Graph Neural Networks with Adaptive Readouts

David Buterez *[1] **Jon Paul Janet** [2] **Steven J. Kiddle** [3] **Dino Oglic** [3] **Pietro Liò** [1]

[1] Department of Computer Science and Technology, University of Cambridge, UK
[2] CVRM, BioPharmaceuticals R&D, AstraZeneca, Sweden
[3] DS&AI, BioPharmaceuticals R&D, AstraZeneca, UK

Abstract

An effective aggregation of node features into a graph-level representation via readout functions is an essential step in numerous learning tasks involving graph neural networks. Typically, readouts are simple and non-adaptive functions designed such that the resulting hypothesis space is permutation invariant. Prior work on deep sets indicates that such readouts might require complex node embeddings that can be difficult to learn via standard neighborhood aggregation schemes. Motivated by this, we investigate the potential of adaptive readouts given by neural networks that do not necessarily give rise to permutation invariant hypothesis spaces. We argue that in some problems such as binding affinity prediction where molecules are typically presented in a canonical form it might be possible to relax the constraints on permutation invariance of the hypothesis space and learn a more effective model of the affinity by employing an adaptive readout function. Our empirical results demonstrate the effectiveness of neural readouts on more than 40 datasets spanning different domains and graph characteristics. Moreover, we observe a consistent improvement over standard readouts (i.e., sum, max, and mean) relative to the number of neighborhood aggregation iterations and different convolutional operators.

1 Introduction

We investigate empirically the potential of adaptive and differentiable readout functions for learning an effective representation of graph structured data (e.g., molecular, social, biological, and other relational data) using graph neural networks (GNNs). Recently, there has been a surge of interest in developing neural architectures from this class [1–5]. Graph neural networks typically employ a permutation invariant neighborhood aggregation scheme that is repeated for several iterations, where in each iteration node representations are updated by aggregating the feature vectors corresponding to their neighbors. The process is repeated for a pre-specified number of iterations and the resulting node representations capture the information contained in the respective neighborhoods given by vertex rooted sub-trees. The final step is aggregation of node features into a graph-level representation using a readout function. The readouts are typically selected such that the resulting hypothesis space is permutation invariant. For instance, simple functions such as sum, mean, and max, all satisfy this requirement [6, 7]. Graph neural networks can, thus, be seen as a special case of representation learning over (node) sets. Zaheer et al. [6] have studied learning on sets and demonstrated that a permutation invariant hypothesis over such domains admits a decomposition as a sum of individual set items represented in a latent space given by a suitable embedding function. In a follow up work, Wagstaff et al. [7] have demonstrated that simple pooling/readout functions such as sum, mean, or max might require complex item/node embedding functions that might be difficult to learn using standard neural networks. The expressiveness of graph neural networks specifically has also been studied in [5], where it has been recommended to use an injective neighborhood aggregation scheme.

*Correspondence to: David Buterez <db804@cam.ac.uk>.

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

For such schemes, it can be demonstrated that graph neural networks can be as expressive as the Weisfeiler–Lehman isomorphism test which is known to be an effective and computationally efficient approximation scheme for differentiating between a large number of graph isomorphism classes [8, 9].

As it can be challenging to learn a permutation invariant hypothesis over graphs using simple readouts, we empirically investigate possible extensions and relaxations for problems where graphs might be presented in a canonical form (i.e., with an identical ordering of vertices). In such cases, it might be possible to relax the constraint on permutation invariance of the hypothesis space. For instance, in problems such as binding affinity prediction, molecular graphs are typically generated from a canonical SMILES representation and, thus, inputs to graph neural networks are graphs with a fixed ordering of nodes. The latter is sufficient to ensure consistent predictions over molecular graphs for graph neural networks with readouts that do not give rise to permutation invariant hypotheses.

We start with a review of graph neural networks and then focus on introducing different classes of adaptive and differentiable readout functions. The first class of such readouts is based on set transformers [10] and it gives rise to permutation invariant hypotheses. We then slightly relax the constraint on permutation invariance of graph-level representations by introducing readouts inspired by a neighborhood aggregation scheme known as Janossy pooling [11]. These approximately permutation invariant readouts are based on multi-layer perceptrons (MLP) and recurrent neural architectures known as GRUs. Finally, we consider neural readouts based on plain MLP and GRU architectures, thus completely lifting the constraint on permutation invariance of the hypothesis space.

Our empirical study is extensive and covers more than 40 datasets across different domains and graph characteristics. The ultimate goal of the study is to explore the potential of learning hypotheses over graph structured data via adaptive and differentiable readouts. To this end, we first consider the most frequently used neighborhood aggregation schemes or convolutional operators and fix the number of iterations to two. Our empirical results demonstrate a significant improvement as a result of employing neural readouts, irrespective of the convolutional operator and dataset/domain. Following this, we then compare our neural readouts to the standard readouts (sum, max, mean) while varying the number of neighborhood aggregation iterations. The results indicate that neural readout functions are again more effective than the standard readouts, with a significant difference in performance between different neighborhood aggregation schemes. We hypothesize that the latter might be due to the expressiveness of different neighborhood aggregation operators. More specifically, in drug design and lead optimization it is typical that through a change in sub-structure of a parent compound that one can improve the potency. These changes are local and we hypothesize that they would be reflected in a small number of node representations, whose signal could be consumed by a large number of noisy nodes within the standard readouts. We also present results of an ablation study on the influence of node permutations on the hypotheses learned by GNNs with neural readouts that do not enforce permutation invariance. The results indicate that there might be some node permutations that are detrimental to the predictions and this might be an interesting avenue for future work.

We conclude with an experiment involving multi-million scale proprietary datasets from AstraZeneca that have been collected by primary screening assays. Our results again demonstrate that the avenue of neural readouts merits further exploration from both theoretical and empirical perspectives. More specifically, we observe that only plain MLP readouts significantly improve the performance on these challenging tasks and they do not give rise to permutation invariant hypotheses. Extensive results, including additional datasets and neural architectures (variational graph autoencoders and visualizations of latent spaces that correspond to different readouts) have been provided in Appendix L. An analysis involving computational and memory costs and trade-offs can be found in Appendix I.

2 Graph Neural Networks with Adaptive and Differentiable Readouts

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a graph, where $\mathcal{V}$ is the set of *nodes* or *vertices* and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is the set of *edges*. Suppose the nodes are associated with d-dimensional feature vectors $\mathbf{x}_u$ for all $u \in \mathcal{V}$. Let A be the adjacency matrix of a graph G such that $A_{uv} = 1$ if $(u, v) \in \mathcal{E}$ and $A_{uv} = 0$ otherwise. For a vertex $u \in \mathcal{V}$ denote the set of neighboring nodes with $\mathcal{N}_u = \{v \mid (u, v) \in \mathcal{E} \vee (v, u) \in \mathcal{E}\}$. Suppose also that a set of graphs with corresponding labels $\{(G_i, y_i)\}_{i=1}^{n}$ has been sampled independently from some target probability measure defined over $\mathcal{G} \times \mathcal{Y}$, where $\mathcal{G}$ is a space of graphs and $\mathcal{Y} \subset \mathbb{R}$ is the set of labels. We are interested in the problem of learning a graph neural network that can approximate well the target label $y \in \mathcal{Y}$ for a given graph $G \in \mathcal{G}$.

Henceforth, we will assume that a graph G is represented with a tuple (X_G, A_G), with X_G denoting the matrix with node features as rows and A_G the adjacency matrix. Graph neural networks take such tuples as inputs and generate predictions over the label space. A function f defined over a graph G is called permutation invariant if there exists a permutation matrix P such that $f(PX_G, PA_GP^\top) = f(X_G, A_G)$. In general, graph neural networks aim at learning permutation invariant hypotheses to have consistent predictions for the same graph when presented with permuted vertices/nodes. This property is achieved through neighborhood aggregation schemes and readouts that give rise to permutation invariant hypotheses. More specifically, the node features X_G and the graph structure (adjacency matrix) A_G are used to first learn representations of nodes h_v, for all $v \in \mathcal{V}$. The neighborhood aggregation schemes enforce permutation invariance by employing standard pooling functions — sum, mean, or max. This step is followed by a readout function that aggregates the node features h_v into a graph representation h_G. As succinctly described in [5], typical neighborhood aggregation schemes characteristic of graph neural networks can be described by two steps:

$$a_v^{(k)} = \text{AGGREGATE}(\{h_u^{(k-1)} \mid u \in \mathcal{N}_v\}) \quad \text{and} \quad h_v^{(k)} = \text{COMBINE}(h_v^{(k-1)}, a_v^{(k-1)}) \tag{1}$$

where $h_u^{(k)}$ is a representation of node $u \in \mathcal{V}$ at the output of the k^{th} iteration. For example, in graph convolutional networks the two steps are realized via mean pooling and a linear transformation [1]:

$$h_v^{(k)} = \sigma\left(\frac{1}{|\mathcal{N}_v^*|}\sum_{u\in\mathcal{N}_v^*} W^{(k)} h_u^{(k-1)}\right) \quad \text{with} \quad \mathcal{N}_v^* = \mathcal{N}_v \cup \{v\}$$

where σ is an activation function and $W^{(k)}$ is a weight matrix for the k^{th} iteration/layer.

After k iterations the representation of a node captures the information contained in its k-hop neighborhood [e.g., see the illustration of a vertex rooted sub-tree in 5, Figure 1]. The node features at the output of the last iteration are aggregated into a graph-level representation using a *readout* function. To enforce permutation invariant hypotheses, it is common to employ the standard pooling functions as readouts — sum, mean, or max. In the next section, we consider possible extensions that would allow for learning readout functions jointly with other parameters of graph neural networks.

2.1 Neural Readouts

Suppose that after completing a pre-specified number of neighborhood aggregation iterations, the resulting node features are collected into a matrix $H \in \mathbb{R}^{M\times D}$, where M is the maximal number of nodes that a graph can have in the dataset and D is the dimension of the output node embedding. For graphs with less than M vertices the padded values in H are set to zero. We also denote with a vector $h \in \mathbb{R}^{M\cdot D}$ the flattened (i.e., concatenated rows) version of the node feature matrix H.

Set Transformer Readouts. Recently, an attention-based neural architecture for learning on sets has been proposed in [10]. The main difference compared to the classical attention model proposed by Vaswani et al. [12] is the absence of positional encoding and dropout layers. The approach can be motivated by the desire to exploit dependencies between set items when learning permutation invariant hypotheses on that domain. More specifically, other approaches within the deep sets framework typically embed set items independently into a latent space and then generate a permutation invariant hypothesis by standard pooling operators (sum, max, or mean). As graphs can be seen as sets of nodes, we propose to exploit this architecture as a readout function in graph neural networks. For the sake of brevity, classical attention models are described in Appendix D and here we summarize the adaptation to sets. The set transformers take as input matrices with items/nodes as rows and generate graph representations by composing attention-based encoder and decoder modules:

$$\text{ST}(H) = \frac{1}{K}\sum_{k=1}^{K}\left[\text{DECODER}\left(\text{ENCODER}\left(H\right)\right)\right]_k \tag{2}$$

where $[\cdot]_k$ refers to computation specific to head k. The encoder-decoder modules are given by [10]:

$$\text{ENCODER}\left(H\right) \coloneqq \text{MAB}^n\left(H, H\right) \quad \text{and} \quad \text{DECODER}(Z) \coloneqq \text{FF}\left(\text{MAB}^m\left(\text{PMA}(Z), \text{PMA}(Z)\right)\right)$$
$$\text{where} \quad \text{PMA}(Z) \coloneqq \text{MAB}(s, \text{FF}(Z)) \quad \text{and} \quad \text{MAB}(X, Y) \coloneqq A + \text{FF}(A)$$
$$\text{with} \quad A \coloneqq X + \text{MULTI-HEAD}(X, Y, Y)\ .$$

Here, H denotes the node features after neighborhood aggregation and Z is the encoder output. The encoder is a chain of n classical multi-head attention blocks (MAB) without positional encoding and dropouts. The decoder component employs a seed vector s within a multi-head attention block to create an initial readout vector that is further processed via a chain of m self-attention modules and a feedforward projection block (FF).

Janossy Readouts. Janossy pooling was proposed in [11] with the goal of providing means for learning flexible permutation invariant hypotheses that in their core employ classical neural architectures such as recurrent and/or convolutional neural networks. The main idea is to process each permutation of set elements with such an architecture and then average the resulting latent representations. Additionally, one could also add a further block of feedforward or recurrent layers to process the permutation invariant latent embedding of a set. Motivated by this pooling function initially designed for node aggregation, we design a readout that is approximately permutation invariant. More specifically, we consider MLP and GRU as base architectures and sample p permutations of graph nodes. The Janossy readout then averages the latent representations of permuted graphs as follows:

$$\text{JANOSSY-MLP}(H) \coloneqq \frac{1}{p}\sum_{i=1}^{p} \text{MLP}(h_{\pi_i}) \quad \text{or} \quad \text{JANOSSY-GRU}(H) \coloneqq \frac{1}{p}\sum_{i=1}^{p} \text{GRU}(H_{\pi_i}), \tag{3}$$

where π_i is a permutation of graph nodes and h_{π_i} is the permuted and then flattened matrix H.

Plain Feedforward/Recurrent Readouts. Having proposed (approximate) permutation invariant readouts, we consider standard feedforward and recurrent neural architectures as well. Our MLP neural readout consists of a two-layer fully connected neural network (i.e., multi-layer perceptron) applied to the flattened node feature matrix H denoted with h:

$$\text{MLP}(H) \coloneqq \text{RELU}(\text{BN}_2(W_2 z_1 + b_2)) \quad \text{with} \quad z_1 = \text{RELU}(\text{BN}_1(W_1 h + b_1)) \tag{4}$$

where $W_1 \in \mathbb{R}^{(M \cdot D) \times d_1}$, $b_1 \in \mathbb{R}^{d_1}$, z_1 is the output of the first layer, $W_2 \in \mathbb{R}^{d_1 \times d_{\text{out}}}$, $b_2 \in \mathbb{R}^{d_{\text{out}}}$, d_1 and d_{out} are hyperparameters, BN_i is a batch normalization layer, and RELU is the rectified linear unit. In our experiments, we also apply Bernoulli dropout with rate $p = 0.4$ as the last operation within MLP. The GRU neural readout is composed of a single-layer, unidirectional gated recurrent unit (GRU, [13]), taking sequences with shape (M, D). We accept the input order on graph nodes as the order within the sequence that is passed to a GRU module. This recurrent module outputs a sequence with shape (M, d_{out}), as well as a tensor of hidden states. The graph-level representation created by this readout is given by the last element of the output sequence.

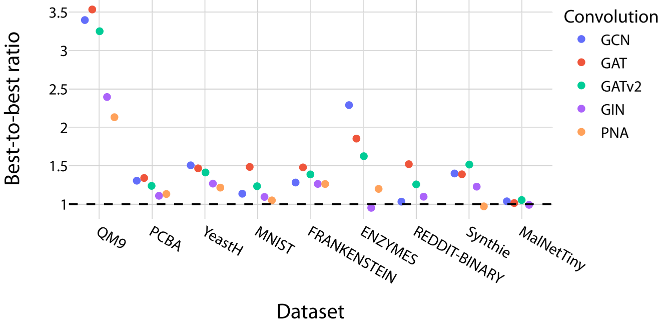
In contrast to typical set-based neural architectures that process individual items in isolation (e.g., deep sets), the presented adaptive readouts account for interactions between all the node representations generated by the neighborhood aggregation scheme. In addition, the dimension of the graph-level representation can now be disentangled from the node output dimension and the aggregation scheme.

3 Experiments

We perform a series of experiments[1] to evaluate the effectiveness of the adaptive and differentiable neural readouts presented in Section 2.1 relative to the standard pooling functions (i.e., sum, max, mean) used for the aggregation of node features into a graph-level representation. In our experiments, we rely on two performance metrics: R^2 for regression tasks and Matthews correlation coefficient (MCC) for classifications tasks. As outlined in prior work [14, 15], these metrics can be better at quantifying performance improvements than typical measures of effectiveness such as mean absolute/squared error, accuracy, and F_1 score. To showcase the potential for learning effective representation models over graph structured data, we use in excess of 40 datasets originating from different domains such as quantum mechanics, biophysics, bioinformatics, computer vision, social networks, synthetic graphs, and function call graphs. We focus on quantifying the difference in performance between the readouts relative to various factors such as: *i*) most frequently used neighborhood aggregations schemes, *ii*) the number of neighborhood aggregation iterations that correspond to layers in graph neural networks, *iii*) convergence rates measured in the number of epochs required for training an effective model, *iv*) different graph characteristics and domains from which the structured data originates, and *v)* the parameter budget employed by each of the neural

[1] The source code is available at https://github.com/davidbuterez/gnn-neural-readouts.

Figure 1: The performance of the best neural relative to the best standard readout on a collection of representative datasets from different domains. We use the ratio between the effectiveness scores (R^2 for QM9 and Matthew correlation coefficient otherwise), computed by averaging over five random splits of the data.



readouts. Moreover, we also evaluate a variety of neighborhood aggregation and readout schemes using large scale proprietary datasets with millions of compounds collected by primary screening assays. This is one of the first studies of a kind that demonstrates the behavior of standard graph neural networks on large scale datasets and illustrates some of the potential shortcomings that might be obfuscated by the small scale benchmarks that are typically used for evaluating their effectiveness in drug design tasks. In all of our experiments, we have used a node output dimension of 50, which gives rise to a graph embedding dimension of that size for the sum, mean, and max readouts. For the adaptive readouts, the dimensionality of the graph representation is a hyperparameter of the model, which was fixed to 64. For the sake of brevity, we present only a part of our analysis in this section and refer to the appendix for more detailed description of experiments and additional plots.

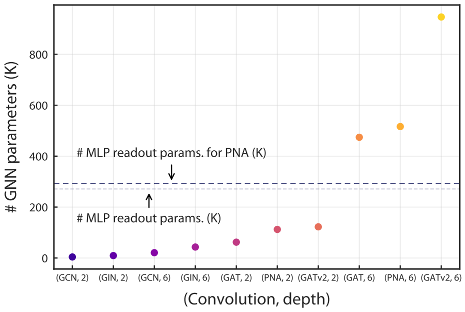
Neural vs standard readouts across different datasets and neighborhood aggregation schemes

The goal of this experiment is to evaluate the effectiveness of neural readouts relative to different datasets (39 in total, Appendix A), graph characteristics, and neighborhood aggregation schemes or convolutional operators. To this end, we opt for graph neural networks with two layers and compare neural to standard readouts across different graph convolutional operators: GCN [1], GAT [3], GATV2 [16], GIN [5], and PNA [17]. A detailed experimental setup (including loss functions, reporting metrics, and other details relevant for reproducibility) has been provided in Appendices B and F.

Figure 1 summarizes the result of this experiment over 9 representative datasets (please see Appendix G, Figures 1 to 3 for the results on the remaining 30 datasets, including other metrics). The figure depicts the ratio between the best neural and best standard readouts for each evaluated configuration (i.e., pair of dataset, convolution). We observe a considerable uplift in the performance on the majority of datasets. More specifically, in regression tasks we measure the performance using the R^2 score and on 36 configurations out of the possible 45 (i.e., 80% of time) there is an improvement (with ratio > 1) as a result of employing neural readouts. In datasets where standard readouts fare better than the neural ones, the relative degradation is minor (i.e., less than 5%). In classification tasks, we use the MCC to measure the performance of graph neural networks and again observe an improvement as a result of employing neural readouts on 93 out of 147 (dataset, convolution) configurations ($\approx 63\%$ of time). We also observe that on 54 configurations where standard readouts are more effective than the neural ones that the relative degradation is below 10% relative. We note that three models also failed to complete due to memory issues when using PNA, leading to a total of 147 configurations for the classification tasks (Appendix T). The minimum observed ratio between neural and standard readouts was 0.79.

It is also worth noting that neural readouts come with hyperparameters that were not tuned/cross-validated in our experiments. For example, set transformer readouts can be configured by specifying the number of attention heads and latent/hidden dimensions. We have, throughout our experiments, followed standard practices and selected such hyperparameters to be powers of two, tailored to the dataset size (more details can be found in Appendix F, Table 4). This was also, in part, motivated by previous knowledge from bio-affinity prediction tasks with graph neural networks. Thus, it is likely that the performance can be further improved by hyperparameter tuning. We also emphasize that for this experiment the architecture is kept fixed across datasets (number of graph layers, hidden

Figure 2: The panel on the left illustrates the parameter budget of GNNs that does not account for the readouts, while varying the layer type and depth (QM9 dataset). The number of parameters for the MLP readout is represented using dashed lines parallel to the x-axis (slightly higher when using PNA as the output node dimension must be divisible by the tower hyperparameter). The panel on the right compares the performance of graph neural networks with set transformers (ST) and plain MLPs as readouts, while varying the number of parameters in the readout layer. This illustrative example has been obtained using the ENZYMES dataset and demonstrates that the effectiveness is not aligned with the number of trainable parameters in the readout layer but the type of architecture realizing it. For example, the ST 1–MAB model employs a single MAB-block encoder and decoder, being simpler in terms of the number of parameters than ST COMPLEX and more complex than ST MINIMAL (Appendix C). The MLP configurations are reported using the format MLP (d_1, d_{out}) (Appendix F).



Readout	# Params.	Avg. MCC
ST MINIMAL	365K	0.35
ST 1–MAB	628K	0.34
ST COMPLEX	1154K	0.36
MLP (32, 32)	130K	0.41
MLP (64, 32)	260K	0.44
MLP (128, 64)	525K	0.40
MLP (64, 128)	267K	0.47

dimensions, etc.), while only varying the graph layer type and readout layer. In addition, over-smoothing is a known problem for vanilla graph neural networks. To avoid any hidden contribution from over-smoothing correction techniques, we opted for shallower architectures. We also note that two-layer graph neural networks performed well on tasks such as bio-affinity prediction on the $1+$ million scale datasets (i.e., can be sufficiently expressive). To validate that additional expressiveness due to more layers can be successfully exploited by adaptive readouts, we performed a separate suite of experiments where we varied the depth (see the next experiment/section). We conclude the discussion of this experiment with an insight into the trainable parameter budgets for a selection of neural readouts. For MLP, with the exception of GCN and GIN which use an extremely small number of trainable weights, the parameter budget/count is on the same scale as the rest of the graph neural networks (see the left panel in Figure 2). Furthermore, simply increasing the number of parameters does not necessarily improve the performance (see the right panel in Figure 2, ST vs MLP rows).

