature sum $\mathbf{ses}$, the edges between two nodes at the same distance from the receiver. Second, SGGNN as specified uses the plain sums of Eqs. (2), (3) and (7) and the normalized mean of Eq. (11), so the statement concerns the idealized layer, not the model as trained. It is the counterpart, for the shells of a fixed receiver, of two observations that each carry their own idealization: k-hop aggregation with a proper encoder can identify triangle-freeness

(Nikolentzos et al., 2020), and simple paths differ from walks in the nodes they revisit, which all-simple-path aggregation exploits (Michel et al., 2023).

B.8 THE BOUNDARY CASE $D_{\max}=1$

This subsection states what the sender's environment reduces to at $D_{\max}=1$.

At $D_{\max}=1$ the sender's environment takes its 1-hop form: $\mathbf{pes}^1_{ij}=\mathbf{E}[i,j]$ is the direct edge, $\mathbf{pls}^1_{ij}=\mathbf{Z}^0[i]$ is the receiver's own ledger state (the only node at distance 0 from the receiver is the receiver itself), and $\mathbf{ses}^1_{ij}$ collects the sender's same-shell edges, that is the third edges of the triangles through the receiver. The message is then an edge-gated 1-hop message as affected by the design of the sender's environment. The $D_{\max}=1$ run keeps these extra terms, so it is *not* exactly a standard 1-hop baseline.

B.9 COMPLEXITY ANALYSIS

This subsection gives the cost of each stage, the total for one forward pass and the scaling with $D_{\max}$.

Symbols and cost conventions. Write N for the atoms, E for the edges, C for the edge-feature width, H for the hidden width, L for the stacked layers and $E_{\text{all}} = \sum_{d=1}^{D_{\max}} |\mathcal{I}_d|$ for the number of shell pairs: in contrast to 1-hop GNNs, which propagate only over the E edges, SGGNN aggregates over all E_{all} pairs. The summation index of each edge sum ranges over all N candidates, so its cost is written in terms of N^2, whereas the aggregation runs over the shell pairs alone, so its cost is written in terms of E_{all}. The expressions below are reported per molecule and per forward pass.

The cost of the BFS distances. An all-source BFS inside each molecule builds the distance matrix; for a molecule with N atoms and E edges this costs $O(N(N+E))$:

$$T_{\text{bfs}} = O\big(N(N+E)\big), \tag{16}$$

This step does not grow with L or $D_{\max}$, since the distance matrix does not depend on the depth.

The cost of the two edge sums. Each depth produces the two $N \times N \times C$ tensors $\mathbf{pes}^d$ and $\mathbf{ses}^d$, that is, two sums per depth:

$$T_{\text{sum}} = O(D_{\max} N^3 C), \qquad M_{\text{sum}} = O(D_{\max} N^2 C). \tag{17}$$

The time carries one more factor of N than the space, because the summation index of Eqs. (2) and (3), the intermediate node, does not appear in the result. Both terms are independent of L, since the two sums depend only on $\mathbf{D}$ and $\mathbf{E}$.

The cost per node pair and of the aggregation. Per pair the time is the three linear maps of Eqs. (4), (9) and (10), $O(H^2)$, and the sum of Eq. (7) that produces the parent ledger state $\mathbf{pls}^d_{ij}$, of $O(N \cdot 2H)$. The aggregation is linear in E_{all} and in H. The number of pair objects at $D_{\max}=1$ is $E_{\text{all}} = 2E$, one per direction of each edge; over the full datasets the ratio $E_{\text{all}}/2E$ at $D_{\max}=4$ is 3.49 on QM9 and 4.92 on QM40.

Time and space per forward pass. Per molecule, in training mode, where the per-layer activations are kept for the backward pass,

$$\begin{aligned} T_{\text{fwd}} &= O\big(D_{\max} N^3 C + L\, E_{\text{all}} H^2 + L\, E_{\text{all}} N H + L\, N (D_{\max}{+}1) H^2\big), \\ M_{\text{fwd}} &= O\big(D_{\max} N^2 C + L\, (D_{\max} N H + E_{\text{all}} H)\big). \end{aligned} \tag{18}$$

The four time terms are the two edge sums of the sender's environment, the three linear maps of Eqs. (4), (9) and (10), the parent ledger state of Eq. (7), and the projection of Eq. (12); the space holds a constant number of width-H tensors per depth and per pair.

The dominant terms of one forward pass. For one molecule, one forward pass and one BFS, adding the stages together:

$$T_{\text{step}} = O\Big(\underbrace{N(N+E) + L\, E_{\text{all}} H^2 + L\, N (D_{\max}{+}1) H^2}_{\text{BFS, three linear maps, projection}} + \underbrace{D_{\max} N^3 C + L\, E_{\text{all}} N H}_{\text{two edge sums, parent ledger state}} \Big). \tag{19}$$

What grows with $D_{\max}$. Three further terms are monotone in $D_{\max}$. (i) The pair set is non-decreasing in $D_{\max}$ and saturates at the graph diameter, so the pair work stops growing there. (ii) The per-depth parameter sets, the per-depth activations and the projection of the $(D_{\max}{+}1)H$ concatenation make the parameter count, the activation memory and the time of the projection linear in $D_{\max}$. (iii) The edge sums grow as in Eq. (17). Raising $D_{\max}$ beyond the diameter adds no aggregation work, and of the three only (ii) and (iii) keep growing.

B.10 THE COMPLETE FORWARD PASS

This subsection collects the full forward pass; the subsections above analyze its components.

Algorithm 1 The SGGNN forward pass over the explicit BFS shells $\mathcal{I}_d$ (Eq. (1)): L residual depth-wise SHELLGATEDLAYER layers producing per-node representations $\mathbf{H}^{(L)}$. A task readout (e.g. mean pooling) and output head are applied downstream; $\odot$ and $\oslash$ denote the elementwise product and ratio.

procedure SGGNN($\mathbf{X}, \mathbf{D}, \mathbf{E}$)
  $\{\mathbf{pes}^d, \mathbf{ses}^d\}_{d=1}^{D_{\max}} \leftarrow$ EDGESUMS($\mathbf{D}, \mathbf{E}$)
  $\mathcal{I}_d \leftarrow \{(i,j) : \mathbf{D}[i,j]{=}d\}$
  $\mathbf{H}^{(0)} \leftarrow \mathrm{MLP}(\mathbf{X})$
  for $l{=}0$ **to** $L-1$ **do**
    $\mathbf{H}^{(l+1)} \leftarrow$ SHELLGATEDLAYER($\mathbf{H}^{(l)}, l, \mathbf{D}, \{\mathcal{I}_d\}, \{\mathbf{pes}^d, \mathbf{ses}^d\}, \mathbf{E}$)
  end for
  return $\mathbf{H}^{(L)}$
end procedure
procedure EDGESUMS($\mathbf{D}, \mathbf{E}$)
  for $d{=}1$ **to** $D_{\max}$ **do**
    $\mathbf{pes}^d_{ij} \leftarrow \sum_p \mathbf{1}\{\mathbf{D}[i,p]{=}d{-}1\}\, \mathbf{E}[p,j]$
    $\mathbf{ses}^d_{ij} \leftarrow \sum_k \mathbf{1}\{\mathbf{D}[i,k]{=}d\}\, \mathbf{E}[k,j]$
  end for
  return $\{\mathbf{pes}^d, \mathbf{ses}^d\}_{d=1}^{D_{\max}}$
end procedure
procedure SHELLGATEDLAYER($\mathbf{H}^{(l)}, l, \mathbf{D}, \{\mathcal{I}_d\}, \{\mathbf{pes}^d, \mathbf{ses}^d\}, \mathbf{E}$)
  $\mathbf{A}[i,j] \leftarrow \mathbf{1}\{\sum_{c=1}^{C} \mathbf{E}[i,j,c]{>}0\} \cdot \mathbf{1}\{i{\neq}j\}$
  $\mathbf{Z}^0 \leftarrow [\,\mathbf{H}^{(l)} \,\|\, \mathbf{0}_{N\times H}\,]$
  for $d{=}1$ **to** $D_{\max}$ **do**
    $\mathbf{E}^d_{\mathrm{acc}} \leftarrow \mathbf{0}$
    $\mathbf{M}^d_{\mathrm{acc}} \leftarrow \mathbf{0}$; $\mathbf{G}^d_{\mathrm{acc}} \leftarrow \mathbf{0}$
    for all $(i,j) \in \mathcal{I}_d$ **do**
      $\mathbf{e}^d_{ij} \leftarrow \mathbf{W}^d_{\mathrm{edge}}[\,\mathbf{pes}^d_{ij} \,\|\, \mathbf{ses}^d_{ij}\,]$
      $\mathbf{E}^d_{\mathrm{acc}}[i] \leftarrow \mathbf{E}^d_{\mathrm{acc}}[i] + \mathbf{e}^d_{ij}$
      $\mathbf{pls}^d_{ij} \leftarrow \sum_p \mathbf{1}\{\mathbf{D}[i,p]{=}d{-}1\}\, \mathbf{1}\{\mathbf{A}[p,j]{=}1\}\, \mathbf{Z}^{d-1}[p]$
      $\mathbf{env}^d_{ij} \leftarrow [\,\mathbf{e}^d_{ij} \,\|\, \mathbf{pls}^d_{ij}\,]$
      $\mathbf{m}^d_{ij} \leftarrow \mathbf{W}^d_{\mathrm{msg}}[\,\mathbf{x}^{(l)}_j \,\|\, \mathbf{env}^d_{ij}\,]$
      $\mathbf{g}^d_{ij} \leftarrow \sigma(\mathbf{W}^d_{\mathrm{gate}}[\,\mathbf{x}^{(l)}_i \,\|\, \mathbf{x}^{(l)}_j \,\|\, \mathbf{env}^d_{ij}\,])$
      $\mathbf{M}^d_{\mathrm{acc}}[i] \leftarrow \mathbf{M}^d_{\mathrm{acc}}[i] + \mathbf{g}^d_{ij} \odot \mathbf{m}^d_{ij}$ ▷ numerator of Eq. (11)
      $\mathbf{G}^d_{\mathrm{acc}}[i] \leftarrow \mathbf{G}^d_{\mathrm{acc}}[i] + \mathbf{g}^d_{ij}$ ▷ denominator of Eq. (11)
    end for
    $\mathbf{H}^d \leftarrow \mathbf{M}^d_{\mathrm{acc}} \oslash (\mathbf{G}^d_{\mathrm{acc}} + \epsilon)$
    $\mathbf{H}^d \leftarrow \mathrm{Dropout}(\mathrm{SiLU}(\mathrm{LN}(\mathbf{H}^d)))$
    $\mathbf{Z}^d \leftarrow \mathbf{Z}^{d-1} + [\,\mathbf{H}^d \,\|\, \mathbf{E}^d_{\mathrm{acc}}\,]$
  end for
  return $\mathrm{Dropout}(\mathrm{SiLU}(\mathrm{LN}(\mathbf{W}^{(l)}_{\mathrm{comb}}[\,\mathbf{H}^{(l)} \,\|\, \mathbf{H}^1 \,\|\, \cdots \,\|\, \mathbf{H}^{D_{\max}}\,]))) + \mathbf{H}^{(l)}$
end procedure