Neural vs standard readouts relative to the number of neighborhood aggregations iterations

The goal of this experiment is two-fold: *i*) assessment of the effectiveness of neural readouts relative to the depth of graph neural networks, and *ii*) validation of the observations in the experiments with two layer graph neural networks, i.e., the improvements characteristic of neural readouts are not due to underfitting that can be caused by employing a small number of neighborhood aggregation iterations along with standard readouts. We perform the experiment with various graph convolutional operators using datasets with different number of instances, ranging from 600 to $132,480$ graphs on ENZYMES and QM9, respectively. In these experiments, the only variable graph neural network hyperparameter is the number of neighborhood aggregation iterations (i.e,. depth). We have also performed this experiment on one of the proprietary bio-affinity datasets with 1.5 million graphs (see Appendix N, Figure 19). Figure 3 summarizes the results for the QM9 experiment (see Appendix H, Figure 4 for ENZYMES). The trends seen for standard readouts are mirrored for the neural ones as the number of neighborhood iterations increases, i.e., deeper graph neural networks can lead to more expressive models, irrespective of the readout type.

Convergence of training algorithms for graph neural networks relative to readouts

As outlined in Section 1, it might be challenging to learn a permutation invariant hypothesis over graphs using simple and non-adaptive readout functions such as sum, mean, or max. Here, we argue that such functions might be creating a tight link between the resulting graph-level representations and the computed node features, in the sense that: *i*) it takes a long time for the graph-level representation to adjust to the prediction task and *ii*) it is difficult for the graph-level representation to diverge from the learned node features and adapt to the target property. To validate this hypothesis, we recorded the graph representations for a random molecule from the QM9 dataset in each training epoch (for multiple convolutions and readouts). We computed the Euclidean distances between the initial graph

Figure 3: Increasing the number of neighborhood aggregation iterations or convolutional layers has different effects on QM9. The trends observed for the standard readouts are mirrored for the neural ones, particularly for the most powerful one on this dataset (SET TRANSFORMER). For GCN, GAT, and GATV2, the performance improves as the depth is increased to 6 layers and drops afterwards. GIN is generally stable relative to the number of layers, while PNA has an initial performance improvement (up to 3, 4 layers) and then plateaus.

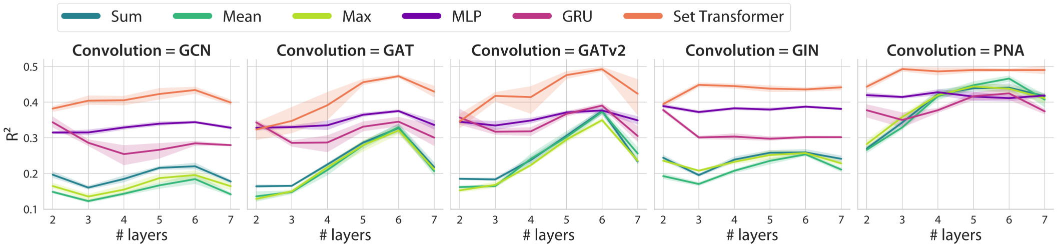


Table 1: R^2 (mean ± standard deviation) on QM9, using the ST MINIMAL and ST COMPLEX architectures.

Aggregator	# Heads	GCN	GAT	GATv2	GIN	PNA
ST MINIMAL	1	0.20 ± 0.02	0.17 ± 0.01	0.20 ± 0.01	0.22 ± 0.01	0.27 ± 0.01
	4	0.27 ± 0.01	0.23 ± 0.02	0.27 ± 0.02	0.32 ± 0.01	0.38 ± 0.02
	8	0.35 ± 0.01	0.24 ± 0.04	0.29 ± 0.01	0.35 ± 0.01	0.43 ± 0.01
	12	0.37 ± 0.01	0.26 ± 0.03	0.27 ± 0.03	0.37 ± 0.01	0.43 ± 0.01
ST COMPLEX	1	0.18 ± 0.02	0.16 ± 0.02	0.20 ± 0.01	0.24 ± 0.01	0.28 ± 0.01
	4	0.35 ± 0.01	0.27 ± 0.02	0.29 ± 0.02	0.35 ± 0.00	0.42 ± 0.01
	8	0.38 ± 0.01	0.30 ± 0.04	0.32 ± 0.02	0.39 ± 0.01	0.44 ± 0.01
	12	0.37 ± 0.02	0.32 ± 0.02	0.33 ± 0.02	0.40 ± 0.01	0.45 ± 0.01

embedding (i.e., the first epoch) and all subsequent epochs (Appendix K, Figure 11), as well as between consecutive epochs (Appendix K, Figure 12). Our empirical results indicate that GNNs with standard readouts take hundreds of epochs to converge (500 to 1,000), with minor changes in the graph representation from one epoch to another. In contrast to this, the models employing neural readouts converge quickly, typically in under 100 epochs. Moreover, the learned representations can span a larger volume, as shown by the initial and converged representations, which are separated by distances that are orders of magnitude larger than for the standard readouts.

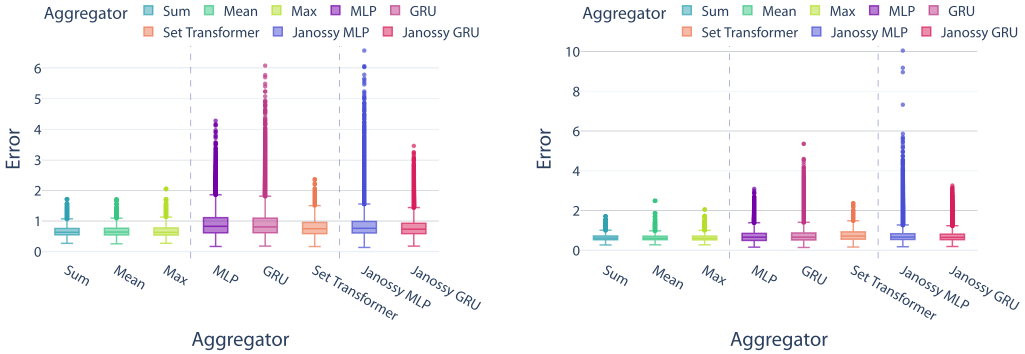
The importance of (approximate) permutation invariance in adaptive readouts

The goal of this experiment is to obtain an insight into the effects of adaptive readouts that give rise to permutation invariant hypothesis spaces on the network's ability to learn a target concept. To this end, we exploit the modularity of SET TRANSFORMERS and consider architectures with 1 to 12 attention heads, as well as a different number of attention blocks: an ST MINIMAL model with one MAB in the encoder and no MABs in the decoder, and ST COMPLEX with two MABs in both the encoder and decoder (for 2-layer GNNs). Table 1 provides a summary of the results for this type of readouts on the QM9 dataset (detailed results can be found in the appendix). With a small number of attention heads (1 and 4), all models with SET TRANSFORMER readouts are able to outperform the ones with standard pooling functions. However, over different convolutions the models with few attention heads are outperformed by the ones with MLP and GRU readouts that do not enforce permutation invariance. Increasing the number of heads to 8 or 12 leads to the best performance on this dataset for all graph convolutions. However, the relative improvement gained by increasing the number of attention heads beyond 4 is generally minor, as is the uplift gained by adding a self-attention block. We also evaluated the impact of enforcing approximate permutation invariance by Janossy readouts on the QM9 dataset (Appendix T, Table 27). The Janossy variants presented in Section 2.1 outperform the three standard readouts in both mean absolute error and R^2, but they score lower than the other neural readouts.

Robustness to node permutations for non-permutation invariant readouts

The objective of this experiment is to evaluate the stability of readouts that either do not enforce permutation invariance of the hypothesis space (i.e., MLP and GRU readouts) or do so only approximately (i.e., JANOSSY readouts). To this end, we generate 50 random node permutations (relative to the order induced by the graphs generated from the canonical SMILES) for 50 randomly selected molecules from the QM9 dataset and run these through the previously trained models with two

Figure 4: A summary of the error distributions for predictions made on random permutations of 50 randomly selected molecules from the QM9 dataset. The error is computed as the absolute difference between the predicted and target labels. The models are fully trained, two-layer graph neural networks. Due to permutation invariance, the predictions made for a given molecule are identical for sum/mean/max, regardless of the permutation. The variance for these readouts is a result of the differences in the predicted labels for the considered 50 molecules. Panels **(a)** and **(b)** reflect different strategies of generating node permutations.



(a) For each molecule, we generate 50 different graphs using random permutations of the nodes originating from the canonical SMILES representation.

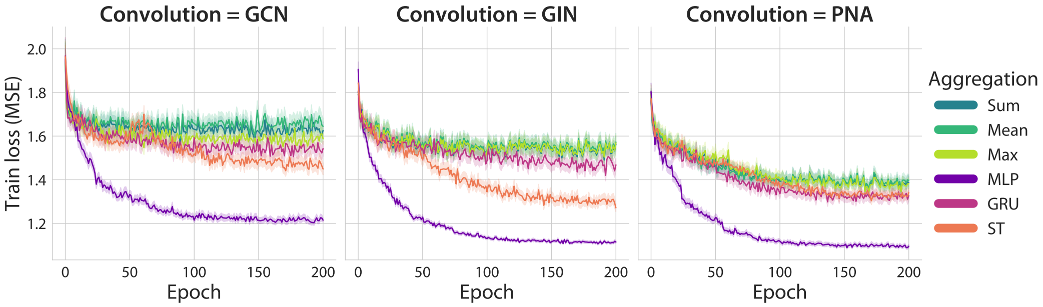
(b) For each molecule, we generate graphs corresponding to different non-canonical SMILES. The number varies from 20 to $1,000$ (per-molecule).

layers (see above for details, neural vs standard readouts). As all graph molecular representations originate from a canonical SMILES string (simplified molecular-input line-entry system, [18]), we also generate permuted graphs by using non-canonical SMILES. For the latter, we have applied repeated sampling of permuted SMILES using RDKit until the total number converged, resulting in permuted molecular graphs. Depending on the molecule, it is possible to generate as little as 20 different representations or as many as $1,000$ (all the resulting graphs are used in the analysis). Permutations due to non-canonical SMILES represent a subset of arbitrary permutations that are specific to the chemistry domain. For each molecule and random permutation, we computed the error as the absolute difference between the predicted and target labels. Figures 4a and 4b summarize the results of this experiment and show that the plain readouts (i.e., MLP and GRU) can be negatively affected by some random permutations. Interestingly, the error for MLP is greatly accentuated on certain QM9 prediction tasks, such as U0, U298, H298, and G298, while the error is very similar to sum, mean, and max on tasks such as CV, ZPVE, and R2 (see Appendix J, Figure 9 for more details). The Janossy readouts are trained not only on the original fixed-order graphs but also on 25 random node permutations computed during training (i.e., a different stage compared to the evaluation presented here). We observe that the JANOSSY GRU readout improves upon the plain GRU, leading to a distribution that is more similar to the SET TRANSFORMER readout, with a reduced number of outliers. In contrast, the JANOSSY-MLP readout appears to be performing the worst in terms of robustness to node permutations. A breakdown of results over different neighborhood aggregation schemes and convolutions can be found in Appendix J, Figures 6 and 7. In summary, all neural readouts exhibit significantly reduced errors for the permutations originating from randomly generated non-canonical SMILES compared to the arbitrary node permutations, despite being trained only using the canonical representations. Moreover, we encounter many different (permuted) molecular representations that attain the minimal error values for MLP readouts (see Appendix J, Figure 8).

The effectiveness of graph neural networks and readouts on multi-million graph datasets

In this experiment, we extend our empirical analysis with three AstraZeneca high-throughput screening assays against varied protein targets. Each dataset consists of 1 to 2 million molecular measurements, such that each molecule is associated with a single label (scalar) corresponding to its activity value (see Appendix A, Table 1 for details). We trained both non-variational and guided variational graph autoencoder [19, 20] models (denoted by GNN and VGAE, respectively) for 200 epochs on the entirety of the available data for each dataset, with the goal of assessing the ability of graph neural networks to learn expressive graph representations at this scale (Figure 5, with the other 5 models in Appendix M). Our analysis considers the performance relative to different neighborhood aggregation schemes or convolutional operators as well as the total number of such iterations that correspond to depth in graph neural networks (separately, see Appendix N, Figure 19). Figure 5 summarizes the results of this experiment and indicates that the models using standard non-adaptive readouts (sum,

Figure 5: Train loss for the VGAE models trained on a proprietary dataset with ≈ 1 million molecular graphs.



mean, or max) generally struggle to model molecular data at the $1+$ million scale, as reflected in the plateaued training losses for each of the models. Depending on the dataset and the neighborhood aggregation scheme, the MLP readout significantly outperforms the other readout functions, with a train loss that rapidly decreases in the first 50 epochs. The SET TRANSFORMER readout is the second-best choice, converging with a slightly a slower rate and occasionally diverging. The GRU readouts offer only a slight improvement compared to the standard readout functions. When training both variational and non-variational models with deeper architectures (2, 3, and 4 layers, excluding μ- and σ-layers for the VGAE), we do not observe significant benefits introduced by additional iterations of neighborhood aggregation schemes. This is in line with the study that introduced these datasets [20], which also evaluated multiple convolutional operators and numbers of such iterations/layers. Instead, as indicated previously, the largest benefits are generally associated with more powerful neighborhood aggregation schemes. The results are further supported by the training metrics (MAE, R^2) after 200 epochs (see also Appendix O, Tables 8 to 13), which provide an insight into the ability of graph neural networks to fit signals on $1+$ million scale datasets. For example, our results for the VGAE GCN model trained on the proprietary dataset with ≈ 1 million graphs show that neural readouts lead to an improvement in the R^2 score from 0.33 to 0.78, with ≈ 1.5 million graphs from 0.07 to 0.64, and with ≈ 2 million graphs from 0.06 to 0.52.

4 Discussion

We have presented an extensive evaluation of graph neural networks with adaptive and differentiable readout functions and various neighborhood aggregation schemes. Our empirical results demonstrate that the proposed readouts can be beneficial on a wide variety of domains with graph structured data, as well as different data scales and graph characteristics. Overall, we observed improvements in over two thirds of the evaluated configurations (given by pairs consisting of datasets and neighborhood aggregation schemes), while performing competitively on the remaining ones. Moreover, we have empirically captured and quantified different aspects and trade-offs specific to adaptive readouts. For instance, the effectiveness of adaptive readouts that do not enforce permutation invariance of hypothesis spaces (MLP and GRU) indicates that it might be possible to relax this constraint for certain tasks. A primary candidate for relaxation are molecular tasks, which are also one of the most popular application domains for graph neural networks. Molecules are typically presented in the canonical form, a strategy also adopted by popular open-source frameworks such as RDKit. Thus, the graph that corresponds to any given molecule comes with a fixed vertex ordering when generated from the canonical SMILES. Our analysis suggests that neural readouts trained on canonical representations can learn chemical motifs that are applicable even to non-canonical inputs, or in other words, generally applicable chemical knowledge. It should be noted here that the canonical representations differ greatly even for extremely similar molecules (see also Appendix R), such that it is improbable that the graph neural networks are learning simple associations based on position or presence of certain atoms. Instead, it might be the case that the networks can learn certain higher-level chemical patterns that are strictly relevant to the task at hand.

We have also discussed possible domain-specific interpretations for the effectiveness of models with adaptive readouts on some tasks. For instance, certain molecular properties tend to be approximated well by neural readouts, while others remain more amenable to standard pooling functions such as sum. Chemically, properties such as the internal energy, enthalpy, or free energy are generally considered additive (e.g., can be approximated by a sum of pairwise bond energies) and extensive (increasing almost linearly with the number of atoms). Such properties are a good fit for standard readouts. Other properties, such as the highest occupied molecular orbital and lowest unoccupied molecular orbital (HOMO and LUMO, respectively) tend to be localized and are considered non-additive, such that a single atom can potentially completely alter the property, or not influence it at all. Popular problems such as bio-affinity prediction are also regarded as highly non-linear. Overall, this interplay suggests hybrid readouts for future research, where certain properties would be learned by functions such as sum, while others are left to more flexible neural readouts.

Regarding practical details such as the choice of the most suitable adaptive readout function for a given dataset, our empirical results indicate that larger and more complex (relative to the number of nodes per graph and dimension of node features) regression tasks see more pronounced performance improvements with adaptive readouts (based on statistically significant results from linear regression models detailed in Appendix Q). We were, however, unable to observe a similar pattern for the considered classification tasks. Thanks to its potential for composing highly expressive neural architectures, SET TRANSFORMER is likely better suited for larger datasets. However, graph neural networks with that readout function tend to occasionally experience divergence on very large datasets or deep architectures (6+ layers), which can most likely be fully resolved with parameter tuning, especially the latent/hidden dimension of the attention mechanism. An avenue that might be promising for further study is pre-training readout networks, such that they can be quickly deployed on related tasks and fine-tuned. One starting point could be pre-training on large molecular databases, such as subsets of GDB-17 [21] with inexpensive to compute molecular tasks (generated with RDKit, for example) as prediction targets, or unsupervised variations.

When it comes to related approaches, the majority of recent efforts have been focused on neighborhood aggregation schemes. This step also requires permutation invariance and it is interesting that a related work by Hamilton et al. [22] has considered relaxation to that constraint and employed an LSTM neural network to aggregate neighborhoods and produce node features. Along these lines, Murphy et al. [11] introduced Janossy pooling, a permutation-invariant pooling technique for neighborhood aggregation, designed for node classification tasks. Perhaps the most related to our direction and readouts is the concurrently developed work by Baek et al. [23] on graph multi-set transformers, i.e., a multi-head attention model based on a global pooling layer that models the relationship between nodes by exploiting their structural dependencies. For the purpose of measuring and fixing the over-squashing problem in graph neural networks, Alon and Yahav [24] proposed a fully-adjacent layer (each pair of nodes is connected by an edge), which greatly improved performance and resembles our use of the MLP readout. Prior work has also considered tunable ℓ_p pooling [25] and more restrictive universal readouts based on deep sets [26]. However, neither of these approaches offers a comprehensive empirical evaluation at the scale provided here.

As adaptive readouts introduce a new differentiable component to graph neural networks, future studies might focus on analyzing properties such as transferability and interpretability. In our empirical study, we did not consider such experiments due to conceptual and practical differences. Conceptually, one of the main motivating factors for studying transferability is a scenario where the graph (network) size changes over time. This is typically encountered in recommendation systems or knowledge graphs which are not considered in our paper. Regarding the graph-to-graph transferability, there are domain-specific particularities that need to be considered. For example, learning on small graphs and transferring to larger graphs is not often required in chemical tasks, as most chemical regression benchmarks and real-world applications use only very small organic molecules (e.g., < 30 atoms or nodes for QM9). There is also the requirement of selectivity, where an active molecule should bind only to a selected target and possible issues can arise with transferring a notion of similarity over the space of molecules that encodes activity to a completely different target. Moreover, there have been reports where the impact of learning (with graph neural networks) certain transferable chemical substructures (scaffolds) was not beneficial [27]. Practically, transferability has been most often studied with node-level tasks [28], while here we focus on graph-level predictions. Overall, we believe that studying the influence of adaptive readouts on transferability is interesting for future studies. Regarding the interpretability, we have in this work focused on allowing for more flexibility in neural readouts (Appendix K) and the structuring effect on the learned latent space, possibly making it amenable to clustering and other downstream tasks (Appendix L).

Acknowledgments: We are grateful for access to the AstraZeneca Scientific Computing Platform. David Buterez has been supported by a fully funded PhD grant from AstraZeneca.

References

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

[2] William L. Hamilton, Rex Ying, and Jure Leskovec. Representation learning on graphs: Methods and applications. *IEEE Data Engineering Bulletin*, 40(3):52–74, 2017.

[3] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *International Conference on Learning Representations*, 2018.

[4] Michael M. Bronstein, Joan Bruna, Taco Cohen, and Petar Velickovic. Geometric deep learning: Grids, groups, graphs, geodesics, and gauges. *CoRR*, abs/2104.13478, 2021.

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

[6] Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Russ R Salakhutdinov, and Alexander J Smola. Deep sets. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[7] Edward Wagstaff, Fabian Fuchs, Martin Engelcke, Ingmar Posner, and Michael A. Osborne. On the Limitations of Representing Functions on Sets. In *Proceedings of the 36th International Conference on Machine Learning*, pages 6487–6494. PMLR, 2019.

[8] Laszlo Babai and Ludik Kucera. Canonical labelling of graphs in linear average time. In *20th Annual Symposium on Foundations of Computer Science (sfcs 1979)*, pages 39–46, 1979.

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

[10] Juho Lee, Yoonho Lee, Jungtaek Kim, Adam Kosiorek, Seungjin Choi, and Yee Whye Teh. Set transformer: A framework for attention-based permutation-invariant neural networks. In *Proceedings of the 36th International Conference on Machine Learning*, pages 3744–3753, 2019.

[11] Ryan L. Murphy, Balasubramaniam Srinivasan, Vinayak Rao, and Bruno Ribeiro. Janossy pooling: Learning deep permutation-invariant functions for variable-size inputs. In *International Conference on Learning Representations*, 2019.

[12] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[13] Kyunghyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder–decoder approaches. In *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, pages 103–111, Doha, Qatar, October 2014. Association for Computational Linguistics. doi: 10.3115/v1/W14-4012.

[14] Davide Chicco and Giuseppe Jurman. The advantages of the matthews correlation coefficient (mcc) over f1 score and accuracy in binary classification evaluation. *BMC Genomics*, 21(1):6, Jan 2020. ISSN 1471-2164.

[15] Davide Chicco, Matthijs J. Warrens, and Giuseppe Jurman. The coefficient of determination r-squared is more informative than smape, mae, mape, mse and rmse in regression analysis evaluation. *PeerJ. Computer science*, 7:e623–e623, Jul 2021. ISSN 2376-5992.