C Experimental Setup Details

C.1 Datasets

The QM9 and QM40 splits are fixed by seed 42; PCQM4Mv2 uses the official SMILES and the official train/validation split of the benchmark.

QM9. QM9 (Ramakrishnan et al., 2014) is one of the most widely used benchmarks for molecular property prediction: it covers up to nine heavy atoms over C/N/O/F, with labels computed at the B3LYP/6-31G(2df,p) level of theory. The raw GDB-9 data (133,885 molecules) are filtered by the official PyG InChI-deduplicated set (130,831), then by RDKit parseability (129,012), and finally by SMILES deduplication (128,910) to remove entries that 2D models cannot distinguish (identical SMILES but different DFT labels). The 19 DFT properties are read from the CSV in original units; the six electronic properties (μ, α, $\varepsilon_{\mathrm{HOMO}}$, $\varepsilon_{\mathrm{LUMO}}$, $\Delta\varepsilon$, $\langle R^2 \rangle$) are read from the corresponding standard QM9 columns, $\langle R^2 \rangle$ from the column `r2`; for $\varepsilon_{\mathrm{HOMO}}$/$\varepsilon_{\mathrm{LUMO}}$/$\Delta\varepsilon$ the Hartree values are converted to meV in the reported tables.

QM40. QM40 (Madushanka et al., 2024) (*Scientific Data* 11, 1376, 2024) is a larger benchmark drawn from the ZINC database (Sterling & Irwin, 2015): 162,954 drug-like molecules that, as the dataset paper reports, span 88% of the FDA-approved drug chemical space; the graphs we build from them contain 10–37 heavy atoms, so every molecule is larger than the largest QM9 molecule, and the element set adds sulfur and chlorine to QM9's C/N/O/F. Its electronic properties are computed at the B3LYP/6-31G(2df,p) level of theory, the *same* method and basis set as QM9 (Ramakrishnan et al., 2014; Madushanka et al., 2024). We take the same six labels from its core main table, read from its columns with no unit conversion ($\varepsilon_{\mathrm{HOMO}}$/$\varepsilon_{\mathrm{LUMO}}$/$\Delta\varepsilon$ are converted to meV only for reporting), so the labels and the units follow QM9. Its gap column carries the opposite sign to QM9's, and since MAE and σ_p are insensitive to an overall sign, the reported numbers are unaffected.

Construction of the splits. QM9 uses a *gap-stratified* split: molecules are assigned to strata by the decile of their gap label and the $80/10/10$ split is drawn within those strata, which keeps the distribution of that label aligned across the three parts. QM40 uses a plain uniform permutation. All models use the same split.

C.2 Preprocessed input features

Input features. Each molecule is converted to a 2D molecular graph with RDKit, with the hydrogen atoms removed. QM9 and QM40 use the same kind of input: per-atom descriptors (element, degree, formal charge, H count, hybridization, aromaticity and ring membership) and a 13-dim per-edge encoding (bond type one-hot for single/double/triple/aromatic, in-ring, conjugated and a 7-way stereochemistry encoding). The two datasets differ only in the size of the element block, which is set by the elements they contain: 24 dimensions on QM9 (five element slots H/C/N/O/F, with the H entry always zero after hydrogen removal) and 25 on QM40 (six elements C/N/O/F/S/Cl). PCQM4Mv2 keeps the same edge encoding and extends the element block to the 24 elements occurring in that dataset plus one slot for elements outside it, giving 44 dimensions; that benchmark requires only that *inference uses 2D information alone*, and does not prescribe the input descriptors.

Unified pipeline. On QM9 and QM40, scripts shipped with the repository build the molecular features and the labels directly from the raw chemical data, and the molecules, the split and the labels are shared by all models and baselines.

C.3 Computational environment

SGGNN is trained and evaluated under Python 3.10.20, PyTorch 2.13.0 (CUDA 13.0), PyTorch Geometric 2.8.0, NumPy 2.2.6, pandas 2.3.3, scikit-learn 1.7.2 and RDKit 2026.03.5, on a single NVIDIA GeForce RTX 5070 Ti Laptop GPU with 12 GB of memory.