[16] Shaked Brody, Uri Alon, and Eran Yahav. How attentive are graph attention networks? In *International Conference on Learning Representations*, 2022.

[17] Gabriele Corso, Luca Cavalleri, Dominique Beaini, Pietro Liò, and Petar Veličković. Principal neighbourhood aggregation for graph nets. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 13260–13271. Curran Associates, Inc., 2020.

[18] David Weininger. Smiles, a chemical language and information system. 1. introduction to methodology and encoding rules. *Journal of Chemical Information and Computer Sciences*, 28 (1):31–36, 1988. doi: 10.1021/ci00057a005.

[19] Thomas N Kipf and Max Welling. Variational graph auto-encoders. *NIPS Workshop on Bayesian Deep Learning*, 2016.

[20] David Buterez, Jon Paul Janet, Steven Kiddle, and Pietro Liò. Multi-fidelity machine learning models for improved high-throughput screening predictions. *ChemRxiv*, 2022.

[21] Lars Ruddigkeit, Ruud van Deursen, Lorenz C. Blum, and Jean-Louis Reymond. Enumeration of 166 billion organic small molecules in the chemical universe database gdb-17. *Journal of Chemical Information and Modeling*, 52(11):2864–2875, 2012.

[22] William L. Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*. Curran Associates Inc., 2017.

[23] Jinheon Baek, Minki Kang, and Sung Ju Hwang. Accurate learning of graph representations with graph multiset pooling. In *ICLR*, 2021.

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

[25] Giovanni Pellegrini, Alessandro Tibo, Paolo Frasconi, Andrea Passerini, and Manfred Jaeger. Learning aggregation functions. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence*. International Joint Conferences on Artificial Intelligence Organization, 2021.

[26] Nicolò Navarin, Dinh Van Tran, and Alessandro Sperduti. Universal readout for graph convolutional neural networks. In *2019 International Joint Conference on Neural Networks*, 2019.

[27] Miyuki Sakai, Kazuki Nagayasu, Norihiro Shibui, Chihiro Andoh, Kaito Takayama, Hisashi Shirakawa, and Shuji Kaneko. Prediction of pharmacological activities from chemical structures with graph convolutional neural networks. *Scientific Reports*, 11(1):525, Jan 2021. ISSN 2045-2322. doi: 10.1038/s41598-020-80113-7.

[28] Luana Ruiz, Luiz Chamon, and Alejandro Ribeiro. Graphon neural networks and the transferability of graph neural networks. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1702–1712. Curran Associates, Inc., 2020.

[29] Zhenqin Wu, Bharath Ramsundar, Evan N. Feinberg, Joseph Gomes, Caleb Geniesse, Aneesh S. Pappu, Karl Leswing, and Vijay Pande. Moleculenet: a benchmark for molecular machine learning. *Chem. Sci.*, 9:513–530, 2018.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes] Please see Sections 3 and 4.
   (b) Did you describe the limitations of your work? [Yes] Covered in Sections 1 and 4.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] Please see Section 4 and Appendix A.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes] Detailed in Appendix A.
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] The code and instructions are available on GitHub.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] In Appendices B, C and F and the associated code.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] Please see Appendix P.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [Yes] where available. Please see Appendix A, Table 2 and Table 3. Old or public domain datasets (originating from PubChem) are exceptions. [No] for the three proprietary molecular datasets.
   (c) Did you include any new assets either in the supplemental material or as a URL? [No]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [No] for the established open source datasets. [Yes] Consent was given to publish results on the proprietary data.
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [Yes] Please see Appendix A.
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

A Summary of the used datasets

All of the public datasets were previously published, either as graph representation learning benchmarks or new datasets for specific graph tasks. The datasets cover a multitude of domains: quantum mechanics, biophysics, bioinformatics, computer vision, social networks, synthetic graphs, and function call graphs. As such, the datasets do not include any personally identifiable information or offensive content. This claim is based both on manual inspection and previous peer-review of the associated publications. Furthermore, no research was performed with human subjects as part of our study. The public benchmarks are listed in Appendix A, Tables 2 and 3. The public and proprietary bio-affinity (high-throughput screening) datasets are listed in Appendix A, Table 1.

Appendix Table 1: Summary of the bio-affinity high-throughput screening datasets.

Availability	Source	Dataset	Size	Splits	Task type	# Tasks
Public	PubChem	AID1949	98,472	No	Regression	1
		AID449762	311,910			
		AID602261	343,811			
Private	Pharmaceutical company		1,013,581	No	Regression	1
			1,482,258			
			1,962,638			

B Experimental design and reporting

Before training each model, a fixed random seed at the beginning of training is set (`pytorch_lightning.seed_everything(0)`). Any convolution-specific hyperparameters (such as the number of attention heads and dropout values for GAT and GATV2) were chosen as reasonable defaults and frozen. As the neural readouts introduce new hyperparameters for the overall GNN architecture, these are also set to reasonable defaults depending on the dataset type and size (Appendix B, Table 4), but are not part of any hyperparameter optimization procedure.

For datasets that did not explicitly provide train, validation, and test sets, we applied an 80%/10%/10% split, for five different times on random permutations of the datasets. This procedure was applied on all MoleculeNet datasets, as well as the TUDataset benchmarks. We did not apply any custom splits on MNIST, CIFAR10, and ZINC. The seeds used for the five random permutations are available in **Supplementary File 1** (available on GitHub) and can be directly used with the provided source code. All models are trained with an early stopping mechanism set to a patience value of 30.

Almost all datasets are completely loaded in memory, with the exception of a few complex datasets (such as the REDDIT datasets). If the default batch size of 32 was too large for such datasets, the maximum batch size that allowed the models to be trained on a GPU with 24GB of VRAM was used.

The loss functions used within the deep learning models are set to standard choices, such as mean squared error (MSE) for regression datasets, binary cross-entropy for binary classification datasets and cross-entropy for multi-class datasets. We also generally tried to use the recommended loss functions for the MoleculeNet datasets according to the original publication, in particular using the mean absolute error (MAE) function for certain datasets [29].

The metrics chosen to report the model performance depend on the task type (regression or classification) and the number of tasks (or classes for classification). For binary classification tasks, we report the area under the receiver operating characteristic curve (AUROC) and the Matthews correlation coefficient (MCC), while for classification tasks with more than 2 classes only the MCC is reported. The MCC was recently reported to be a more helpful metric compared to popular choices such as the accuracy or the F_1 score and is considered one of the best summaries of the confusion matrix [14].

For regression tasks, we report the MAE and the coefficient of determination (R^2) for all datasets. R^2 was also recently reported as a regression metric that is more informative compared to the traditional choices of MAE, MSE, RMSE, and others [15].

Appendix Table 2: Summary of all the used benchmarks (datasets), including domain, datasets statistics, random splitting procedures, and source. DeepChem (DC) and PyTorch Geometric (PyG) are released under the MIT license. Several molecular datasets originate from PubChem (public domain). ENZYMES uses CC BY 4.0. reddit_threads and twitch_egos use GPL-3.0. COLORS and TRIANGLES use ECL-2.0. CIFAR10, MNIST, and ZINC use MIT.

Collection	Domain	Dataset	Type	# Tasks	Size	Avg. nodes	Avg. edges	Node attr.	Splits	Source
MoleculeNet	Quantum Mechanics	QM9	Regr.	12	132,480	17.99	37.15	30	5 random	DC
		QM8	Regr.	12	21,747	16.09	32.81	30	5 random	DC
		QM7	Regr.	1	6,834	15.54	30.39	30	5 random	DC
	Physical Chemistry	ESOL	Regr.	1	1,127	13.30	27.38	30	5 random	DC
		FreeSolv	Regr.	1	639	8.76	16.85	30	5 random	DC
		Lipophilicity	Regr.	1	4,200	27.04	59.00	30	5 random	DC
	Biophysics	PCBA	Cls.	128	437,918	25.97	56.22	30	5 random	DC
		HIV	Cls.	1	41,127	25.51	54.94	30	5 random	DC
		BACE	Cls.	1	1,513	34.09	73.72	30	5 random	DC
		BACE	Regr.	1	1,513	34.09	73.72	30	5 random	DC
	Physiology	BBBP	Cls.	1	2,039	24.06	51.91	30	5 random	DC
		SIDER	Cls.	27	1,396	34.36	72.29	30	5 random	DC
TUDataset	Bioinformatics	ENZYMES	Cls.	6	600	32.63	62.14	18	5 random	PyG
		PROTEINS_full	Cls.	2	1,113	39.06	72.82	29	5 random	PyG
	Computer Vision	COIL-DEL	Cls.	100	3,900	21.54	54.24	2	5 random	PyG
		COIL-RAG	Cls.	100	3,900	3.01	3.02	64	5 random	PyG
		Cuneiform	Cls.	30	267	21.27	44.80	3	5 random	PyG
	Social Networks	github_stargazers	Cls.	2	12,725	113.79	234.64	-	5 random	PyG
		IMDB-BINARY	Cls.	2	1,000	19.77	96.53	-	5 random	PyG
		REDDIT-BINARY	Cls.	2	2,000	429.63	497.75	-	5 random	PyG
		REDDIT-MULTI-12K	Cls.	11	11,929	391.41	456.89	-	5 random	PyG
		reddit_threads	Cls.	2	203,088	23.93	24.99	-	5 random	PyG
		twitch_egos	Cls.	2	127,094	29.67	86.59	-	5 random	PyG
		TWITTER-Real-Graph-Partial	Cls.	2	144,033	4.03	4.98	-	5 random	PyG
	Synthetic	COLORS-3	Cls.	11	10,500	61.31	91.03	4	5 random	PyG
		SYNTHETIC	Cls.	2	300	100.00	196.00	1	5 random	PyG
		SYNTHETICnew	Cls.	2	300	100.00	196.25	1	5 random	PyG
		Synthie	Cls.	4	400	95.00	172.93	15	5 random	PyG
		TRIANGLES	Cls.	10	45,000	20.85	32.74	-	5 random	PyG
GNNBenchmarkDataset	Computer Vision	MNIST	Cls.	10	55,000	70.60	564.50	3	Provided	PyG
		CIFAR10	Cls.	10	45,000	117.60	941.20	5	Provided	PyG
ZINC	Drug-like molecules	ZINC	Regr.	1	249,456	23.15	49.80	1	Provided	PyG

Appendix Table 3: Summary of all the used benchmarks (datasets), including domain, datasets statistics, random splitting procedures, and source. PyG, PyTorch Geometric.

Collection	Domain	Dataset	Type	# Tasks	Size	Avg. nodes	Avg. edges	Node attr.	Splits	Source
TUDataset	Small molecules	AIDS	Cls.	2	2,000	15.69	16.20	4	5 random	PyG
		alchemy_full	Regr.	12	202,579	10.10	10.44	3	5 random	PyG
		FRANKENSTEIN	Cls.	2	4,337	16.90	17.88	780	5 random	PyG
		Mutagenicity	Cls.	2	4,337	30.32	30.77	-	5 random	PyG
		MUTAG	Cls.	2	188	17.93	19.79	-	5 random	PyG
		YeastH	Cls.	2	79,601	39.44	40.74	-	5 random	PyG
MalNetTiny	Function call graphs	MalNetTiny	Cls.	5	5,000	1410.31	2859.94	-	5 random	PyG

To simplify reporting the results, for multi-label datasets the appropriate metrics are computed between flattened representations of the predictions and the ground truth values.

C Set Transformer aggregator architectures

Our default Set Transformer architecture, simply referred to as SET TRANSFORMER for the majority of the paper, and ST COMPLEX in Figure 2 and Appendix T, Table 1, uses multiple SAB blocks, where a SAB block is defined as $\textsc{sab}(A) = \textsc{mab}(A, A)$:

$$\textsc{encoder}(A) \coloneqq \textsc{sab}^2(A) \quad \text{and} \quad \textsc{decoder}(C) \coloneqq \textsc{ff}(\textsc{sab}^2(\textsc{pma}_k(C))) \tag{5}$$

We also evaluate a variation termed ST MINIMAL, with an architecture reduced to:

$$\textsc{encoder}(A) \coloneqq \textsc{sab}(A) \quad \text{and} \quad \textsc{decoder}(C) \coloneqq \textsc{ff}(\textsc{pma}_k(C)) \tag{6}$$

D Summary of multi-head attention

First of all, we recapitulate the original definition of attention, which works by initializing three learnable matrices, often called Q for *queries*, K for *keys* and V for *values*. For self attention, $Q = K = V$. The attention computation is then defined as

$$\textsc{attention}(Q, K, V) \coloneqq \text{softmax}(\frac{QK^\top}{\sqrt{d_k}})V \tag{7}$$

where softmax is defined element-wise as $\text{softmax}(\mathbf{x})_i \coloneqq \frac{\exp(\mathbf{x}_i)}{\sum_j \exp(\mathbf{x}_j)}$

It is often beneficial to perform multiple attention computations concurrently, using different parameters (weights W), called multi-head attention with h heads

$$\textsc{multi-head}(Q, K, V) \coloneqq \text{Concatenate}(\text{head}_1, \dots, \text{head}_h)W^O \tag{8}$$

$$\text{head}_i \coloneqq \textsc{attention}(QW_i^Q, KW_i^K, VW_i^V) \tag{9}$$

E Set Transformer for variable-sized inputs

The SET TRANSFORMER readout supports variable-sized inputs, i.e. it is theoretically possible to start from a non-rectangular (ragged) tensor, such that zero-padding is avoided in the flattened representation. However, as our chosen deep learning library (PyTorch) does not support ragged tensors at the time of writing, our default implementation uses the already defined representations (denoted by H and h in the main text). We did, however, experiment with a computationally inefficient implementation relying on a for-loop instead of batching, without observing notable performance differences (not shown).

F Hyperparameters specific to neural readouts

Appendix Table 4: Summary of the hyperparameters used for neural aggregators for each dataset. The hyperparameter names follow the same notation as introduced in the main text. Dim., dimension.

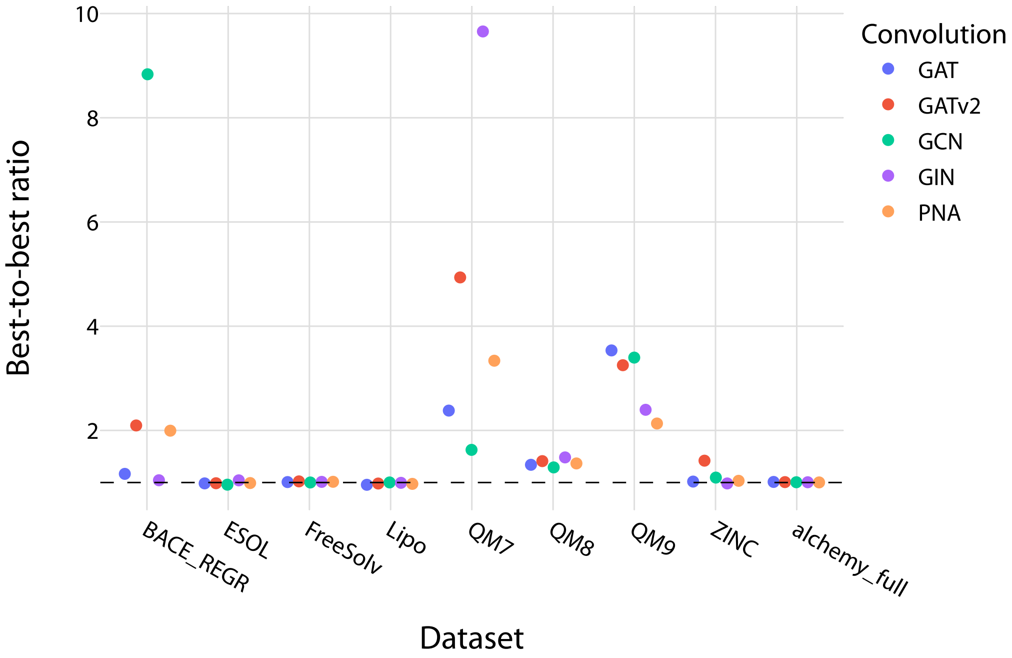
Dataset	**MLP**		**Set Transformer**		
	d_1	d_{out}	k	Hidden dim.	n_h
ESOL	64	32	8	32	4
FreeSolv	64	32	8	32	4
Lipo	64	32	8	32	4
BACE_REGR	64	32	8	32	4
BACE_CLS	64	32	8	32	4
BBBP	64	32	8	32	4
SIDER	64	32	8	32	4
QM7	128	64	8	64	8
QM8	128	64	8	64	8
QM9	256	128	8	512	8
PCBA	256	128	8	512	8
HIV	256	128	8	512	4
ENZYMES	256	128	8	64	8
PROTEINS_full	256	128	8	64	8
COIL-DEL	256	128	8	64	8
COIL-RAG	256	128	8	64	8
Cuneiform	256	128	8	64	8
github_stargazers	256	128	8	64	8
IMDB-BINARY	256	128	8	64	8
REDDIT-BINARY	256	128	8	64	8
REDDIT-MULTI-12K	256	128	8	64	8
reddit_threads	256	128	8	64	8
twitch_egos	256	128	8	64	8
TWITTER-Real-Graph-Partial	256	128	8	64	8
COLORS-3	256	128	8	64	8
SYNTHETIC	256	128	8	64	8
SYNTHETICnew	256	128	8	64	8
Synthie	256	128	8	64	8
TRIANGLES	256	128	8	64	8
MNIST	256	128	8	512	4
CIFAR10	256	128	8	512	4
ZINC	256	128	8	512	4

Appendix Table 5: (continued from Appendix F, Table 4) Summary of the hyperparameters used for neural aggregators for each dataset. The hyperparameter names follow the same notation as introduced in the main text. Dim., dimension.

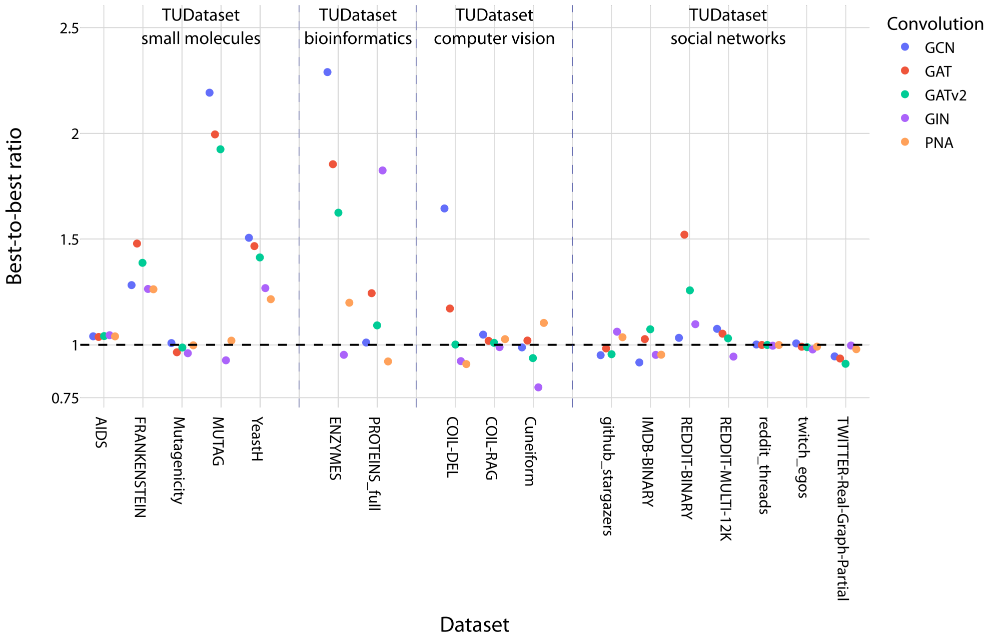
Dataset	MLP		Set Transformer		
	d_1	d_{out}	k	Hidden dim.	n_h
AIDS	256	128	8	256	8
FRANKENSTEIN	256	128	8	256	8
Mutagenicity	256	128	8	256	8
MUTAG	256	128	8	256	8
YeastH	256	128	8	256	8
alchemy_full	256	128	8	256	8
MalNetTiny	256	128	8	192	12

G Best-to-best ratios for all datasets

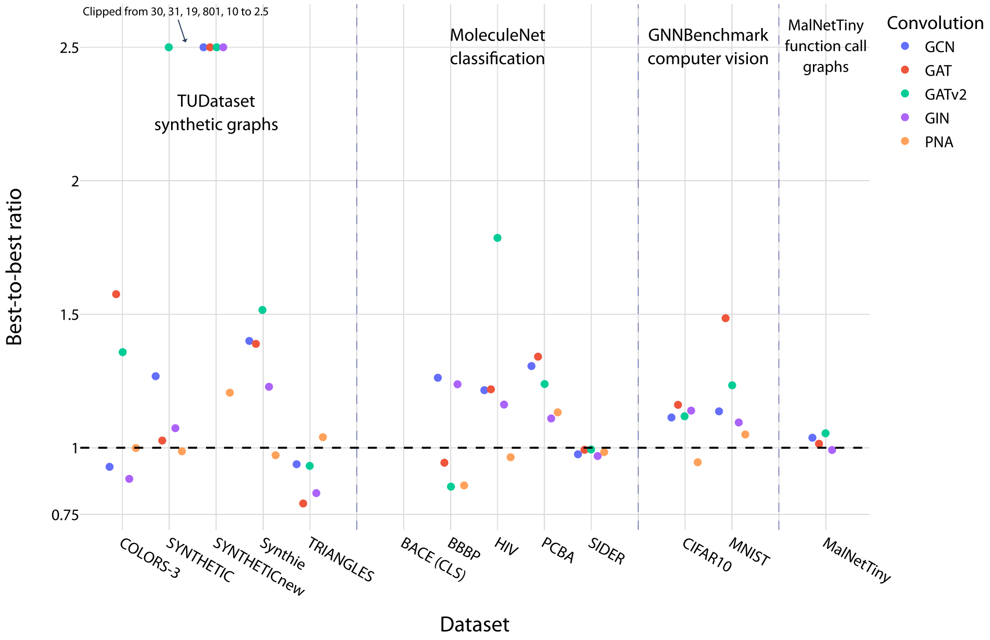
Appendix Figure 1: The performance of the best neural relative to the best standard readout on all regression benchmarks. We use the ratio between the effectiveness scores (R^2), computed by averaging over five random splits of the data. The differences are best appreciated by studying the associated tables (Appendix T, Tables 17, 26 and 31).



Appendix Figure 2: The performance of the best neural relative to the best standard readout on several classification benchmarks. We use the ratio between the effectiveness scores (MCC), computed by averaging over five random splits of the data. The differences are best appreciated by studying the associated tables (Appendix T, Tables 19, 20, 23 to 25 and 30).

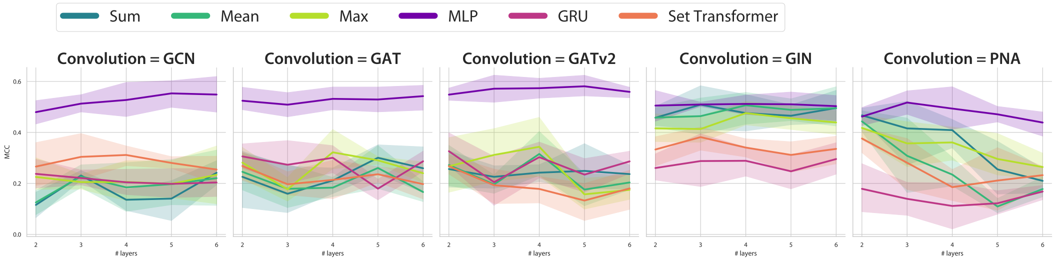


Appendix Figure 3: The performance of the best neural relative to the best standard readout on the rest of the classification benchmarks. We use the ratio between the effectiveness scores (MCC), computed by averaging over five random splits of the data. The differences are best appreciated by studying the associated tables (Appendix T, Tables 18, 21, 22, 28 and 29).



H Deeper GNN models for ENZYMES

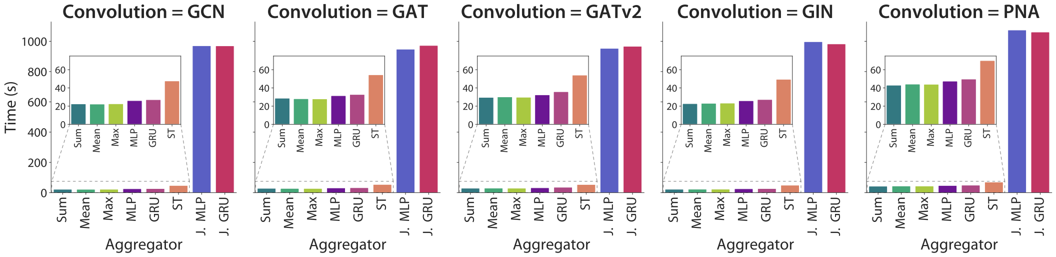
Appendix Figure 4: Increasing the number of neighborhood aggregation iterations or convolutional layers does not produce large differences on ENZYMES. For GCN, the MLP readout improves with deeper networks, whereas most readout are relatively stable regarding the number of layers on GAT, GATV2, and GIN. For PNA, the performance decreases drastically with more than 3 layers for the majority of readouts.



I Time and memory analysis

We benchmarked the elapsed training time and memory utilization on the QM9 dataset $(132,480$ data points) for one epoch on a modern high-end GPU (Nvidia RTX 3090, also see Appendix P), averaged from 5 epochs for each model (differences too small to plot error bars). As expected, the JANOSSY variants are not competitive in terms of training time. However, the MLP and GRU aggregators lead to a minimal increase of a few seconds per single epoch compared to the simple classical functions. The full SET TRANSFORMER architecture (also referred to as ST COMPLEX) incurs an increase close to 50%, which is, however, in line with the cost of transitioning from GCN to PNA. The training cost can be minimized by adopting ISAB blocks (trading off performance). The SET TRANSFORMER readout is applicable to large scale datasets, as exemplified by our evaluation which includes SET TRANSFORMER + PNA models with a maximum of 4 GNN layers for datasets of up to 2 million data points (also tested for 5 PNA layers).

Appendix Figure 5: Training times for all aggregators and convolutions on QM9 (seconds). The models are 2-layer GNNs. J., Janossy.

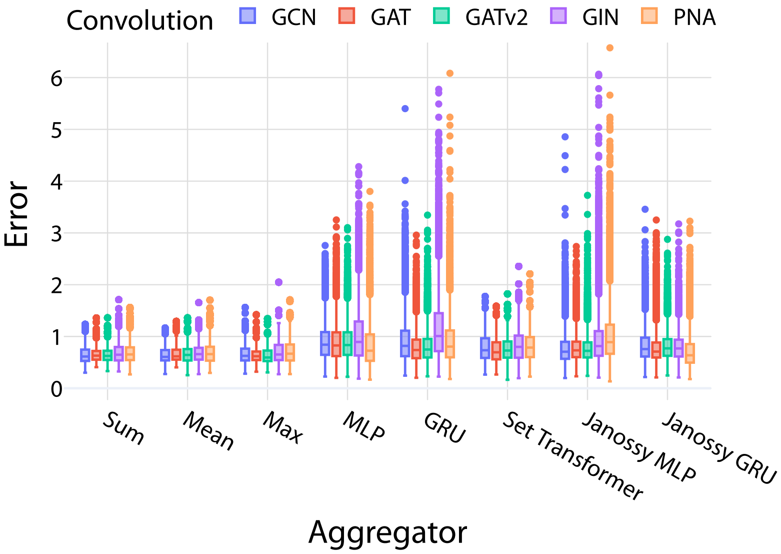


In terms of memory usage, the maximum amount of reserved (higher than allocated) memory as reported by the PyTorch profiler (version 1.10.1) was just under 149MB for the GRU + PNA model, a 27MB increase from the most efficient non-neural aggregator for PNA (mean). It should be noted that it is common for deep learning frameworks such as PyTorch and TensorFlow to automatically reserve or prepare large amounts of memory even if only a portion is allocated during training.

J Robustness to random node permutations of QM9 molecules

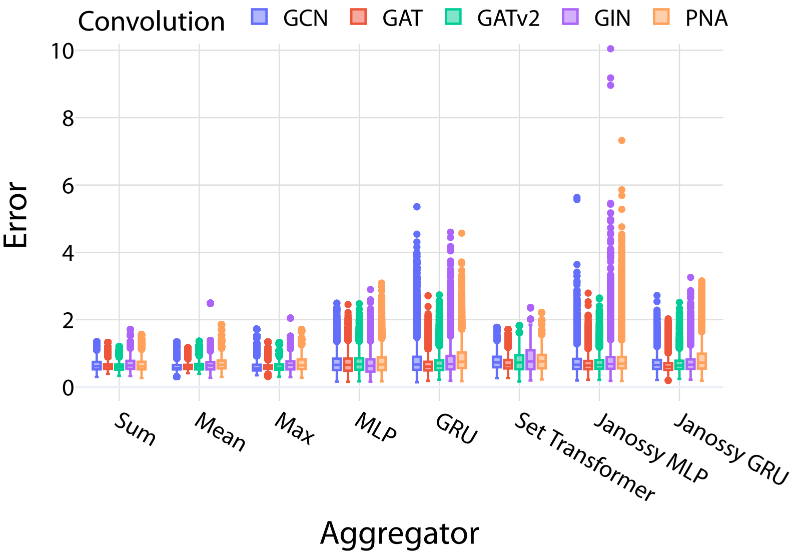
J.1 Random permutations of nodes

Appendix Figure 6: A summary of the error distributions for predictions made on random permutations of 50 randomly selected molecules from the QM9 dataset, presented per-convolution for the arbitrary random permutations strategy.

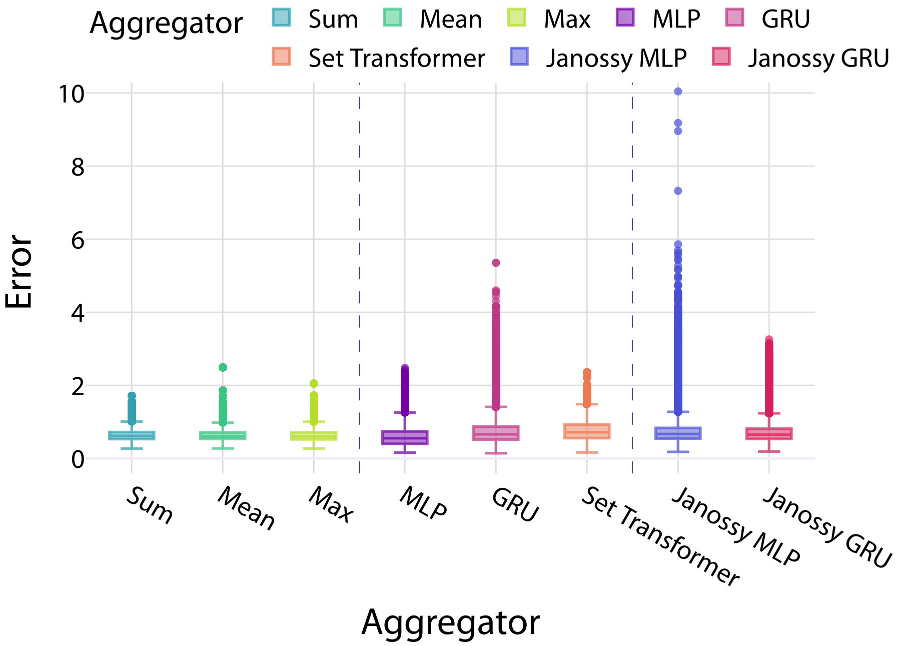


J.2 Random, non-canonical SMILES

Appendix Figure 7: A summary of the error distributions for predictions made on random permutations of 50 randomly selected molecules from the QM9 dataset, presented per-convolution for the random non-canonical SMILES strategy.

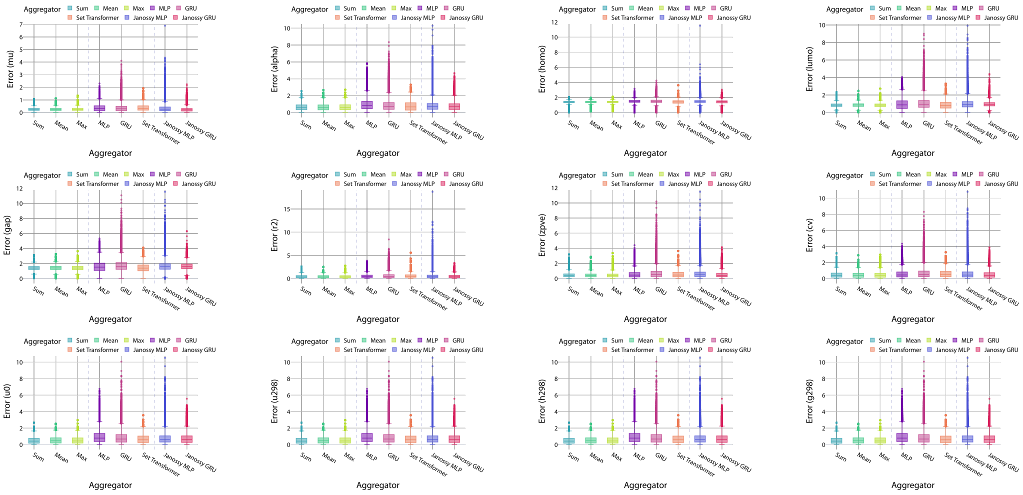


Appendix Figure 8: A summary of the error distributions for predictions made on random permutations of 50 randomly selected molecules from the QM9 dataset for the random non-canonical SMILES strategy, where we selected the top 50 lowest errors for each molecule from the multitude of non-canonical SMILES inputs.



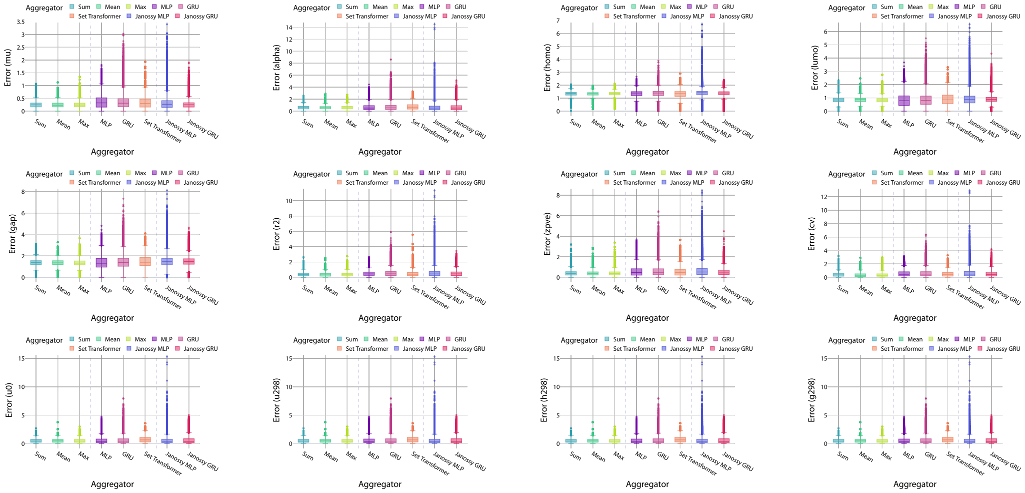
J.3 Error for each QM9 prediction task for random permutations of nodes

Appendix Figure 9: A summary of the error distributions for predictions made on random permutations of 50 randomly selected molecules from the QM9 dataset, presented per QM9 task (12 in total) for the arbitrary random permutations strategy.



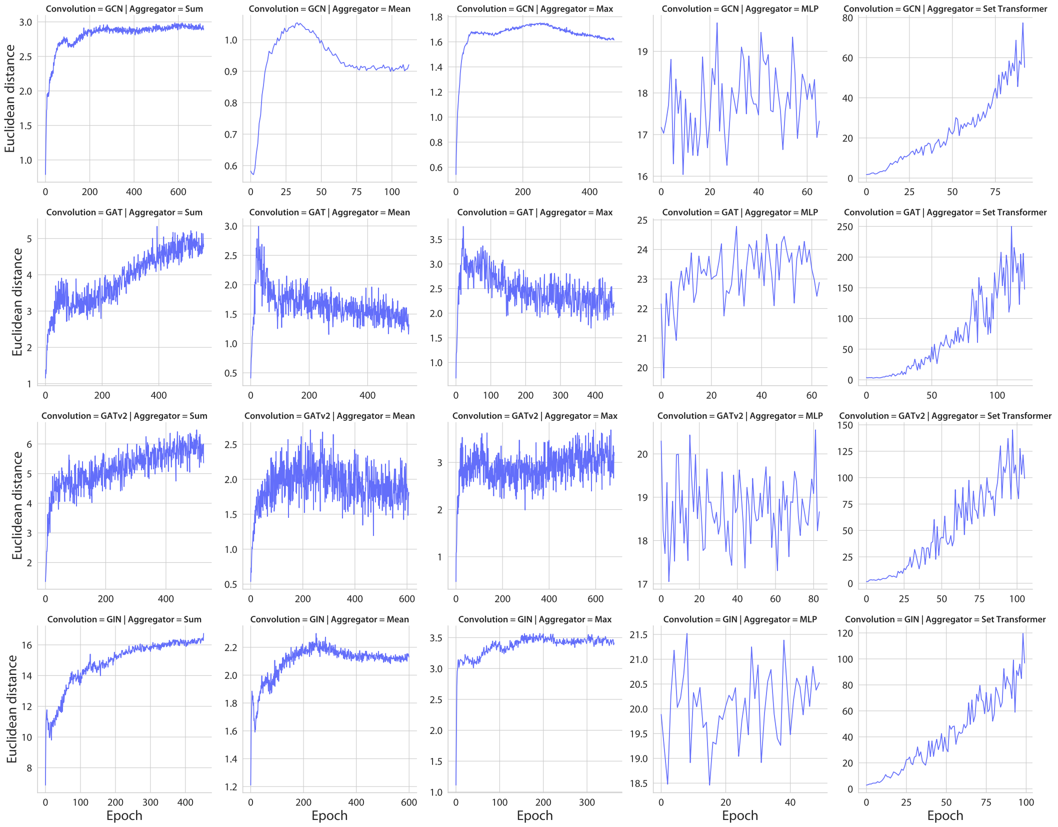
J.4 Error for each QM9 prediction task for random SMILES

Appendix Figure 10: A summary of the error distributions for predictions made on random permutations of 50 randomly selected molecules from the QM9 dataset, presented per QM9 task (12 in total) for the random non-canonical SMILES strategy.

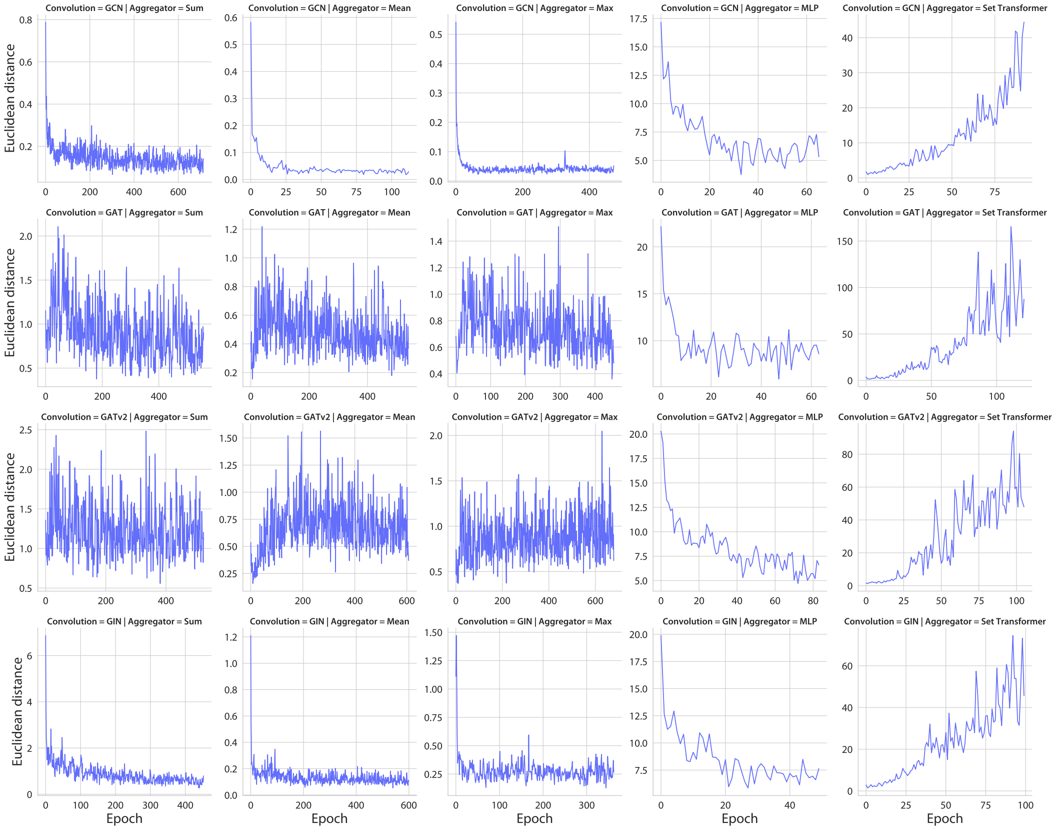


K Distances between graph representations

Appendix Figure 11: The Euclidean distance was computed between the initial graph representation (i.e. after the first epoch) and all subsequent epochs for a random molecule of the QM9 dataset, for multiple graph convolution types and readouts (2-layer GNNs). Generally, models using standard aggregators (sum, mean, max) take a long time to converge (500 to $1,000$ epochs), with only minor modifications to the graph representation. The models using neural readouts typically converge in under 100 epochs and are able to explore a much larger hypothesis space, as indicated by the large distances between the initial and final trained representations.



Appendix Figure 12: The Euclidean distance was computed between the graph representations from consecutive epochs for a random molecule of the QM9 dataset (same as Appendix Figure 11), for multiple graph convolution types and readouts (2-layer GNNs). Generally, models using standard aggregators (sum, mean, max) take a long time to converge (500 to $1,000$ epochs), with only minor modifications to the graph representation. The models using neural readouts typically converge in under 100 epochs and are able to explore a much larger hypothesis space, as indicated by the large distances between the initial and final trained representations.