C.4 HYPERPARAMETERS AND OTHER SETTINGS

Table 5 collects the SGGNN configuration used throughout, and reports only SGGNN's own settings.

Table 5: SGGNN hyperparameters and other settings for the two configurations used in this work, QM9/QM40 and PCQM4Mv2.

Hyperparameter	**QM9, QM40**	**PCQM4Mv2**
Loss function	L1	L1
Optimizer	AdamW, weight decay 10^{-5}	AdamW, weight decay 10^{-5}
Learning rate	10^{-3}	2×10^{-3}
Learning-rate schedule	cosine annealing; gradient clipping at 1.0	cosine annealing; gradient clipping at 1.0
Weight initialization	orthogonal (gain 1.68), zero bias	orthogonal (gain 1.68), zero bias
Epochs	100 (QM9) / 15 (QM40), validation-selected	50
Batch size	128	384
Hidden width H	256	256
ShellGatedLayer layers L	3 (residual)	3 (residual)
Maximum shell depth $D_{\max}$	4 (validation-selected)	4
Activation	SiLU	SiLU
Dropout	0.1	0.1
Normalization constant ϵ	10^{-6}	10^{-6}
Node features	QM9 24 / QM40 25-dimensional	44-dimensional
Edge features	13-dimensional	13-dimensional
Input embedding	2-layer MLP $F \to H \to H$ with LayerNorm and SiLU	2-layer MLP $F \to H \to H$ with LayerNorm and SiLU
Pooling	mean pooling	mean pooling
Output head	3-layer MLP $H \to H/2 \to H/4 \to 6$	3-layer MLP $H \to H/2 \to H/4 \to 1$
Trainable parameters	QM9 8,406,214 / QM40 8,406,470	8,411,009

C.5 EXTRA SETTINGS

Chemprop is distributed as a maintained software toolkit: we run its official code (ChemProp Development Team, MIT) v2.2.3.

D ADDITIONAL EXPERIMENTAL RESULTS

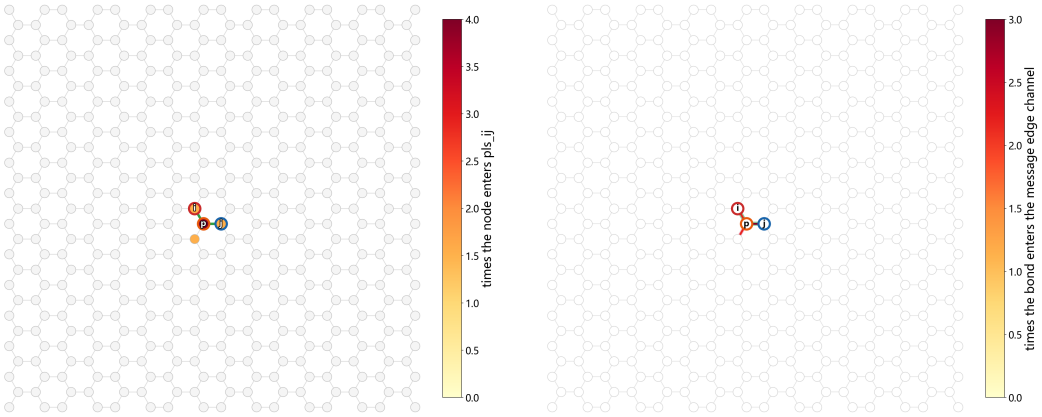
This appendix gives an experiment that does not appear in the main text, and the tables that supplement data the main text reports only in summary form.

D.1 THE COVERAGE OF THE SENDER'S ENVIRONMENT

We measure the coverage of the sender's environment used in training directly: the input features of one node at a time are shifted by a constant and the layer is run again; a node counts as covered by a tensor when that tensor moves by more than 10^{-9}. The coverage is fixed by the shells and the parent sets, not by the learned weights, so the measurement runs on a randomly initialized SGGNN (L=3, H=256, $D_{\max}$=4, seed 0, evaluation mode).

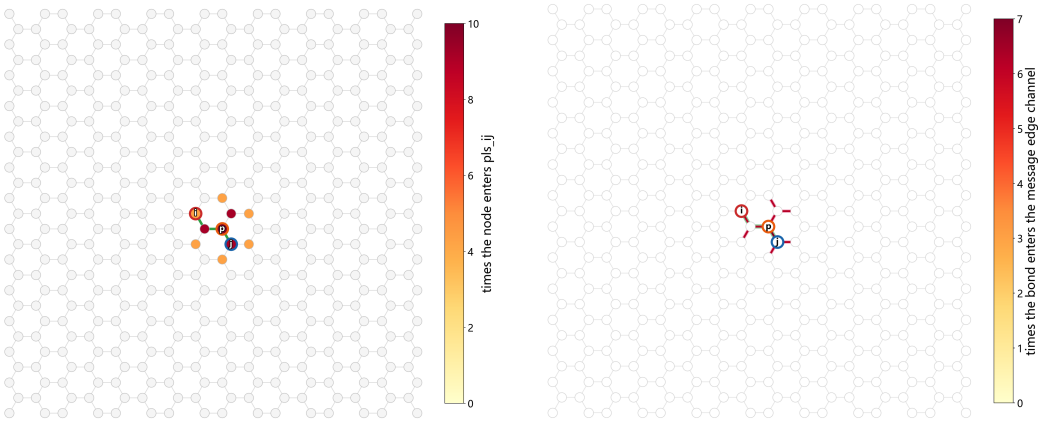
Figures 5–9 draw the coverage of the environment of a single depth-d message, $d = 2, \ldots, 6$, on a honeycomb flake: the left panel counts how many nodes enter $\mathbf{pls}_{ij}^{d}$, the right panel the parent and same-shell edges whose features enter the message, counted from their index sets. Both grow with d. The same measurement on a 14-node path, with the receiver at one end, node features drawn as $\mathcal{N}(0, 0.5^2)$, a perturbation of $+1.0$ and a threshold of 10^{-12}, returns, for the coverage radius, that is