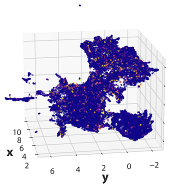
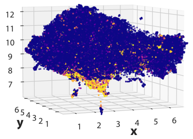
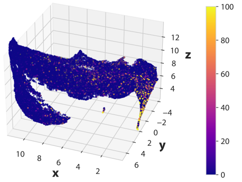
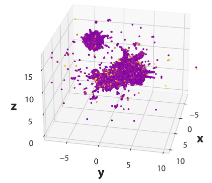
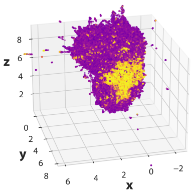
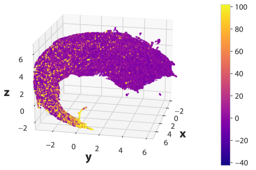
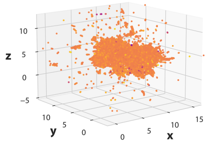
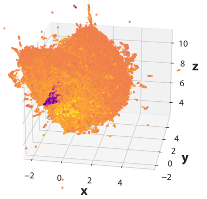
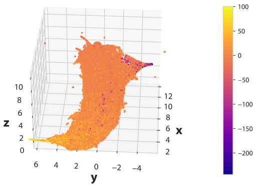


L Visualization of the learned latent space for different aggregators

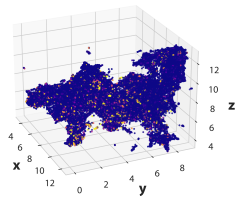
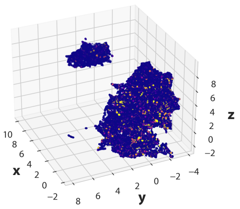
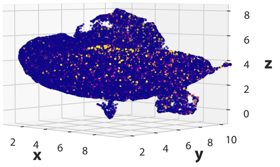
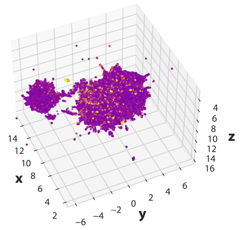
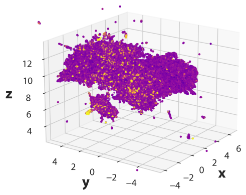
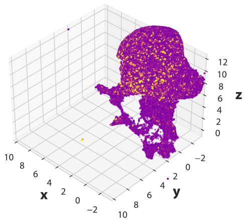
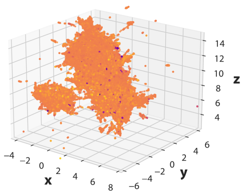
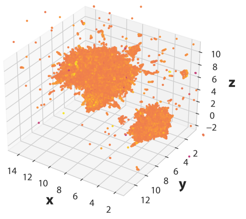
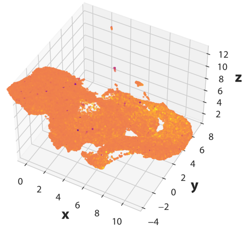
Guided VGAE architecture

We adapted our fixed architecture into a variational graph autoencoder (VGAE, as introduced by Kipf and Welling [19]). The changes include two additional layers for the μ and σ parameters, as well as the reconstruction and regularization losses (as provided by PyTorch Geometric). The graph embeddings learned by the VGAE are fed into a predictor network in an end-to-end fashion, such that the task is supervised.

Appendix Table 6: Visualization of the learned latent space of graphs (molecules) for three recently-introduced bio-affinity datasets, using UMAP projections in 3 dimensions. The figure presents a selection of 3 readouts. Angles are chosen to best highlight the 3D space structure. The activity of molecules, as reported in the bioassay, is illustrated according to the color bars.

Dataset	Sum	MLP	Set Transformer
AID1949			
AID449762			
AID602261			

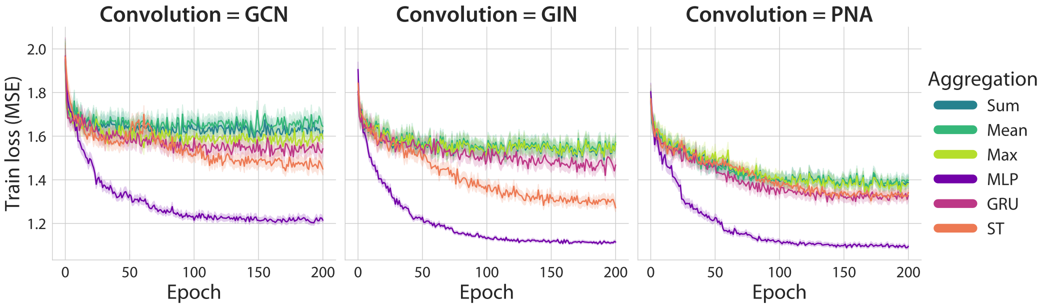
Appendix Table 7: Visualization of the learned latent space of graphs (molecules) for three recently-introduced bio-affinity datasets, using UMAP projections in 3 dimensions. The figure presents the other 3 readouts. Angles are chosen to best highlight the 3D space structure.

Dataset	Mean	Max	GRU
AID1949			
AID449762			
AID602261			

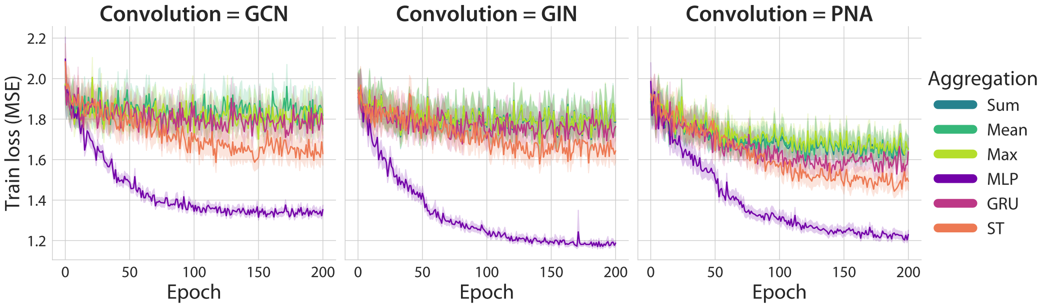
M Train losses plotted for the multi-million scale pharma datasets

M.1 VGAE models

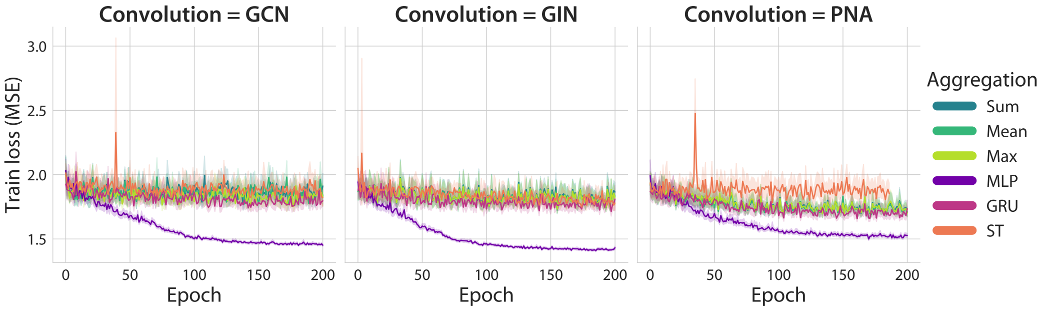
Appendix Figure 13: Train losses for the VGAE models trained on the proprietary dataset with ≈ 1 million molecules.



Appendix Figure 14: Train losses for the VGAE models trained on the proprietary dataset with ≈ 1.5 million molecules.

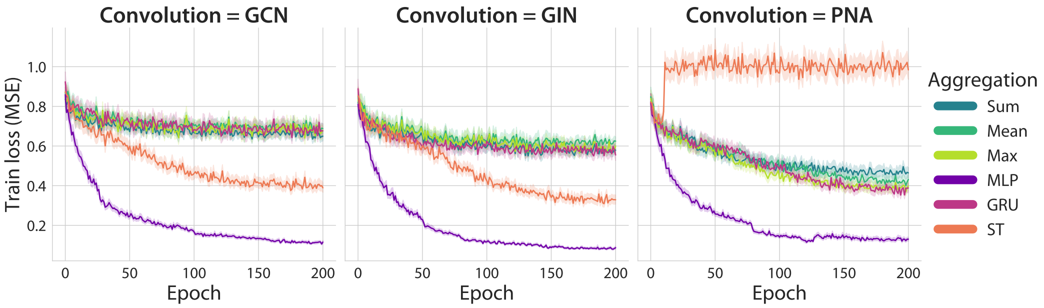


Appendix Figure 15: Train losses for the VGAE models trained on the proprietary dataset with ≈ 2 million molecules.

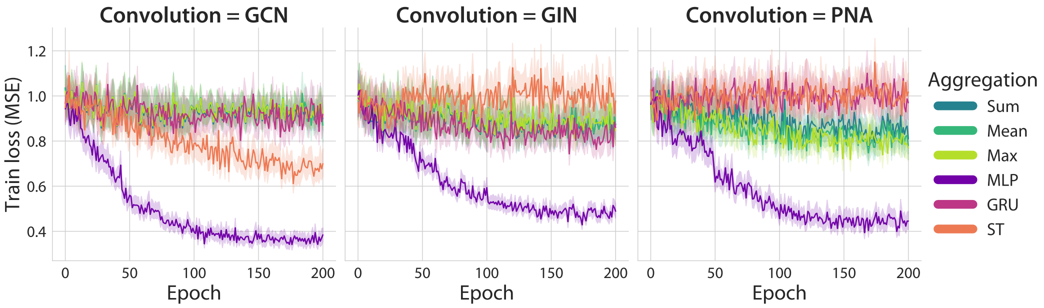


M.2 GNN models

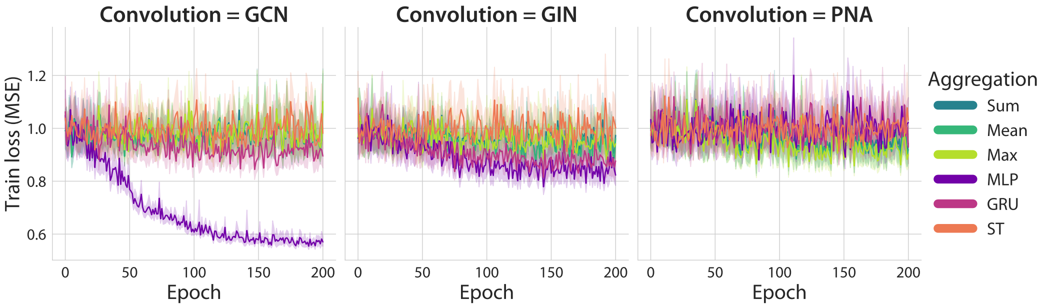
Appendix Figure 16: Train losses for the GNN models trained on the proprietary dataset with ≈ 1 million molecules.



Appendix Figure 17: Train losses for the GNN models trained on the proprietary dataset with ≈ 1.5 million molecules.

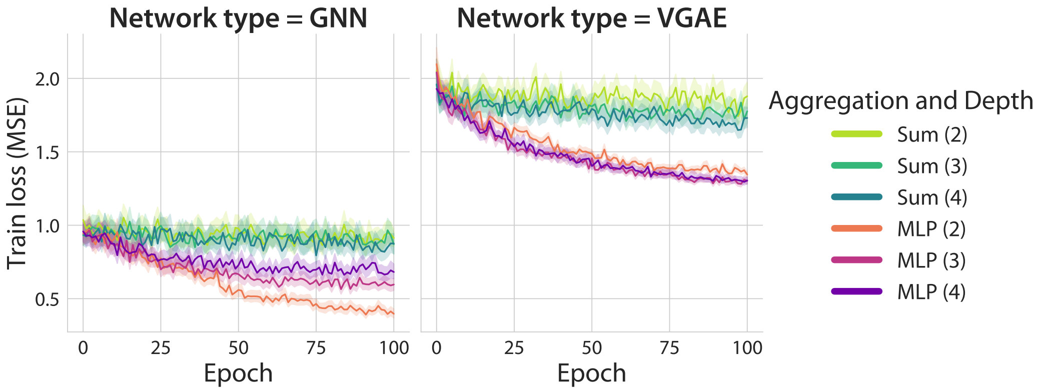


Appendix Figure 18: Train losses for the GNN models trained on the proprietary dataset with ≈ 2 million molecules.



N Train losses for varying GNN depths on the proprietary dataset with 1.5 million molecules

Appendix Figure 19: Train losses (MSE) for the GNN and guided VGAE models trained on the proprietary dataset with ≈ 1.5 million molecules. The illustration includes representative aggregators for each category (SUM, respectively MLP) evaluated with GCN layers across 3 different GNN depths (2, 3, and 4 layers). The VGAE is higher as it includes additional loss terms.



O Train metrics for the multi-million scale pharma datasets

Appendix Table 8: Train metrics for the VGAE models trained on the proprietary dataset with ≈1 million molecules. A p-value of $< \epsilon$ indicates that the number returned by `scipy` (`stats.pearsonr`) was below the machine precision (thus, reported as 0). A value of 'N/A' indicates that it was not possible to compute the metric.

Dataset	GNN or VGAE	Convolution	Aggregator	MAE	R^2	R^2 p-value
Bio-affinity 1 mil.	VGAE	GCN	Sum	1.13	0.29	$< \epsilon$
			Mean	1.15	0.27	$< \epsilon$
			Max	1.11	0.33	$< \epsilon$
			MLP	**0.68**	**0.78**	$< \epsilon$
			GRU	1.08	0.37	$< \epsilon$
			ST	1.00	0.47	$< \epsilon$
		GIN	Sum	1.07	0.38	$< \epsilon$
			Mean	1.08	0.37	$< \epsilon$
			Max	1.07	0.37	$< \epsilon$
			MLP	**0.55**	**0.86**	$< \epsilon$
			GRU	1.03	0.44	$< \epsilon$
			ST	0.84	0.65	$< \epsilon$
		PNA	Sum	0.93	0.55	$< \epsilon$
			Mean	0.94	0.53	$< \epsilon$
			Max	1.03	0.54	$< \epsilon$
			MLP	**0.56**	**0.86**	$< \epsilon$
			GRU	0.89	0.59	$< \epsilon$
			ST	0.76	0.72	$< \epsilon$

Appendix Table 9: Train metrics for the GNN models trained on the proprietary dataset with $\approx$1 million molecules. A p-value of $< \epsilon$ indicates that the number returned by `scipy` (`stats.pearsonr`) was below the machine precision (thus, reported as 0). A value of 'N/A' indicates that it was not possible to compute the metric.

Dataset	GNN or VGAE	Convolution	Aggregator	MAE	R^2	R^2 p-value
Bio-affinity 1 mil.	GNN	GCN	Sum	1.10	0.34	$< \epsilon$
			Mean	1.12	0.30	$< \epsilon$
			Max	1.11	0.32	$< \epsilon$
			MLP	**0.47**	**0.89**	$< \epsilon$
			GRU	1.11	0.32	$< \epsilon$
			ST	0.86	0.60	$< \epsilon$
		GIN	Sum	1.03	0.43	$< \epsilon$
			Mean	1.06	0.38	$< \epsilon$
			Max	1.04	0.41	$< \epsilon$
			MLP	**0.40**	**0.92**	$< \epsilon$
			GRU	1.04	0.42	$< \epsilon$
			ST	0.79	0.66	$< \epsilon$
		PNA	Sum	0.94	0.54	$< \epsilon$
			Mean	0.90	0.59	$< \epsilon$
			Max	0.87	0.61	$< \epsilon$
			MLP	**0.46**	**0.87**	$< \epsilon$
			GRU	0.85	0.62	$< \epsilon$
			ST	1.34	N/A	N/A

Appendix Table 10: Train metrics for the VGAE models trained on the proprietary dataset with ≈1.5 million molecules. A p-value of $< \epsilon$ indicates that the number returned by `scipy` (`stats.pearsonr`) was below the machine precision (thus, reported as 0). A value of 'N/A' indicates that it was not possible to compute the metric.

Dataset	GNN or VGAE	Convolution	Aggregator	MAE	R^2	R^2 p-value
Bio-affinity 1.5 mil.	VGAE	GCN	Sum	1.28	0.07	$< \epsilon$
			Mean	1.28	0.05	$< \epsilon$
			Max	1.32	0.07	$< \epsilon$
			MLP	**0.93**	**0.64**	$< \epsilon$
			GRU	1.28	0.10	$< \epsilon$
			ST	1.17	0.29	$< \epsilon$
		GIN	Sum	1.26	0.11	$< \epsilon$
			Mean	1.25	0.10	$< \epsilon$
			Max	1.31	0.10	$< \epsilon$
			MLP	**0.78**	**0.78**	$< \epsilon$
			GRU	1.26	0.13	$< \epsilon$
			ST	1.19	0.27	$< \epsilon$
		PNA	Sum	1.21	0.26	$< \epsilon$
			Mean	1.21	0.26	$< \epsilon$
			Max	1.34	0.24	$< \epsilon$
			MLP	**0.83**	**0.73**	$< \epsilon$
			GRU	1.19	0.30	$< \epsilon$
			ST	1.09	0.43	$< \epsilon$

Appendix Table 11: Train metrics for the GNN models trained on the proprietary dataset with ≈1.5 million molecules. A p-value of $< \epsilon$ indicates that the number returned by `scipy` (`stats.pearsonr`) was below the machine precision (thus, reported as 0). A value of 'N/A' indicates that it was not possible to compute the metric.

Dataset	GNN or VGAE	Convolution	Aggregator	MAE	R^2	R^2 p-value
Bio-affinity 1.5 mil.	GNN	GCN	Sum	1.29	0.07	$< \epsilon$
			Mean	1.29	0.06	$< \epsilon$
			Max	1.29	0.06	$< \epsilon$
			MLP	**0.90**	**0.64**	$< \epsilon$
			GRU	1.28	0.09	$< \epsilon$
			ST	1.15	0.32	$< \epsilon$
		GIN	Sum	1.26	0.13	$< \epsilon$
			Mean	1.27	0.12	$< \epsilon$
			Max	1.27	0.11	$< \epsilon$
			MLP	**1.04**	**0.50**	$< \epsilon$
			GRU	1.25	0.16	$< \epsilon$
			ST	1.30	N/A	N/A
		PNA	Sum	1.25	0.16	$< \epsilon$
			Mean	1.24	0.19	$< \epsilon$
			Max	1.23	0.20	$< \epsilon$
			MLP	**0.96**	**0.56**	$< \epsilon$
			GRU	1.30	0.00	4.22×10^{21}
			ST	1.30	N/A	N/A

Appendix Table 12: Train metrics for the VGAE models trained on the proprietary dataset with ≈2 million molecules. A p-value of $< \epsilon$ indicates that the number returned by `scipy` (`stats.pearsonr`) was below the machine precision (thus, reported as 0). A value of 'N/A' indicates that it was not possible to compute the metric.

Dataset	GNN or VGAE	Convolution	Aggregator	MAE	R^2	R^2 p-value
Bio-affinity 2 mil.	VGAE	GCN	Sum	0.83	0.02	$< \epsilon$
			Mean	0.83	0.03	$< \epsilon$
			Max	0.84	0.06	$< \epsilon$
			MLP	**0.63**	**0.52**	$< \epsilon$
			GRU	0.82	0.09	$< \epsilon$
			ST	0.83	N/A	N/A
		GIN	Sum	0.82	0.06	$< \epsilon$
			Mean	0.82	0.07	$< \epsilon$
			Max	0.83	0.08	$< \epsilon$
			MLP	**0.61**	**0.55**	$< \epsilon$
			GRU	0.82	0.11	$< \epsilon$
			ST	0.82	0.10	$< \epsilon$
		PNA	Sum	0.82	0.15	$< \epsilon$
			Mean	0.82	0.15	$< \epsilon$
			Max	0.84	0.15	$< \epsilon$
			MLP	**0.70**	**0.42**	$< \epsilon$
			GRU	0.81	0.17	$< \epsilon$
			ST	0.83	N/A	N/A

Appendix Table 13: Train metrics for the GNN models trained on the proprietary dataset with ≈2 million molecules. A p-value of $< \epsilon$ indicates that the number returned by `scipy` (`stats.pearsonr`) was below the machine precision (thus, reported as 0). A value of 'N/A' indicates that it was not possible to compute the metric.

Dataset	**GNN or VGAE**	**Convolution**	**Aggregator**	**MAE**	**R^2**	**R^2 p-value**
Bio-affinity 2 mil.	GNN	GCN	Sum	0.83	0.03	$< \epsilon$
			Mean	0.83	0.03	$< \epsilon$
			Max	0.83	0.02	$< \epsilon$
			MLP	**0.68**	**0.43**	$< \epsilon$
			GRU	0.82	0.09	$< \epsilon$
			ST	0.83	N/A	N/A
		GIN	Sum	0.82	0.05	$< \epsilon$
			Mean	0.82	0.06	$< \epsilon$
			Max	0.82	0.05	$< \epsilon$
			MLP	**0.80**	**0.16**	$< \epsilon$
			GRU	0.81	0.12	$< \epsilon$
			ST	0.83	N/A	N/A
		PNA	Sum	0.82	0.05	$< \epsilon$
			Mean	0.82	0.06	$< \epsilon$
			Max	**0.82**	**0.07**	$< \epsilon$
			MLP	0.83	0.00	1.29×10^{-13}
			GRU	0.83	0.00	$< \epsilon$
			ST	0.83	N/A	N/A

P Experimental platform

We used two different platforms for training all the models discussed in the paper. Firstly, a workstation equipped with an AMD Ryzen 5950X processor with 16 cores and 32 threads, an Nvidia RTX 3090 graphics card with 24GB of VRAM, and 64GB of DDR4 RAM. The used operating system is Ubuntu 21.10, with Python 3.9.9, PyTorch 1.10.1 with CUDA 11.3, PyTorch Geometric 2.0.3, and PyTorch Lightning 1.5.7.

Secondly, we used GPU-enabled systems from the AstraZeneca Scientific Computing Platform, generally equipped with Intel processors, Nvidia Tesla V100 GPUs with either 16GB or 32GB, and as much RAM as needed for the experiments. The cloud systems run CentOS Linux 7, with Python 3.9.7, PyTorch 1.8.2 and CUDA 10.2, PyTorch Geometric 2.0.1, and PyTorch Lightning 1.5.5.

Q Statistical significance of dataset attributes for the observed performance

We computed the difference between the best neural aggregator and the best non-neural aggregator for each dataset and for each one of the five random splits (ΔR^2 for regression datasets and ΔMCC for classification datasets). Using these resulting metrics, we fitted multiple linear regression models (`lm` in R) to explain and quantify the relationship between dataset attributes (size, average number of nodes per graph, and others) and the difference in performance.

For the regression datasets, all the dataset attributes from Appendix A, Table 2 except the number of tasks were statistically significant at different levels (Appendix Q, Table 14). Most had a positive relationship to the ΔR^2 except the average number of edges per graph and the number of tasks per dataset. These results suggest that larger regression datasets with more nodes per graph and more node features are likely to see large improvements when using neural aggregators.

For the classification datasets there were no statistically significant dataset attributes (Appendix Q, Table 15), indicating that the proposed techniques are not particularly tied to characteristics like dataset size or graph properties.

Appendix Table 14: Summary of the multiple linear regression model that explains the ΔR^2 in terms of the dataset attributes. p-value significance is indicated by the ‘Sign.’ column (‘***’ for $<$ 0.001, ‘**’ for $<$ 0.01, ‘.’ for $<$ 0.1).