the largest BFS distance from the receiver among the nodes that enter the message, exactly $1, 2, 4, 7$ for $d = 1, \ldots, 4$ (the thresholds 10^{-9} and 10^{-12} give the same radius); this matches the bound $1 + \frac{d(d-1)}{2}$ of Appendix B.3 and shows that it is tight.



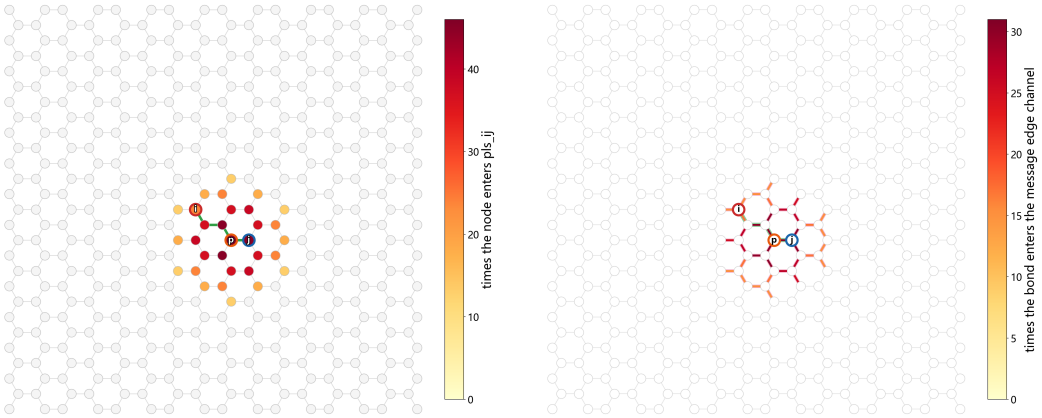
(a) nodes summed into $\mathbf{pls}_{ij}^{d}$ (b) parent and same-shell edges

Figure 5: Coverage of the sender's environment of one depth-2 message, $\mathbf{D}[i, j] = 2$, on a honeycomb flake: the nodes of (a) and the edges of (b) are shaded by how many times they enter $\mathbf{pls}_{ij}^{d}$ and the edge channel of the message. The marked nodes are the receiver i, whose representation is updated, the sender j, the other end of the pair at distance d from i, and one parent p of the sender on a shortest path to i.



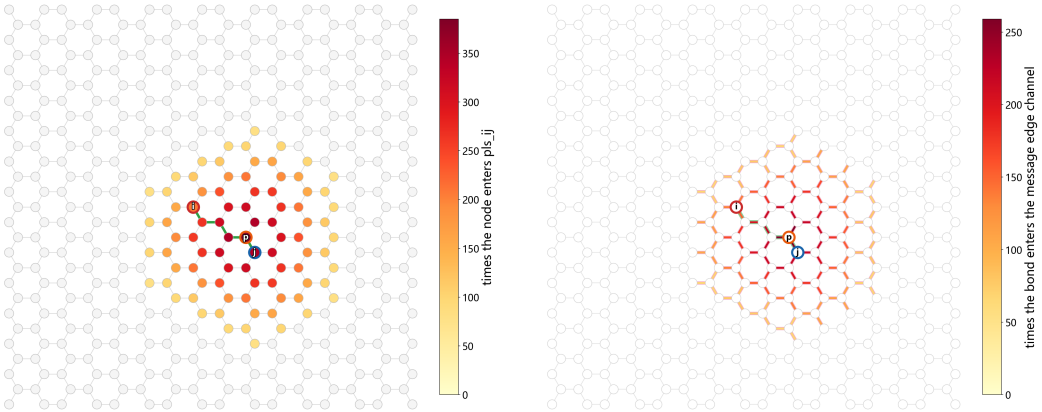
(a) nodes summed into $\mathbf{pls}_{ij}^{d}$ (b) parent and same-shell edges

Figure 6: Coverage of the sender's environment of one depth-3 message, $\mathbf{D}[i, j] = 3$. Same conventions as Figure 5.