Predictors	Estimates	95% Confidence Interval			p-value	Sign.
(Intercept)	-7.06×10^{-01}	-9.88×10^{-01}	–	-4.24×10^{-01}	1.33×10^{-05}	***
Size	1.84×10^{-06}	8.99×10^{-07}	–	2.78×10^{-06}	3.46×10^{-04}	***
Avg. nodes	6.32×10^{-02}	2.71×10^{-02}	–	9.94×10^{-02}	1.14×10^{-03}	**
Avg. edges	-2.36×10^{-02}	-3.92×10^{-02}	–	-8.01×10^{-03}	4.11×10^{-03}	**
Node attr.	1.83×10^{-02}	1.03×10^{-02}	–	2.63×10^{-02}	4.96×10^{-05}	***
Num. tasks	-8.59×10^{-03}	-1.73×10^{-02}	–	9.44×10^{-05}	5.24×10^{-02}	.

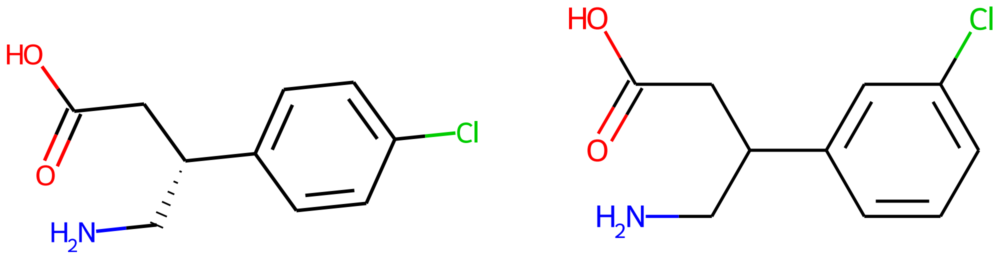
Appendix Table 15: Summary of the multiple linear regression model that explains the ΔMCC in terms of the dataset attributes. p-value significance is indicated by the ‘Sign.’ column (‘***’ for $<$ 0.001).

Predictors	Estimates	95% Confidence Interval			p-value	Sign.
(Intercept)	8.83×10^{-02}	5.40×10^{-02}	–	1.23×10^{-01}	1.10×10^{-06}	***
Size	-1.60×10^{-07}	-4.94×10^{-07}	–	1.73×10^{-07}	0.344	
Avg. nodes	2.10×10^{-04}	-2.10×10^{-04}	–	6.29×10^{-04}	0.326	
Avg. edges	-8.56×10^{-05}	-2.65×10^{-04}	–	9.43×10^{-05}	0.349	
Node attr.	-3.36×10^{-05}	-8.60×10^{-05}	–	1.88×10^{-05}	0.207	
Num. tasks	-3.70×10^{-04}	-1.29×10^{-03}	–	5.47×10^{-04}	0.426	

R Similar molecules with different representations

The example illustrates two very similar molecules with greatly different adjacency matrix representations (Appendix R, Figure 20). Despite the similarity, only 6 out of 14 rows are identical in both matrices (rows 1, 2, 3, 4, 6, 8, numbering from 0). There are no rows which occur in both matrices but in different orders.

Appendix Figure 20: Similar molecules with different adjacency representations. The SMILES is provided for both molecules.



(a) `NC[C@H](CC(=O)O)c1ccc(Cl)cc1`

(b) `C1=CC(=CC(=C1)Cl)C(CC(=O)O)CN`

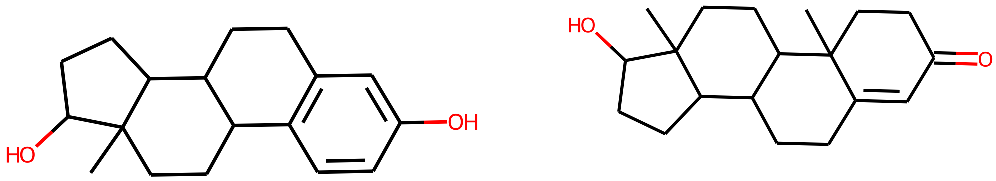
$$\begin{pmatrix}
0&1&0&0&0&0&0&0&0&0&0&0&0&0\\
1&0&1&0&0&0&0&0&0&0&0&0&0&0\\
0&1&0&1&0&0&0&1&0&0&0&0&0&0\\
0&0&1&0&1&0&0&0&0&0&0&0&0&0\\
0&0&0&1&0&1&1&0&0&0&0&0&0&0\\
0&0&0&0&1&0&0&0&0&0&0&0&0&0\\
0&0&0&0&1&0&0&0&0&0&0&0&0&0\\
0&0&1&0&0&0&0&0&1&0&0&0&0&1\\
0&0&0&0&0&0&0&1&0&1&0&0&0&0\\
0&0&0&0&0&0&0&0&1&0&1&0&0&0\\
0&0&0&0&0&0&0&0&0&1&0&1&1&0\\
0&0&0&0&0&0&0&0&0&0&1&0&0&0\\
0&0&0&0&0&0&0&0&0&0&1&0&0&1\\
0&0&0&0&0&0&0&1&0&0&0&0&1&0
\end{pmatrix}
\begin{pmatrix}
0&1&0&0&0&1&0&0&0&0&0&0&0&0\\
1&0&1&0&0&0&0&0&0&0&0&0&0&0\\
0&1&0&1&0&0&0&1&0&0&0&0&0&0\\
0&0&1&0&1&0&0&0&0&0&0&0&0&0\\
0&0&0&1&0&1&1&0&0&0&0&0&0&0\\
1&0&0&0&1&0&0&0&0&0&0&0&0&0\\
0&0&0&0&1&0&0&0&0&0&0&0&0&0\\
0&0&1&0&0&0&0&0&1&0&0&0&1&0\\
0&0&0&0&0&0&0&1&0&1&0&0&0&0\\
0&0&0&0&0&0&0&0&1&0&1&1&0&0\\
0&0&0&0&0&0&0&0&0&1&0&0&0&0\\
0&0&0&0&0&0&0&0&0&1&0&0&0&0\\
0&0&0&0&0&0&0&1&0&0&0&0&0&1\\
0&0&0&0&0&0&0&0&0&0&0&0&1&0
\end{pmatrix}$$

S Similar molecules and the expressiveness of aggregators

As a proof of concept, we selected two well-known molecules with similar structures, estradiol and testosterone (Appendix S, Figure 21). Despite the similarity, their induced biological effects can be very different. We can easily find several examples where the two compounds have different activity levels (active vs inactive) in high-throughput assays from PubChem, for example AID 588544, AID 1347036, AID 624032, or AID 1259394. The compound identifiers (CIDs) for the two compounds are 5757 for estradiol and 6013 for testosterone.

However, GNNs using sum, mean, or max readouts might find it challenging to discern between the two for bio-affinity predictions tasks. On a toy GNN with a single GCN layer, DeepChem featurisation (30 node features), 5 output features from the GCN layer, and random initialization with a seed of 1 (`pytorch_lightning.seed_everything(1)`), the three classical aggregators produced extremely close outputs (Appendix S, Table 16). Although this is only a toy example, it highlights one possible limitation of the simple, existing readout functions.

Appendix Figure 21: Example of similar molecules with different properties.



(a) Estradiol
`CC12CCC3C(C1CCC2O)CCC4=C3C=CC(=C4)O`

(b) Testosterone
`CC12CCC3C(C1CCC2O)CCC4=CC(=O)CCC34C`

Appendix Table 16: Output of the three simple functions for the two similar molecules.

Aggregator	**Molecule**	
	Estradiol	Testosterone
Sum	-39.296	-40.433
Mean	-0.393	-0.385
Max	0.829	0.762

T Detailed metrics for all 39 datasets/benchmarks

The metrics for each layer type, readout, and random split are available in **Supplementary File 2** (available on GitHub).

T.1 MoleculeNet regression models

Appendix Table 17: Detailed metrics (mean ± standard deviation) for the MoleculeNet regression datasets. For QM9, any differences in performance compared to other 2-layer models such as those in Figure 3 might be due to different GNN hyperparameters, such as the output or intermediate node dimension (QM9-specific experiments generally used larger dimensions).

		GCN		GAT		GATv2		GIN		PNA	
Data.	Agg.	MAE	R^2	MAE	R^2	MAE	R^2	MAE	R^2	MAE	R^2
QM9	Sum	0.74 ± 0.00	0.09 ± 0.00	0.76 ± 0.00	0.05 ± 0.01	0.75 ± 0.00	0.08 ± 0.01	0.71 ± 0.00	0.15 ± 0.00	0.70 ± 0.00	0.17 ± 0.01
	Mean	0.73 ± 0.00	0.10 ± 0.00	0.75 ± 0.01	0.07 ± 0.01	0.74 ± 0.00	0.10 ± 0.00	0.72 ± 0.00	0.13 ± 0.00	0.70 ± 0.00	0.16 ± 0.01
	Max	0.73 ± 0.00	0.11 ± 0.00	0.74 ± 0.00	0.10 ± 0.00	0.73 ± 0.00	0.11 ± 0.00	0.70 ± 0.00	0.16 ± 0.00	0.68 ± 0.00	0.20 ± 0.00
	MLP	0.63 ± 0.00	0.31 ± 0.00	0.64 ± 0.00	0.30 ± 0.01	0.64 ± 0.01	0.29 ± 0.01	0.60 ± 0.00	0.38 ± 0.00	0.58 ± 0.00	0.41 ± 0.00
	GRU	0.62 ± 0.01	0.34 ± 0.02	0.62 ± 0.01	0.34 ± 0.01	0.61 ± 0.02	0.35 ± 0.03	0.60 ± 0.00	0.38 ± 0.00	0.60 ± 0.01	0.37 ± 0.02
	ST	0.60 ± 0.01	0.38 ± 0.01	0.63 ± 0.02	0.30 ± 0.04	0.62 ± 0.01	0.32 ± 0.02	0.59 ± 0.00	0.39 ± 0.01	0.57 ± 0.01	0.44 ± 0.01
QM8	Sum	0.58 ± 0.01	0.08 ± 0.01	0.58 ± 0.01	0.08 ± 0.01	0.58 ± 0.01	0.09 ± 0.00	0.59 ± 0.01	0.07 ± 0.01	0.58 ± 0.01	0.09 ± 0.01
	Mean	0.59 ± 0.01	0.07 ± 0.01	0.59 ± 0.01	0.06 ± 0.00	0.59 ± 0.01	0.07 ± 0.01	0.60 ± 0.01	0.07 ± 0.01	0.58 ± 0.01	0.09 ± 0.01
	Max	0.59 ± 0.01	0.07 ± 0.01	0.59 ± 0.01	0.07 ± 0.01	0.59 ± 0.01	0.08 ± 0.01	0.59 ± 0.01	0.07 ± 0.01	0.59 ± 0.01	0.09 ± 0.01
	MLP	0.58 ± 0.01	0.10 ± 0.00	0.57 ± 0.01	0.11 ± 0.01	0.58 ± 0.01	0.10 ± 0.01	0.57 ± 0.01	0.11 ± 0.01	0.56 ± 0.01	0.13 ± 0.02
	GRU	0.57 ± 0.01	0.10 ± 0.01	0.57 ± 0.01	0.11 ± 0.01	0.56 ± 0.01	0.13 ± 0.01	0.58 ± 0.01	0.10 ± 0.01	0.58 ± 0.01	0.10 ± 0.01
	ST	0.59 ± 0.01	0.08 ± 0.01	0.58 ± 0.01	0.10 ± 0.01	0.57 ± 0.01	0.10 ± 0.01	0.59 ± 0.01	0.09 ± 0.01	0.57 ± 0.01	0.12 ± 0.01
QM7	Sum	0.75 ± 0.04	0.15 ± 0.13	0.76 ± 0.03	0.10 ± 0.01	0.78 ± 0.03	0.05 ± 0.05	0.82 ± 0.03	0.00 ± 0.00	0.79 ± 0.03	0.06 ± 0.02
	Mean	0.80 ± 0.03	0.03 ± 0.01	0.79 ± 0.04	0.03 ± 0.02	0.79 ± 0.04	0.03 ± 0.02	0.81 ± 0.03	0.01 ± 0.00	0.79 ± 0.03	0.03 ± 0.02
	Max	0.78 ± 0.04	0.06 ± 0.02	0.78 ± 0.04	0.06 ± 0.03	0.79 ± 0.04	0.03 ± 0.03	0.81 ± 0.04	0.01 ± 0.01	0.78 ± 0.03	0.05 ± 0.03
	MLP	0.79 ± 0.03	0.05 ± 0.02	0.79 ± 0.04	0.06 ± 0.02	0.79 ± 0.04	0.06 ± 0.02	0.80 ± 0.03	0.02 ± 0.01	0.80 ± 0.03	0.02 ± 0.01
	GRU	0.70 ± 0.03	0.24 ± 0.04	0.70 ± 0.03	0.24 ± 0.03	0.69 ± 0.03	0.25 ± 0.05	0.85 ± 0.06	0.09 ± 0.02	0.72 ± 0.02	0.21 ± 0.04
	ST	0.80 ± 0.03	0.03 ± 0.01	0.81 ± 0.04	0.02 ± 0.02	0.81 ± 0.04	0.01 ± 0.01	0.81 ± 0.04	0.01 ± 0.01	0.81 ± 0.04	0.01 ± 0.01
BACE	Sum	1.10 ± 0.00	0.06 ± 0.00	1.05 ± 0.01	0.12 ± 0.05	1.05 ± 0.00	0.14 ± 0.00	1.01 ± 0.00	0.14 ± 0.00	1.06 ± 0.01	0.23 ± 0.15
	Mean	1.05 ± 0.00	0.07 ± 0.00	1.07 ± 0.00	0.02 ± 0.00	1.09 ± 0.00	0.29 ± 0.00	0.97 ± 0.03	0.53 ± 0.02	1.07 ± 0.00	0.15 ± 0.00
	Max	1.05 ± 0.00	0.00 ± 0.00	1.06 ± 0.00	0.50 ± 0.00	1.04 ± 0.00	0.28 ± 0.00	1.04 ± 0.00	0.27 ± 0.00	1.07 ± 0.00	0.29 ± 0.01
	MLP	0.72 ± 0.03	0.63 ± 0.03	0.76 ± 0.02	0.54 ± 0.03	0.75 ± 0.02	0.55 ± 0.03	0.77 ± 0.02	0.56 ± 0.04	0.71 ± 0.03	0.57 ± 0.04
	GRU	0.76 ± 0.00	0.61 ± 0.00	0.77 ± 0.00	0.58 ± 0.00	0.79 ± 0.00	0.60 ± 0.00	0.79 ± 0.00	0.49 ± 0.00	0.79 ± 0.01	0.57 ± 0.01
	ST	1.03 ± 0.00	0.01 ± 0.00	1.04 ± 0.00	0.24 ± 0.01	1.06 ± 0.00	0.40 ± 0.00	1.09 ± 0.00	0.40 ± 0.00	1.09 ± 0.00	0.41 ± 0.00
ESOL	Sum	0.25 ± 0.02	0.89 ± 0.01	0.25 ± 0.02	0.89 ± 0.01	0.24 ± 0.01	0.90 ± 0.01	0.33 ± 0.01	0.81 ± 0.03	0.26 ± 0.03	0.88 ± 0.02
	Mean	0.36 ± 0.03	0.78 ± 0.05	0.33 ± 0.02	0.81 ± 0.02	0.32 ± 0.02	0.82 ± 0.03	0.35 ± 0.04	0.79 ± 0.05	0.34 ± 0.01	0.80 ± 0.02
	Max	0.33 ± 0.02	0.81 ± 0.01	0.38 ± 0.02	0.78 ± 0.03	0.39 ± 0.02	0.78 ± 0.03	0.35 ± 0.02	0.78 ± 0.02	0.33 ± 0.04	0.80 ± 0.04
	MLP	0.30 ± 0.04	0.85 ± 0.02	0.28 ± 0.03	0.85 ± 0.02	0.29 ± 0.03	0.85 ± 0.02	0.33 ± 0.04	0.82 ± 0.03	0.30 ± 0.03	0.84 ± 0.03
	GRU	0.27 ± 0.03	0.85 ± 0.02	0.25 ± 0.02	0.88 ± 0.02	0.25 ± 0.02	0.88 ± 0.02	0.30 ± 0.03	0.85 ± 0.05	0.41 ± 0.16	0.70 ± 0.22
	ST	0.30 ± 0.03	0.86 ± 0.02	0.29 ± 0.03	0.86 ± 0.02	0.28 ± 0.03	0.86 ± 0.02	0.32 ± 0.03	0.84 ± 0.03	0.27 ± 0.02	0.87 ± 0.02
FreeSolv	Sum	0.19 ± 0.04	0.90 ± 0.06	0.20 ± 0.03	0.90 ± 0.05	0.21 ± 0.03	0.89 ± 0.05	0.27 ± 0.05	0.83 ± 0.07	0.22 ± 0.02	0.88 ± 0.05
	Mean	0.26 ± 0.09	0.85 ± 0.12	0.23 ± 0.03	0.89 ± 0.03	0.25 ± 0.02	0.88 ± 0.03	0.27 ± 0.04	0.86 ± 0.03	0.25 ± 0.03	0.88 ± 0.04
	Max	0.20 ± 0.04	0.91 ± 0.04	0.24 ± 0.03	0.89 ± 0.02	0.26 ± 0.02	0.89 ± 0.03	0.25 ± 0.03	0.88 ± 0.02	0.21 ± 0.02	0.91 ± 0.03
	MLP	0.25 ± 0.03	0.89 ± 0.06	0.22 ± 0.01	0.91 ± 0.03	0.21 ± 0.02	0.91 ± 0.04	0.26 ± 0.04	0.87 ± 0.05	0.21 ± 0.03	0.92 ± 0.03
	GRU	0.28 ± 0.21	0.76 ± 0.33	0.20 ± 0.04	0.90 ± 0.06	0.19 ± 0.02	0.90 ± 0.05	0.24 ± 0.06	0.88 ± 0.09	0.25 ± 0.07	0.88 ± 0.05
	ST	0.21 ± 0.03	0.91 ± 0.04	0.21 ± 0.03	0.90 ± 0.04	0.21 ± 0.02	0.91 ± 0.03	0.22 ± 0.03	0.89 ± 0.03	0.19 ± 0.02	0.92 ± 0.02
Lipo	Sum	0.42 ± 0.04	0.68 ± 0.03	0.39 ± 0.04	0.72 ± 0.03	0.39 ± 0.03	0.72 ± 0.02	0.40 ± 0.02	0.70 ± 0.02	0.36 ± 0.03	0.75 ± 0.03
	Mean	0.50 ± 0.07	0.58 ± 0.09	0.48 ± 0.05	0.63 ± 0.04	0.45 ± 0.04	0.66 ± 0.02	0.46 ± 0.05	0.64 ± 0.05	0.44 ± 0.04	0.67 ± 0.03
	Max	0.48 ± 0.03	0.60 ± 0.03	0.55 ± 0.04	0.53 ± 0.02	0.57 ± 0.02	0.49 ± 0.02	0.44 ± 0.03	0.65 ± 0.04	0.42 ± 0.02	0.68 ± 0.04
	MLP	0.54 ± 0.03	0.52 ± 0.03	0.53 ± 0.04	0.54 ± 0.04	0.53 ± 0.03	0.53 ± 0.04	0.53 ± 0.02	0.53 ± 0.03	0.48 ± 0.02	0.61 ± 0.02
	GRU	0.50 ± 0.04	0.58 ± 0.05	0.50 ± 0.08	0.61 ± 0.07	0.52 ± 0.06	0.62 ± 0.06	0.48 ± 0.05	0.61 ± 0.05	0.51 ± 0.14	0.54 ± 0.24
	ST	0.43 ± 0.01	0.68 ± 0.03	0.42 ± 0.02	0.69 ± 0.03	0.41 ± 0.03	0.71 ± 0.03	0.41 ± 0.01	0.70 ± 0.03	0.39 ± 0.03	0.73 ± 0.03

T.2 MoleculeNet classification models

Appendix Table 18: Detailed metrics (mean ± standard deviation) for the MoleculeNet classification datasets.

		GCN		GAT		GATv2		GIN		PNA	
Data.	Agg.	AUROC	MCC	AUROC	MCC	AUROC	MCC	AUROC	MCC	AUROC	MCC
PCBA	Sum	0.53 ± 0.00	0.20 ± 0.00	0.54 ± 0.00	0.22 ± 0.01	0.54 ± 0.00	0.24 ± 0.00	0.55 ± 0.00	0.25 ± 0.01	0.56 ± 0.01	0.27 ± 0.01
	Mean	0.53 ± 0.00	0.19 ± 0.01	0.53 ± 0.00	0.20 ± 0.01	0.54 ± 0.01	0.22 ± 0.01	0.55 ± 0.00	0.24 ± 0.01	0.56 ± 0.01	0.27 ± 0.01
	Max	0.53 ± 0.00	0.20 ± 0.01	0.53 ± 0.00	0.20 ± 0.01	0.53 ± 0.00	0.21 ± 0.01	0.55 ± 0.00	0.24 ± 0.01	0.57 ± 0.01	0.28 ± 0.01
	MLP	0.52 ± 0.01	0.12 ± 0.06	0.52 ± 0.01	0.13 ± 0.04	0.51 ± 0.01	0.12 ± 0.04	0.52 ± 0.02	0.15 ± 0.06	0.52 ± 0.01	0.15 ± 0.05
	GRU	0.53 ± 0.00	0.17 ± 0.01	0.55 ± 0.01	0.24 ± 0.01	0.55 ± 0.01	0.24 ± 0.01	0.55 ± 0.00	0.24 ± 0.01	0.56 ± 0.01	0.25 ± 0.01
	ST	0.56 ± 0.01	0.26 ± 0.01	0.58 ± 0.01	0.30 ± 0.01	0.58 ± 0.00	0.29 ± 0.01	0.57 ± 0.01	0.27 ± 0.01	0.59 ± 0.01	0.31 ± 0.01
BACE	Sum	0.65 ± 0.00	0.32 ± 0.00	0.67 ± 0.00	0.37 ± 0.00	0.70 ± 0.01	0.42 ± 0.02	0.62 ± 0.01	0.28 ± 0.01	0.54 ± 0.01	0.13 ± 0.04
	Mean	0.61 ± 0.01	0.23 ± 0.01	0.61 ± 0.00	0.22 ± 0.01	0.64 ± 0.02	0.28 ± 0.04	0.58 ± 0.00	0.16 ± 0.00	0.66 ± 0.01	0.32 ± 0.02
	Max	0.62 ± 0.01	0.25 ± 0.04	0.70 ± 0.00	0.40 ± 0.00	0.72 ± 0.00	0.43 ± 0.00	0.69 ± 0.02	0.38 ± 0.03	0.47 ± 0.00	-0.09 ± 0.01
	MLP	0.66 ± 0.01	0.32 ± 0.03	0.62 ± 0.00	0.25 ± 0.00	0.65 ± 0.02	0.31 ± 0.03	0.68 ± 0.00	0.36 ± 0.00	0.68 ± 0.01	0.36 ± 0.03
	GRU	0.69 ± 0.00	0.37 ± 0.00	0.70 ± 0.00	0.38 ± 0.00	0.73 ± 0.00	0.45 ± 0.00	0.66 ± 0.01	0.32 ± 0.02	0.63 ± 0.02	0.26 ± 0.03
	ST	0.64 ± 0.00	0.30 ± 0.01	0.69 ± 0.00	0.38 ± 0.00	0.66 ± 0.00	0.36 ± 0.00	0.68 ± 0.01	0.38 ± 0.02	0.65 ± 0.01	0.34 ± 0.01
BBBP	Sum	0.59 ± 0.00	0.21 ± 0.00	0.60 ± 0.00	0.23 ± 0.00	0.62 ± 0.00	0.27 ± 0.01	0.54 ± 0.00	0.11 ± 0.01	0.59 ± 0.01	0.21 ± 0.01
	Mean	0.50 ± 0.00	0.00 ± 0.00	0.52 ± 0.03	0.06 ± 0.08	0.55 ± 0.01	0.15 ± 0.04	0.50 ± 0.00	0.00 ± 0.00	0.58 ± 0.01	0.20 ± 0.02
	Max	0.50 ± 0.00	0.00 ± 0.00	0.63 ± 0.01	0.29 ± 0.02	0.63 ± 0.02	0.30 ± 0.04	0.57 ± 0.01	0.18 ± 0.01	0.62 ± 0.01	0.27 ± 0.03
	MLP	0.63 ± 0.01	0.27 ± 0.03	0.62 ± 0.02	0.27 ± 0.03	0.61 ± 0.02	0.26 ± 0.05	0.57 ± 0.01	0.22 ± 0.02	0.61 ± 0.02	0.24 ± 0.03
	GRU	0.53 ± 0.00	0.12 ± 0.00	0.58 ± 0.00	0.19 ± 0.01	0.58 ± 0.00	0.18 ± 0.01	0.56 ± 0.01	0.14 ± 0.03	0.52 ± 0.02	0.05 ± 0.05
	ST	0.54 ± 0.00	0.11 ± 0.01	0.54 ± 0.00	0.14 ± 0.01	0.53 ± 0.00	0.12 ± 0.00	0.53 ± 0.00	0.11 ± 0.01	0.54 ± 0.01	0.11 ± 0.02
SIDER	Sum	0.74 ± 0.01	0.50 ± 0.02	0.74 ± 0.01	0.50 ± 0.02	0.74 ± 0.01	0.50 ± 0.02	0.74 ± 0.01	0.50 ± 0.02	0.74 ± 0.01	0.50 ± 0.02
	Mean	0.74 ± 0.01	0.50 ± 0.01	0.73 ± 0.01	0.50 ± 0.01	0.73 ± 0.01	0.50 ± 0.02	0.74 ± 0.01	0.51 ± 0.01	0.73 ± 0.01	0.50 ± 0.01
	Max	0.74 ± 0.01	0.50 ± 0.02	0.74 ± 0.01	0.50 ± 0.01	0.74 ± 0.01	0.50 ± 0.02	0.74 ± 0.01	0.50 ± 0.01	0.74 ± 0.01	0.50 ± 0.02
	MLP	0.73 ± 0.01	0.49 ± 0.03	0.73 ± 0.00	0.49 ± 0.01	0.73 ± 0.01	0.48 ± 0.02	0.73 ± 0.00	0.49 ± 0.01	0.73 ± 0.01	0.49 ± 0.01
	GRU	0.73 ± 0.01	0.48 ± 0.02	0.74 ± 0.01	0.50 ± 0.03	0.74 ± 0.01	0.50 ± 0.02	0.73 ± 0.01	0.47 ± 0.02	0.73 ± 0.01	0.49 ± 0.01
	ST	0.73 ± 0.01	0.49 ± 0.02	0.73 ± 0.00	0.50 ± 0.01	0.73 ± 0.01	0.49 ± 0.01	0.73 ± 0.00	0.49 ± 0.01	0.73 ± 0.00	0.49 ± 0.01
HIV	Sum	0.55 ± 0.02	0.15 ± 0.03	0.56 ± 0.02	0.21 ± 0.06	0.55 ± 0.04	0.17 ± 0.08	0.58 ± 0.02	0.23 ± 0.05	0.62 ± 0.01	0.35 ± 0.01
	Mean	0.58 ± 0.03	0.18 ± 0.07	0.51 ± 0.01	0.02 ± 0.02	0.56 ± 0.02	0.12 ± 0.02	0.53 ± 0.03	0.08 ± 0.07	0.57 ± 0.00	0.25 ± 0.01
	Max	0.58 ± 0.03	0.21 ± 0.04	0.56 ± 0.04	0.13 ± 0.09	0.58 ± 0.01	0.19 ± 0.04	0.58 ± 0.02	0.20 ± 0.04	0.58 ± 0.01	0.30 ± 0.04
	MLP	0.55 ± 0.01	0.22 ± 0.01	0.55 ± 0.01	0.23 ± 0.04	0.54 ± 0.01	0.18 ± 0.04	0.56 ± 0.01	0.25 ± 0.03	0.54 ± 0.01	0.23 ± 0.02
	GRU	0.58 ± 0.00	0.26 ± 0.01	0.56 ± 0.01	0.25 ± 0.01	0.56 ± 0.01	0.23 ± 0.02	0.55 ± 0.02	0.16 ± 0.02	0.52 ± 0.02	0.09 ± 0.08
	ST	0.57 ± 0.02	0.23 ± 0.13	0.55 ± 0.03	0.22 ± 0.08	0.59 ± 0.01	0.33 ± 0.00	0.57 ± 0.01	0.27 ± 0.03	0.61 ± 0.02	0.34 ± 0.04

T.3 Social networks classification models

Appendix Table 19: Detailed metrics (mean ± standard deviation) for the social network classification datasets. 'OOM' stands for out-of-memory (RAM).

		GCN		GAT		GATv2		GIN		PNA	
Data.	Agg.	AUROC	MCC	AUROC	MCC	AUROC	MCC	AUROC	MCC	AUROC	MCC
IMDB-BINARY	Sum	0.74 ± 0.03	0.49 ± 0.07	0.73 ± 0.02	0.47 ± 0.05	0.72 ± 0.03	0.44 ± 0.06	0.74 ± 0.03	0.49 ± 0.05	0.72 ± 0.03	0.45 ± 0.05
	Mean	0.73 ± 0.04	0.46 ± 0.08	0.72 ± 0.03	0.45 ± 0.05	0.71 ± 0.03	0.42 ± 0.07	0.72 ± 0.02	0.44 ± 0.03	0.72 ± 0.03	0.46 ± 0.08
	Max	0.72 ± 0.03	0.44 ± 0.06	0.72 ± 0.04	0.44 ± 0.09	0.72 ± 0.05	0.44 ± 0.11	0.74 ± 0.02	0.47 ± 0.03	0.71 ± 0.02	0.43 ± 0.03
	MLP	0.71 ± 0.03	0.42 ± 0.07	0.70 ± 0.02	0.40 ± 0.04	0.71 ± 0.05	0.41 ± 0.10	0.71 ± 0.03	0.42 ± 0.06	0.71 ± 0.05	0.42 ± 0.09
	GRU	0.72 ± 0.04	0.45 ± 0.08	0.74 ± 0.03	0.48 ± 0.06	0.73 ± 0.02	0.46 ± 0.05	0.73 ± 0.04	0.47 ± 0.08	0.68 ± 0.04	0.36 ± 0.09
	ST	0.72 ± 0.04	0.44 ± 0.08	0.72 ± 0.04	0.45 ± 0.07	0.73 ± 0.03	0.47 ± 0.06	0.72 ± 0.06	0.45 ± 0.11	0.72 ± 0.03	0.44 ± 0.06
TWITTER-Real-Graph-Partial	Sum	0.64 ± 0.00	0.28 ± 0.00	0.62 ± 0.01	0.25 ± 0.01	0.62 ± 0.01	0.25 ± 0.01	0.62 ± 0.00	0.24 ± 0.01	0.62 ± 0.00	0.23 ± 0.01
	Mean	0.65 ± 0.01	0.29 ± 0.01	0.64 ± 0.00	0.28 ± 0.01	0.64 ± 0.00	0.29 ± 0.00	0.62 ± 0.01	0.25 ± 0.01	0.62 ± 0.01	0.23 ± 0.01
	Max	0.64 ± 0.01	0.29 ± 0.01	0.63 ± 0.00	0.27 ± 0.01	0.64 ± 0.00	0.28 ± 0.00	0.62 ± 0.01	0.25 ± 0.01	0.61 ± 0.01	0.23 ± 0.02
	MLP	0.62 ± 0.01	0.23 ± 0.01	0.61 ± 0.01	0.23 ± 0.01	0.62 ± 0.01	0.24 ± 0.01	0.62 ± 0.01	0.25 ± 0.01	0.61 ± 0.01	0.22 ± 0.01
	GRU	0.63 ± 0.01	0.28 ± 0.01	0.63 ± 0.00	0.26 ± 0.01	0.63 ± 0.00	0.26 ± 0.01	0.62 ± 0.00	0.25 ± 0.01	0.61 ± 0.01	0.23 ± 0.01
	ST	0.63 ± 0.00	0.25 ± 0.01	0.62 ± 0.00	0.24 ± 0.01	0.62 ± 0.01	0.24 ± 0.01	0.62 ± 0.01	0.24 ± 0.01	0.61 ± 0.00	0.22 ± 0.01
reddit_threads	Sum	0.78 ± 0.00	0.56 ± 0.01	0.78 ± 0.00	0.56 ± 0.01	0.77 ± 0.00	0.56 ± 0.00	0.78 ± 0.00	0.56 ± 0.00	0.77 ± 0.00	0.55 ± 0.01
	Mean	0.77 ± 0.00	0.56 ± 0.01	0.77 ± 0.00	0.55 ± 0.01	0.77 ± 0.00	0.55 ± 0.01	0.77 ± 0.00	0.55 ± 0.01	0.77 ± 0.00	0.56 ± 0.01
	Max	0.77 ± 0.00	0.55 ± 0.01	0.76 ± 0.01	0.55 ± 0.01	0.77 ± 0.00	0.55 ± 0.01	0.77 ± 0.00	0.55 ± 0.00	0.77 ± 0.00	0.55 ± 0.00
	MLP	0.75 ± 0.00	0.50 ± 0.01	0.76 ± 0.00	0.53 ± 0.01	0.76 ± 0.00	0.53 ± 0.01	0.75 ± 0.00	0.51 ± 0.01	0.75 ± 0.00	0.51 ± 0.00
	GRU	0.77 ± 0.00	0.56 ± 0.00	0.77 ± 0.00	0.56 ± 0.01	0.77 ± 0.00	0.56 ± 0.00	0.77 ± 0.00	0.55 ± 0.01	0.77 ± 0.00	0.54 ± 0.00
	ST	0.78 ± 0.00	0.56 ± 0.00	0.77 ± 0.00	0.56 ± 0.01	0.77 ± 0.00	0.55 ± 0.01	0.77 ± 0.00	0.55 ± 0.00	0.77 ± 0.00	0.55 ± 0.01
REDDIT-BINARY	Sum	0.82 ± 0.04	0.65 ± 0.08	0.71 ± 0.03	0.42 ± 0.05	0.74 ± 0.03	0.48 ± 0.06	0.80 ± 0.02	0.61 ± 0.04	OOM	OOM
	Mean	0.76 ± 0.03	0.51 ± 0.06	0.67 ± 0.01	0.36 ± 0.03	0.71 ± 0.02	0.43 ± 0.04	0.78 ± 0.04	0.57 ± 0.08	OOM	OOM
	Max	0.77 ± 0.04	0.53 ± 0.08	0.59 ± 0.04	0.26 ± 0.05	0.50 ± 0.01	0.04 ± 0.05	0.79 ± 0.05	0.58 ± 0.08	OOM	OOM
	MLP	0.84 ± 0.04	0.67 ± 0.08	0.82 ± 0.03	0.64 ± 0.06	0.80 ± 0.03	0.60 ± 0.05	0.83 ± 0.02	0.67 ± 0.05	OOM	OOM
	GRU	0.76 ± 0.02	0.52 ± 0.05	0.77 ± 0.03	0.55 ± 0.06	0.77 ± 0.01	0.53 ± 0.03	0.82 ± 0.05	0.64 ± 0.09	OOM	OOM
	ST	0.78 ± 0.06	0.58 ± 0.10	0.69 ± 0.02	0.40 ± 0.05	0.67 ± 0.03	0.36 ± 0.06	0.80 ± 0.04	0.60 ± 0.08	OOM	OOM
twitch_egos	Sum	0.69 ± 0.01	0.39 ± 0.01	0.69 ± 0.00	0.38 ± 0.01	0.69 ± 0.01	0.39 ± 0.01	0.69 ± 0.00	0.37 ± 0.01	0.69 ± 0.01	0.39 ± 0.01
	Mean	0.69 ± 0.01	0.39 ± 0.01	0.69 ± 0.00	0.39 ± 0.01	0.69 ± 0.00	0.39 ± 0.01	0.69 ± 0.01	0.38 ± 0.01	0.69 ± 0.01	0.39 ± 0.01
	Max	0.69 ± 0.01	0.39 ± 0.01	0.68 ± 0.01	0.38 ± 0.01	0.69 ± 0.00	0.38 ± 0.01	0.69 ± 0.00	0.38 ± 0.01	0.69 ± 0.01	0.39 ± 0.01
	MLP	0.63 ± 0.01	0.26 ± 0.01	0.64 ± 0.01	0.29 ± 0.02	0.65 ± 0.01	0.32 ± 0.02	0.64 ± 0.01	0.29 ± 0.01	0.63 ± 0.01	0.25 ± 0.01
	GRU	0.69 ± 0.01	0.40 ± 0.01	0.69 ± 0.00	0.39 ± 0.01	0.69 ± 0.01	0.39 ± 0.01	0.68 ± 0.01	0.37 ± 0.01	0.69 ± 0.00	0.38 ± 0.01
	ST	0.69 ± 0.01	0.39 ± 0.01	0.69 ± 0.00	0.39 ± 0.01	0.69 ± 0.01	0.39 ± 0.01	0.68 ± 0.00	0.37 ± 0.01	0.69 ± 0.01	0.39 ± 0.01
github_stargazers	Sum	0.64 ± 0.01	0.29 ± 0.02	0.61 ± 0.01	0.24 ± 0.02	0.61 ± 0.01	0.25 ± 0.02	0.61 ± 0.01	0.25 ± 0.01	0.64 ± 0.01	0.29 ± 0.02
	Mean	0.63 ± 0.01	0.28 ± 0.02	0.61 ± 0.01	0.24 ± 0.03	0.61 ± 0.00	0.24 ± 0.01	0.62 ± 0.01	0.25 ± 0.02	0.64 ± 0.01	0.29 ± 0.02
	Max	0.62 ± 0.01	0.26 ± 0.02	0.58 ± 0.02	0.21 ± 0.03	0.59 ± 0.01	0.22 ± 0.02	0.62 ± 0.01	0.24 ± 0.02	0.63 ± 0.01	0.27 ± 0.03
	MLP	0.60 ± 0.01	0.21 ± 0.02	0.60 ± 0.01	0.20 ± 0.02	0.59 ± 0.01	0.19 ± 0.02	0.63 ± 0.01	0.26 ± 0.02	0.63 ± 0.02	0.27 ± 0.04
	GRU	0.62 ± 0.02	0.25 ± 0.04	0.61 ± 0.01	0.24 ± 0.01	0.61 ± 0.01	0.24 ± 0.01	0.63 ± 0.01	0.27 ± 0.02	0.59 ± 0.02	0.18 ± 0.03
	ST	0.63 ± 0.01	0.27 ± 0.03	0.60 ± 0.02	0.23 ± 0.03	0.61 ± 0.01	0.24 ± 0.01	0.62 ± 0.02	0.26 ± 0.03	0.65 ± 0.01	0.30 ± 0.02

Appendix Table 20: Detailed metrics (mean ± sd.) for the REDDIT-MULTI-12K dataset. 'OOM' stands for out-of-memory (RAM). Only the MCC is reported as this is a multi-label task.

Dataset	Aggregator	GCN	GAT	GATv2	GIN	PNA
REDDIT-MULTI-12K	Sum	0.33 ± 0.01	0.29 ± 0.03	0.29 ± 0.03	0.33 ± 0.02	OOM
	Mean	0.28 ± 0.01	0.22 ± 0.01	0.23 ± 0.02	0.29 ± 0.01	OOM
	Max	0.28 ± 0.02	0.11 ± 0.01	0.09 ± 0.02	0.27 ± 0.01	OOM
	MLP	0.24 ± 0.01	0.25 ± 0.02	0.24 ± 0.02	0.27 ± 0.03	OOM
	GRU	0.18 ± 0.03	0.26 ± 0.07	0.25 ± 0.03	0.26 ± 0.02	OOM
	ST	0.36 ± 0.01	0.31 ± 0.02	0.30 ± 0.01	0.31 ± 0.02	OOM

T.4 Synthetic classification models

Appendix Table 21: Detailed metrics (mean ± standard deviation) for the SYNTHETIC and SYNTHETICnew datasets.

		GCN		GAT		GATv2		GIN		PNA	
Dataset	Agg.	AUROC	MCC	AUROC	MCC	AUROC	MCC	AUROC	MCC	AUROC	MCC
SYNTHETIC	Sum	0.87 ± 0.18	0.74 ± 0.36	0.99 ± 0.02	0.97 ± 0.04	0.51 ± 0.01	0.03 ± 0.07	0.94 ± 0.03	0.89 ± 0.06	1.00 ± 0.00	1.00 ± 0.00
	Mean	0.50 ± 0.00	0.00 ± 0.00	0.68 ± 0.23	0.36 ± 0.46	0.50 ± 0.00	0.00 ± 0.00	0.91 ± 0.13	0.86 ± 0.22	0.60 ± 0.22	0.20 ± 0.45
	Max	0.87 ± 0.21	0.75 ± 0.42	0.78 ± 0.25	0.59 ± 0.47	0.50 ± 0.00	0.00 ± 0.00	0.95 ± 0.04	0.91 ± 0.08	0.99 ± 0.01	0.99 ± 0.03
	MLP	0.97 ± 0.03	0.95 ± 0.06	1.00 ± 0.00	1.00 ± 0.00	0.99 ± 0.02	0.97 ± 0.04	0.97 ± 0.03	0.95 ± 0.06	0.98 ± 0.03	0.97 ± 0.06
	GRU	0.87 ± 0.21	0.75 ± 0.42	0.79 ± 0.27	0.59 ± 0.54	0.70 ± 0.27	0.40 ± 0.55	0.99 ± 0.02	0.97 ± 0.04	0.99 ± 0.02	0.99 ± 0.03
	ST	0.69 ± 0.26	0.39 ± 0.53	0.50 ± 0.00	0.00 ± 0.00	0.50 ± 0.00	0.00 ± 0.00	0.92 ± 0.05	0.86 ± 0.08	0.79 ± 0.26	0.57 ± 0.52
SYNTHETICnew	Sum	0.46 ± 0.04	-0.08 ± 0.08	0.51 ± 0.07	0.02 ± 0.15	0.51 ± 0.02	0.04 ± 0.09	0.52 ± 0.08	0.05 ± 0.16	0.76 ± 0.11	0.55 ± 0.19
	Mean	0.50 ± 0.01	-0.00 ± 0.02	0.51 ± 0.02	0.02 ± 0.03	0.50 ± 0.00	0.00 ± 0.00	0.53 ± 0.10	0.06 ± 0.20	0.59 ± 0.12	0.17 ± 0.27
	Max	0.47 ± 0.06	-0.06 ± 0.14	0.50 ± 0.04	-0.01 ± 0.16	0.50 ± 0.00	0.00 ± 0.00	0.53 ± 0.09	0.07 ± 0.19	0.91 ± 0.06	0.83 ± 0.12
	MLP	0.90 ± 0.05	0.82 ± 0.09	0.85 ± 0.06	0.73 ± 0.12	0.89 ± 0.04	0.79 ± 0.08	0.85 ± 0.06	0.71 ± 0.11	1.00 ± 0.00	1.00 ± 0.00
	GRU	0.78 ± 0.16	0.56 ± 0.33	0.77 ± 0.19	0.57 ± 0.32	0.50 ± 0.00	0.00 ± 0.00	0.82 ± 0.06	0.64 ± 0.12	0.99 ± 0.02	0.97 ± 0.04
	ST	0.51 ± 0.02	0.02 ± 0.04	0.51 ± 0.02	0.02 ± 0.03	0.50 ± 0.00	0.00 ± 0.00	0.54 ± 0.03	0.10 ± 0.07	0.78 ± 0.17	0.57 ± 0.34

Appendix Table 22: Detailed metrics (mean ± standard deviation) for the Synthie, TRIANGLES, and COLORS-3 datasets. Only the MCC is reported as this is a multi-label task.