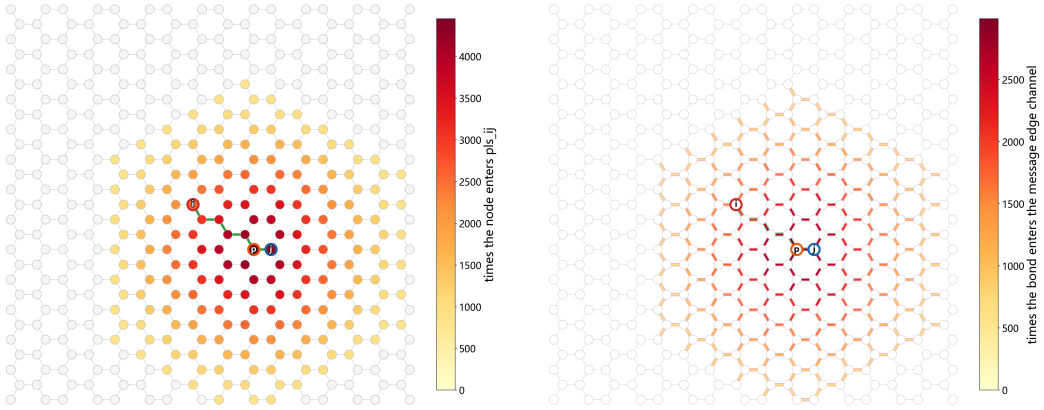
(a) nodes summed into $\mathbf{pls}_{ij}^{d}$ (b) parent and same-shell edges

Figure 7: Coverage of the sender's environment of one depth-4 message, $\mathbf{D}[i, j] = 4$. Same conventions as Figure 5.



(a) nodes summed into $\mathbf{pls}_{ij}^{d}$ (b) parent and same-shell edges

Figure 8: Coverage of the sender's environment of one depth-5 message, $\mathbf{D}[i, j] = 5$. Same conventions as Figure 5.



(a) nodes summed into $\mathbf{pls}_{ij}^{d}$

(b) parent and same-shell edges

Figure 9: Coverage of the sender's environment of one depth-6 message, $\mathbf{D}[i, j] = 6$. Same conventions as Figure 5.

D.2 Supplementary tables for the main text

Table 6: Per-property test MAE of the ablation study of Table 4 ($D_{\max}$=4 on QM9; lower is better, $\varepsilon_{\mathrm{HOMO}}$/$\varepsilon_{\mathrm{LUMO}}$/$\Delta\varepsilon$ in meV). Each entry is the mean over three independent runs with the sample std in parentheses. Rows and row labels follow Table 4, which reports the unitless std-MIX and the reductions relative to the first row.

$\mathbf{e}_{ij}^{d}$	$\mathbf{pls}_{ij}^{d}$	comb.	μ	α	$\varepsilon_{\mathrm{HOMO}}$	$\varepsilon_{\mathrm{LUMO}}$	$\Delta\varepsilon$	$\langle R^2 \rangle$
×	×	sum	0.3364(0.0020)	0.2786(0.0004)	67.84(0.28)	71.73(0.09)	92.16(0.24)	17.9333(0.1170)
×	×	cat	0.3263(0.0016)	0.2699(0.0021)	64.48(0.30)	67.62(0.25)	87.13(0.36)	17.0125(0.0565)
×	✓	cat	0.3030(0.0020)	0.2432(0.0023)	58.26(0.12)	61.88(0.38)	78.99(0.05)	15.6304(0.0920)
✓	×	cat	0.3180(0.0019)	0.2537(0.0013)	60.81(0.32)	64.08(0.16)	81.80(0.45)	16.4687(0.0497)
✓	✓	sum	0.3148(0.0003)	0.2515(0.0032)	60.26(0.05)	63.88(0.27)	81.50(0.34)	16.1149(0.0986)
✓	✓	cat	0.3003(0.0008)	0.2421(0.0018)	57.59(0.08)	60.86(0.27)	77.75(0.18)	15.5275(0.1005)

Range sweep tables. The range sweep discussed in Section 4.3 is reported here as two separate tables, one per dataset, to keep each readable. The per-property test MAE use the same units as the main tables (Tables 1 and 2, with $\varepsilon_{\mathrm{HOMO}}$/$\varepsilon_{\mathrm{LUMO}}$/$\Delta\varepsilon$ in meV); the last column of each table is the unitless std-MIX (lower is better). $D_{\max}$=1 switches off all multi-hop shells (approximately 1-hop message passing); the best std-MIX is in bold ($D_{\max}$=4 on both datasets; on QM9 the neighbouring depths lie within the quoted std of that value, so the minimum is not resolved there). Each depth is its own set of three runs; the $D_{\max}$=4 row of each dataset is the same set of three runs that produces the SGGNN row of the corresponding main table (Tables 1 and 2).

Table 7: Range sweep $D_{\max}$=1–9 on QM9: per-property test MAE and std-MIX (last column, unitless, lower is better). Each depth is trained three times; every entry is the mean over those three runs, with that depth's own run-to-run sample std in parentheses ($\varepsilon_{\mathrm{HOMO}}$/$\varepsilon_{\mathrm{LUMO}}$/$\Delta\varepsilon$ in meV). A std of the last column that would round to zero at four decimals is given one extra digit, and depth-to-depth differences are only resolved where they exceed the quoted std.

$D_{\max}$	μ	α	$\varepsilon_{\mathrm{HOMO}}$	$\varepsilon_{\mathrm{LUMO}}$	$\Delta\varepsilon$	$\langle R^2 \rangle$	std-MIX
1	0.3462(0.0014)	0.3302(0.0036)	70.30(0.12)	74.32(0.42)	95.30(0.15)	19.1420(0.0550)	0.0982(0.0001)
2	0.3105(0.0020)	0.2545(0.0015)	59.93(0.29)	63.87(0.44)	81.52(0.48)	16.3415(0.1023)	0.0850(0.0004)
3	0.3027(0.0005)	0.2450(0.0021)	58.08(0.10)	61.75(0.35)	78.78(0.39)	15.9768(0.1003)	0.0826(0.0002)
4	0.3016(0.0012)	0.2434(0.0017)	57.52(0.31)	60.97(0.24)	77.51(0.39)	15.5938(0.1420)	**0.0817(0.0004)**
5	0.3050(0.0015)	0.2353(0.0010)	57.90(0.24)	61.23(0.28)	77.92(0.26)	15.3329(0.0729)	0.0820(0.00004)
6	0.3035(0.0014)	0.2389(0.0011)	57.38(0.16)	61.19(0.48)	78.00(0.20)	15.3963(0.0742)	0.0818(0.0001)
7	0.3027(0.0002)	0.2373(0.0014)	57.50(0.26)	61.14(0.19)	77.87(0.14)	15.5496(0.1076)	0.0818(0.00002)
8	0.3029(0.0014)	0.2377(0.0019)	57.79(0.15)	61.38(0.29)	78.01(0.13)	15.4679(0.0976)	0.0819(0.0003)
9	0.3078(0.0018)	0.2367(0.0016)	58.35(0.15)	61.55(0.26)	78.65(0.23)	15.4683(0.0271)	0.0827(0.0002)

Table 8: Range sweep $D_{\max}$=1–9 on QM40: per-property test MAE and std-MIX (last column, unitless, lower is better). Each depth is trained three times; every entry is the mean over those three runs, with that depth's own run-to-run sample std in parentheses ($\varepsilon_{\mathrm{HOMO}}$/$\varepsilon_{\mathrm{LUMO}}$/$\Delta\varepsilon$ in meV). A std of the last column that would round to zero at four decimals is given one extra digit, and depth-to-depth differences are only resolved where they exceed the quoted std.

$D_{\max}$	μ	α	$\varepsilon_{\mathrm{HOMO}}$	$\varepsilon_{\mathrm{LUMO}}$	$\Delta\varepsilon$	$\langle R^2 \rangle$	std-MIX
1	1.2052(0.0017)	4.4829(0.0721)	122.82(0.22)	116.64(0.23)	153.73(0.36)	1209.9729(5.5868)	0.2137(0.0004)
2	1.1702(0.0021)	3.1465(0.0278)	114.78(0.14)	107.17(0.23)	139.74(0.15)	1094.5952(3.2753)	0.1989(0.0003)
3	1.1614(0.0015)	2.7028(0.0076)	112.42(0.28)	103.89(0.04)	136.05(0.12)	1070.1151(1.4043)	0.1947(0.0002)
4	1.1615(0.0011)	2.4590(0.0071)	111.59(0.18)	102.57(0.21)	134.52(0.13)	1063.5413(1.3831)	**0.1933(0.000033)**
5	1.1749(0.0018)	2.3627(0.0127)	112.27(0.34)	103.34(0.22)	135.49(0.26)	1062.7024(1.8984)	0.1946(0.0004)
6	1.1770(0.0026)	2.2761(0.0102)	111.98(0.38)	103.21(0.41)	135.09(0.27)	1060.5470(2.4756)	0.1943(0.0002)
7	1.1847(0.0011)	2.2468(0.0096)	112.68(0.16)	103.15(0.07)	135.53(0.36)	1061.2046(3.5546)	0.1951(0.0001)
8	1.1895(0.0054)	2.2145(0.0116)	112.26(0.16)	103.45(0.24)	135.69(0.52)	1064.0234(1.5260)	0.1954(0.0005)
9	1.1922(0.0017)	2.1802(0.0013)	113.48(0.19)	103.68(0.17)	136.78(0.20)	1060.3765(1.8866)	0.1961(0.0002)

Table 9: Training-set standard deviation σ_p used to normalize Eq. (13), in the raw unit of each property (Hartree for $\varepsilon_{\mathrm{HOMO}}$/$\varepsilon_{\mathrm{LUMO}}$/$\Delta\varepsilon$, as in the data); the ratio is unitless, so the meV display of Tables 1 and 2 leaves std-MIX unchanged.

Dataset	μ	α	$\varepsilon_{\mathrm{HOMO}}$	$\varepsilon_{\mathrm{LUMO}}$	$\Delta\varepsilon$	$\langle R^2 \rangle$
QM9	1.5003	8.2103	0.02204	0.04664	0.04704	281.9965
QM40	2.2413	74.0166	0.01808	0.03241	0.03792	7903.1572

The two figures below give the same sweep property by property, for QM9 in Fig. 10 and for QM40 in Fig. 11, and the curves differ from one another. Measured as the fall from each property's own error at the shallowest setting, the polarizability α improves the most of the six and the dipole moment μ the least. On QM40 the polarizability keeps improving all the way to the deepest depth of the sweep, whereas the dipole moment reaches its minimum early in the sweep and then slowly worsens. The frontier-orbital energies, their gap and the electronic spatial extent lie in between, and beyond $D_{\max}$=4, where the aggregate std-MIX reaches its minimum, their remaining variation is small next to the fall before it. The polarizability improves further on QM40 than on QM9.

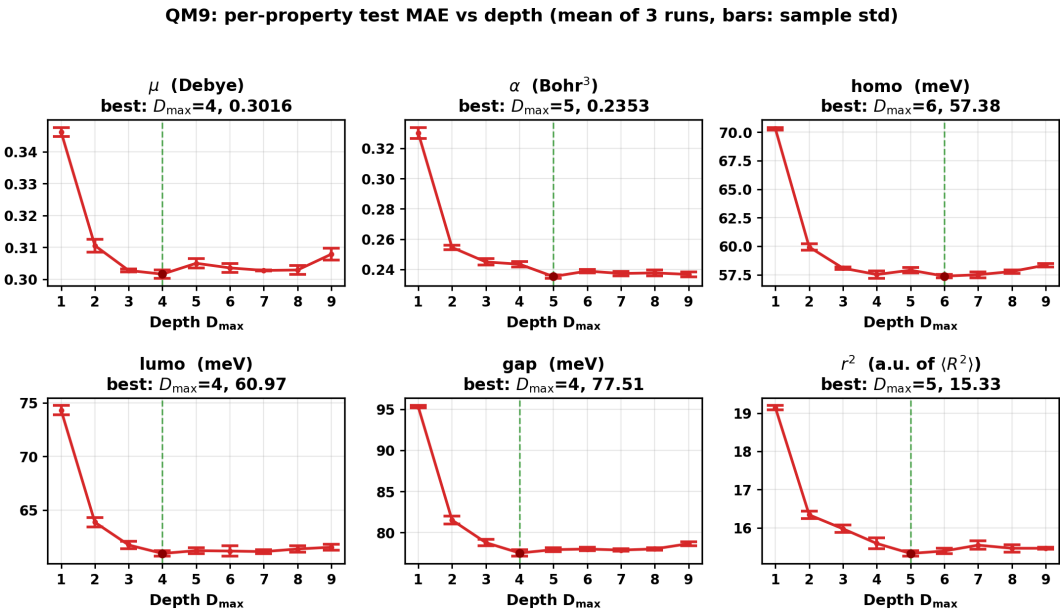


Figure 10: Per-property test MAE against depth on QM9: $\varepsilon_{\mathrm{HOMO}}/\varepsilon_{\mathrm{LUMO}}/\Delta\varepsilon$ in meV, the other properties in their raw unit (same runs as Table 7). Every point is the mean over three runs and the bars are the run-to-run sample std; the dashed green line and the dark point mark that property's minimum.

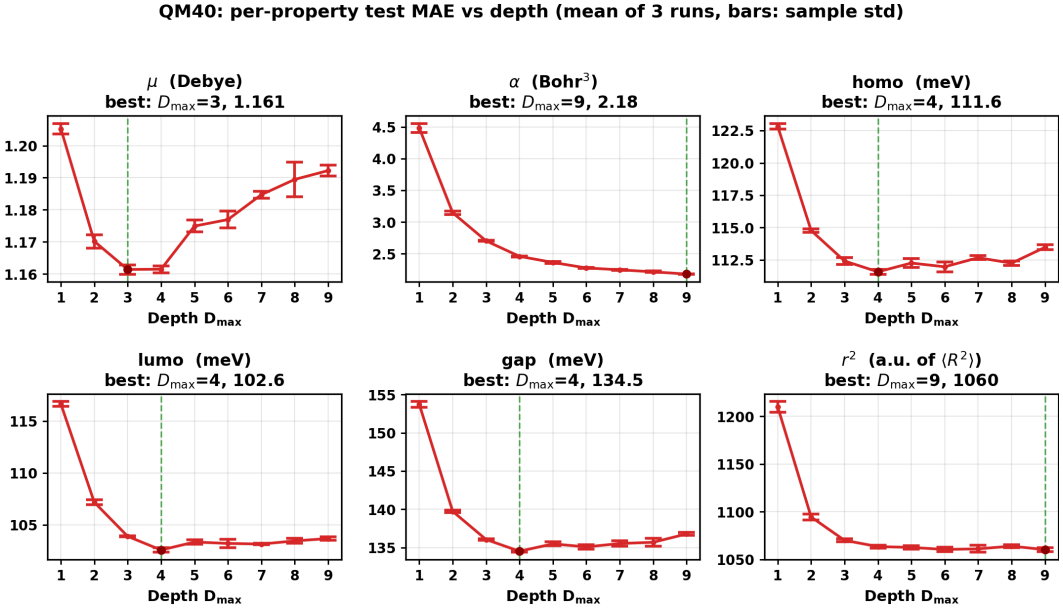


Figure 11: Per-property test MAE against depth on QM40, same conventions as Fig. 10 (same runs as Table 8).