Dataset	Aggregator	GCN	GAT	GATv2	GIN	PNA
Synthie	Sum	0.54 ± 0.06	0.52 ± 0.12	0.46 ± 0.09	0.53 ± 0.11	0.95 ± 0.03
	Mean	0.45 ± 0.11	0.36 ± 0.21	0.52 ± 0.11	0.58 ± 0.07	0.93 ± 0.05
	Max	0.14 ± 0.04	0.24 ± 0.07	0.23 ± 0.07	0.30 ± 0.14	0.59 ± 0.04
	MLP	0.71 ± 0.08	0.72 ± 0.13	0.75 ± 0.05	0.59 ± 0.08	0.93 ± 0.03
	GRU	0.75 ± 0.12	0.48 ± 0.21	0.79 ± 0.11	0.72 ± 0.03	0.79 ± 0.08
	ST	0.37 ± 0.23	0.61 ± 0.08	0.71 ± 0.11	0.60 ± 0.09	0.88 ± 0.12
TRIANGLES	Sum	0.11 ± 0.01	0.18 ± 0.01	0.17 ± 0.01	0.13 ± 0.01	0.10 ± 0.04
	Mean	0.14 ± 0.02	0.17 ± 0.01	0.18 ± 0.02	0.12 ± 0.01	0.11 ± 0.02
	Max	0.12 ± 0.01	0.21 ± 0.01	0.21 ± 0.01	0.14 ± 0.02	0.09 ± 0.01
	MLP	0.10 ± 0.01	0.13 ± 0.01	0.10 ± 0.01	0.10 ± 0.01	0.10 ± 0.05
	GRU	0.10 ± 0.02	0.13 ± 0.04	0.15 ± 0.03	0.12 ± 0.01	0.10 ± 0.02
	ST	0.13 ± 0.01	0.17 ± 0.01	0.20 ± 0.02	0.11 ± 0.02	0.12 ± 0.02
COLORS-3	Sum	0.98 ± 0.00	0.37 ± 0.04	0.38 ± 0.03	0.89 ± 0.01	1.00 ± 0.00
	Mean	0.28 ± 0.04	0.28 ± 0.02	0.26 ± 0.02	0.32 ± 0.01	0.38 ± 0.02
	Max	0.41 ± 0.01	0.20 ± 0.05	0.20 ± 0.05	0.53 ± 0.02	0.59 ± 0.02
	MLP	0.58 ± 0.01	0.48 ± 0.01	0.51 ± 0.02	0.41 ± 0.02	0.55 ± 0.04
	GRU	0.91 ± 0.01	0.59 ± 0.06	0.52 ± 0.11	0.79 ± 0.02	1.00 ± 0.00
	ST	0.83 ± 0.03	0.57 ± 0.06	0.52 ± 0.08	0.76 ± 0.05	0.92 ± 0.04

T.5 Bioinformatics classification models

Appendix Table 23: Detailed metrics (mean ± standard deviation) for the PROTEINS_full dataset.

		GCN		GAT		GATv2		GIN		PNA	
Dataset	Agg.	AUROC	MCC	AUROC	MCC	AUROC	MCC	AUROC	MCC	AUROC	MCC
PROTEINS_full	Sum	0.67 ± 0.06	0.38 ± 0.13	0.52 ± 0.03	0.05 ± 0.07	0.57 ± 0.09	0.14 ± 0.17	0.55 ± 0.04	0.18 ± 0.12	0.64 ± 0.04	0.33 ± 0.07
	Mean	0.69 ± 0.06	0.41 ± 0.11	0.67 ± 0.06	0.35 ± 0.10	0.69 ± 0.07	0.39 ± 0.15	0.52 ± 0.02	0.09 ± 0.11	0.71 ± 0.06	0.47 ± 0.09
	Max	0.69 ± 0.04	0.41 ± 0.08	0.51 ± 0.02	0.06 ± 0.06	0.53 ± 0.03	0.13 ± 0.10	0.57 ± 0.06	0.21 ± 0.16	0.72 ± 0.02	0.44 ± 0.05
	MLP	0.62 ± 0.11	0.26 ± 0.24	0.58 ± 0.12	0.17 ± 0.23	0.59 ± 0.11	0.21 ± 0.22	0.66 ± 0.05	0.33 ± 0.09	0.67 ± 0.06	0.35 ± 0.13
	GRU	0.69 ± 0.02	0.39 ± 0.05	0.71 ± 0.02	0.43 ± 0.04	0.71 ± 0.01	0.43 ± 0.03	0.66 ± 0.03	0.39 ± 0.06	0.65 ± 0.05	0.33 ± 0.10
	ST	0.70 ± 0.04	0.42 ± 0.07	0.67 ± 0.05	0.34 ± 0.09	0.70 ± 0.05	0.39 ± 0.09	0.60 ± 0.06	0.30 ± 0.12	0.62 ± 0.06	0.33 ± 0.15

Appendix Table 24: Detailed metrics (mean ± standard deviation) for the ENZYMES dataset. Only the MCC is reported as this is a multi-label task.

Dataset	Aggregator	GCN	GAT	GATv2	GIN	PNA
ENZYMES	Sum	0.16 ± 0.06	0.21 ± 0.14	0.27 ± 0.05	0.50 ± 0.07	0.42 ± 0.08
	Mean	0.16 ± 0.09	0.24 ± 0.10	0.32 ± 0.15	0.44 ± 0.06	0.39 ± 0.11
	Max	0.20 ± 0.10	0.28 ± 0.11	0.25 ± 0.14	0.41 ± 0.05	0.40 ± 0.11
	MLP	0.47 ± 0.08	0.53 ± 0.04	0.52 ± 0.07	0.48 ± 0.03	0.51 ± 0.07
	GRU	0.24 ± 0.07	0.32 ± 0.10	0.27 ± 0.13	0.38 ± 0.09	0.17 ± 0.05
	ST	0.23 ± 0.12	0.25 ± 0.13	0.28 ± 0.07	0.37 ± 0.04	0.29 ± 0.10

T.6 TUDataset computer vision models

Appendix Table 25: Detailed metrics (mean ± standard deviation) for the TUDataset computer vision datasets. Only the MCC is reported as this is a multi-label task.

Dataset	Aggregator	GCN	GAT	GATv2	GIN	PNA
Cuneiform	Sum	0.51 ± 0.29	0.56 ± 0.25	0.66 ± 0.09	0.73 ± 0.07	0.58 ± 0.34
	Mean	0.00 ± 0.02	0.15 ± 0.25	-0.05 ± 0.04	0.80 ± 0.09	0.03 ± 0.06
	Max	0.26 ± 0.35	0.36 ± 0.33	0.26 ± 0.35	0.77 ± 0.12	0.55 ± 0.30
	MLP	0.50 ± 0.13	0.58 ± 0.17	0.62 ± 0.11	0.64 ± 0.09	0.64 ± 0.11
	GRU	0.18 ± 0.08	0.05 ± 0.09	0.11 ± 0.15	0.51 ± 0.12	0.16 ± 0.14
	ST	0.32 ± 0.19	0.21 ± 0.32	0.40 ± 0.25	0.47 ± 0.15	0.40 ± 0.31
COIL-RAG	Sum	0.91 ± 0.01	0.94 ± 0.02	0.94 ± 0.02	0.95 ± 0.02	0.90 ± 0.02
	Mean	0.89 ± 0.02	0.90 ± 0.02	0.93 ± 0.01	0.94 ± 0.02	0.90 ± 0.02
	Max	0.90 ± 0.02	0.91 ± 0.02	0.93 ± 0.01	0.94 ± 0.02	0.91 ± 0.02
	MLP	0.96 ± 0.01	0.96 ± 0.01	0.95 ± 0.01	0.94 ± 0.02	0.93 ± 0.02
	GRU	0.68 ± 0.11	0.76 ± 0.12	0.72 ± 0.04	0.85 ± 0.04	0.74 ± 0.07
	ST	0.82 ± 0.03	0.82 ± 0.04	0.84 ± 0.04	0.90 ± 0.02	0.82 ± 0.04
COIL-DEL	Sum	0.25 ± 0.04	0.50 ± 0.05	0.61 ± 0.06	0.49 ± 0.04	0.72 ± 0.03
	Mean	0.28 ± 0.02	0.42 ± 0.03	0.57 ± 0.03	0.40 ± 0.05	0.67 ± 0.05
	Max	0.30 ± 0.02	0.44 ± 0.03	0.56 ± 0.03	0.47 ± 0.05	0.70 ± 0.02
	MLP	0.42 ± 0.02	0.58 ± 0.04	0.61 ± 0.04	0.45 ± 0.03	0.55 ± 0.04
	GRU	0.15 ± 0.02	0.33 ± 0.06	0.38 ± 0.04	0.28 ± 0.03	0.23 ± 0.03
	ST	0.49 ± 0.03	0.31 ± 0.04	0.42 ± 0.08	0.40 ± 0.09	0.65 ± 0.03

T.7 ZINC regression models

Appendix Table 26: Detailed metrics for the ZINC dataset. The results are reported only for the provided train/validation/test splits (no custom random splits).

		GCN		**GAT**		**GATv2**		**GIN**		**PNA**	
Dataset	Agg.	**MAE**	**R^2**	**MAE**	**R^2**	**MAE**	**R^2**	**MAE**	**R^2**	**MAE**	**R^2**
ZINC	Sum	0.80	0.59	0.86	0.59	1.06	0.47	0.51	0.80	0.36	0.87
	Mean	0.74	0.63	0.90	0.54	1.26	0.43	0.57	0.77	0.39	0.86
	Max	0.88	0.57	1.25	0.50	1.34	0.39	0.64	0.72	0.47	0.86
	MLP	0.68	0.69	0.79	0.60	1.49	0.42	0.50	0.79	0.36	0.88
	GRU	0.92	0.51	1.46	0.06	0.91	0.67	0.56	0.79	0.33	0.89
	ST	0.72	0.64	1.04	0.47	2.39	0.42	0.53	0.79	0.43	0.85

T.8 QM9 regression models with Janossy neural aggregation

Appendix Table 27: Detailed metrics (mean ± standard deviation) for the QM9 dataset, including the two Janossy variants. Any differences in performance compared to other 2-layer models such as those in Figure 3 might be due to different GNN hyperparameters, such as the output or intermediate node dimension (QM9-specific experiments generally used larger dimensions).

		GCN		**GAT**		**GATv2**		**GIN**		**PNA**	
Data.	Agg.	**MAE**	**R^2**	**MAE**	**R^2**	**MAE**	**R^2**	**MAE**	**R^2**	**MAE**	**R^2**
QM9	Sum	0.74 ± 0.00	0.09 ± 0.00	0.76 ± 0.00	0.05 ± 0.01	0.75 ± 0.00	0.08 ± 0.01	0.71 ± 0.00	0.15 ± 0.00	0.70 ± 0.00	0.17 ± 0.01
	Mean	0.73 ± 0.00	0.10 ± 0.00	0.75 ± 0.01	0.07 ± 0.01	0.74 ± 0.00	0.10 ± 0.00	0.72 ± 0.00	0.13 ± 0.00	0.70 ± 0.00	0.16 ± 0.01
	Max	0.73 ± 0.00	0.11 ± 0.00	0.74 ± 0.00	0.10 ± 0.00	0.73 ± 0.00	0.11 ± 0.00	0.70 ± 0.00	0.16 ± 0.00	0.68 ± 0.00	0.20 ± 0.00
	MLP	0.63 ± 0.00	0.31 ± 0.00	0.64 ± 0.00	0.30 ± 0.01	0.64 ± 0.01	0.29 ± 0.01	0.60 ± 0.00	0.38 ± 0.00	0.58 ± 0.00	0.41 ± 0.00
	GRU	0.62 ± 0.01	0.34 ± 0.02	0.62 ± 0.01	0.34 ± 0.01	0.61 ± 0.02	0.35 ± 0.03	0.60 ± 0.00	0.38 ± 0.00	0.60 ± 0.01	0.37 ± 0.02
	ST	0.60 ± 0.01	0.38 ± 0.01	0.63 ± 0.02	0.30 ± 0.04	0.62 ± 0.01	0.32 ± 0.02	0.59 ± 0.00	0.39 ± 0.01	0.57 ± 0.01	0.44 ± 0.01
	Janossy MLP	0.67 ± 0.02	0.23 ± 0.04	0.68 ± 0.01	0.22 ± 0.03	0.67 ± 0.01	0.23 ± 0.02	0.61 ± 0.01	0.34 ± 0.02	0.61 ± 0.01	0.34 ± 0.02
	Janossy GRU	0.67 ± 0.02	0.22 ± 0.04	0.67 ± 0.01	0.23 ± 0.02	0.71 ± 0.02	0.15 ± 0.04	0.66 ± 0.03	0.25 ± 0.06	0.63 ± 0.01	0.32 ± 0.02

T.9 MalNetTiny models

Appendix Table 28: Detailed metrics (mean ± standard deviation) for the MalNetTiny dataset. 'OOM' stands for out-of-memory (RAM). Only the MCC is reported as this is a multi-label task.

Dataset	Aggregator	**GCN**	**GAT**	**GATv2**	**GIN**	**PNA**
MalNetTiny	Sum	0.81 ± 0.04	0.81 ± 0.05	0.80 ± 0.02	0.87 ± 0.03	OOM
	Mean	0.72 ± 0.02	0.73 ± 0.02	0.75 ± 0.05	0.88 ± 0.01	OOM
	Max	0.81 ± 0.02	0.77 ± 0.04	0.76 ± 0.04	0.89 ± 0.02	OOM
	MLP	0.80 ± 0.02	0.81 ± 0.01	0.80 ± 0.01	0.82 ± 0.02	OOM
	GRU	0.68 ± 0.03	0.69 ± 0.04	0.70 ± 0.02	0.70 ± 0.02	OOM
	ST	0.84 ± 0.03	0.82 ± 0.03	0.84 ± 0.02	0.88 ± 0.03	OOM

T.10 GNNBenchmark computer vision models

Appendix Table 29: Detailed metrics for the MNIST and CIFAR10 datasets. All models use the provided train/validation/test splits (no custom splits), which are used in five different runs and aggregated (mean ± standard deviation). Only the MCC is reported as this is a multi-label task.

Dataset	Aggregator	**GCN**	**GAT**	**GATv2**	**GIN**	**PNA**
MNIST	Sum	0.42 ± 0.01	0.35 ± 0.04	0.52 ± 0.03	0.51 ± 0.01	0.74 ± 0.00
	Mean	0.39 ± 0.01	0.29 ± 0.01	0.33 ± 0.07	0.46 ± 0.01	0.72 ± 0.01
	Max	0.25 ± 0.03	0.40 ± 0.05	0.43 ± 0.17	0.43 ± 0.01	0.71 ± 0.01
	MLP	0.28 ± 0.02	0.28 ± 0.01	0.26 ± 0.04	0.31 ± 0.01	0.47 ± 0.03
	GRU	0.36 ± 0.03	0.25 ± 0.04	0.37 ± 0.09	0.44 ± 0.02	0.66 ± 0.02
	ST	0.48 ± 0.01	0.60 ± 0.03	0.64 ± 0.02	0.56 ± 0.01	0.77 ± 0.00
CIFAR10	Sum	0.28 ± 0.01	0.32 ± 0.01	0.34 ± 0.03	0.31 ± 0.01	0.50 ± 0.00
	Mean	0.27 ± 0.00	0.29 ± 0.01	0.30 ± 0.01	0.30 ± 0.01	0.51 ± 0.00
	Max	0.29 ± 0.00	0.36 ± 0.00	0.35 ± 0.01	0.31 ± 0.00	0.46 ± 0.01
	MLP	0.17 ± 0.00	0.25 ± 0.00	0.19 ± 0.02	0.17 ± 0.00	0.29 ± 0.01
	GRU	0.24 ± 0.01	0.31 ± 0.02	0.29 ± 0.07	0.26 ± 0.01	0.45 ± 0.01
	ST	0.32 ± 0.01	0.42 ± 0.01	0.39 ± 0.01	0.35 ± 0.00	0.48 ± 0.00

T.11 TUDataset small molecules models

Appendix Table 30: Detailed metrics (mean ± standard deviation) for the AIDS, FRANKENSTEIN, MUTAG, and Mutagenicity datasets.

		GCN		GAT		GATv2		GIN		PNA	
Dataset	Agg.	AUROC	MCC	AUROC	MCC	AUROC	MCC	AUROC	MCC	AUROC	MCC
AIDS	Sum	0.97 ± 0.02	0.96 ± 0.03	0.97 ± 0.03	0.95 ± 0.04	0.97 ± 0.02	0.95 ± 0.02	0.96 ± 0.02	0.95 ± 0.02	0.97 ± 0.02	0.96 ± 0.02
	Mean	0.97 ± 0.02	0.95 ± 0.02	0.98 ± 0.02	0.96 ± 0.03	0.97 ± 0.02	0.96 ± 0.03	0.96 ± 0.02	0.95 ± 0.02	0.97 ± 0.02	0.96 ± 0.02
	Max	0.97 ± 0.02	0.96 ± 0.02	0.97 ± 0.02	0.96 ± 0.02	0.97 ± 0.02	0.94 ± 0.03	0.97 ± 0.01	0.95 ± 0.01	0.97 ± 0.02	0.96 ± 0.03
	MLP	1.00 ± 0.00	1.00 ± 0.01	1.00 ± 0.00	1.00 ± 0.01	1.00 ± 0.00	1.00 ± 0.01	1.00 ± 0.00	1.00 ± 0.01	1.00 ± 0.00	1.00 ± 0.01
	GRU	1.00 ± 0.00	1.00 ± 0.01	1.00 ± 0.00	1.00 ± 0.01	1.00 ± 0.00	1.00 ± 0.01	1.00 ± 0.01	0.99 ± 0.01	1.00 ± 0.00	0.99 ± 0.01
	ST	0.97 ± 0.02	0.96 ± 0.03	0.98 ± 0.02	0.96 ± 0.03	0.97 ± 0.02	0.95 ± 0.03	0.96 ± 0.02	0.89 ± 0.09	0.98 ± 0.02	0.97 ± 0.03
FRANKENSTEIN	Sum	0.62 ± 0.01	0.25 ± 0.02	0.60 ± 0.00	0.21 ± 0.02	0.59 ± 0.02	0.19 ± 0.03	0.60 ± 0.03	0.20 ± 0.06	0.65 ± 0.03	0.29 ± 0.06
	Mean	0.60 ± 0.03	0.21 ± 0.06	0.60 ± 0.02	0.20 ± 0.04	0.59 ± 0.03	0.19 ± 0.05	0.61 ± 0.04	0.22 ± 0.08	0.62 ± 0.03	0.25 ± 0.04
	Max	0.60 ± 0.02	0.20 ± 0.03	0.60 ± 0.02	0.20 ± 0.04	0.61 ± 0.03	0.21 ± 0.06	0.62 ± 0.02	0.23 ± 0.05	0.63 ± 0.03	0.25 ± 0.05
	MLP	0.61 ± 0.02	0.23 ± 0.03	0.63 ± 0.02	0.27 ± 0.05	0.65 ± 0.02	0.29 ± 0.05	0.65 ± 0.02	0.30 ± 0.05	0.68 ± 0.01	0.37 ± 0.03
	GRU	0.66 ± 0.03	0.32 ± 0.07	0.66 ± 0.01	0.32 ± 0.02	0.65 ± 0.02	0.29 ± 0.04	0.62 ± 0.03	0.23 ± 0.05	0.58 ± 0.03	0.17 ± 0.07
	ST	0.61 ± 0.03	0.22 ± 0.06	0.59 ± 0.02	0.17 ± 0.04	0.58 ± 0.02	0.16 ± 0.04	0.62 ± 0.02	0.24 ± 0.04	0.63 ± 0.02	0.26 ± 0.04
MUTAG	Sum	0.65 ± 0.18	0.30 ± 0.30	0.66 ± 0.17	0.33 ± 0.28	0.67 ± 0.16	0.36 ± 0.26	0.90 ± 0.08	0.75 ± 0.17	0.84 ± 0.13	0.67 ± 0.23
	Mean	0.68 ± 0.19	0.32 ± 0.30	0.69 ± 0.19	0.35 ± 0.32	0.70 ± 0.20	0.37 ± 0.34	0.90 ± 0.07	0.78 ± 0.14	0.79 ± 0.12	0.53 ± 0.19
	Max	0.71 ± 0.23	0.38 ± 0.42	0.72 ± 0.24	0.40 ± 0.43	0.65 ± 0.22	0.29 ± 0.40	0.91 ± 0.07	0.78 ± 0.14	0.90 ± 0.06	0.75 ± 0.18
	MLP	0.91 ± 0.12	0.84 ± 0.24	0.90 ± 0.15	0.80 ± 0.30	0.84 ± 0.07	0.67 ± 0.14	0.87 ± 0.13	0.71 ± 0.26	0.89 ± 0.11	0.75 ± 0.22
	GRU	0.81 ± 0.19	0.55 ± 0.32	0.78 ± 0.17	0.51 ± 0.29	0.87 ± 0.06	0.71 ± 0.09	0.84 ± 0.14	0.65 ± 0.26	0.88 ± 0.07	0.76 ± 0.10
	ST	0.85 ± 0.08	0.68 ± 0.17	0.85 ± 0.11	0.69 ± 0.18	0.83 ± 0.12	0.69 ± 0.15	0.89 ± 0.09	0.72 ± 0.19	0.83 ± 0.09	0.67 ± 0.13
Mutagenicity	Sum	0.79 ± 0.02	0.59 ± 0.03	0.76 ± 0.01	0.53 ± 0.02	0.79 ± 0.01	0.57 ± 0.02	0.81 ± 0.02	0.63 ± 0.04	0.82 ± 0.02	0.64 ± 0.05
	Mean	0.78 ± 0.02	0.56 ± 0.04	0.77 ± 0.01	0.55 ± 0.03	0.75 ± 0.01	0.52 ± 0.03	0.81 ± 0.01	0.61 ± 0.03	0.80 ± 0.03	0.60 ± 0.06
	Max	0.79 ± 0.02	0.57 ± 0.03	0.74 ± 0.01	0.50 ± 0.02	0.71 ± 0.01	0.44 ± 0.03	0.83 ± 0.01	0.65 ± 0.02	0.81 ± 0.02	0.62 ± 0.04
	MLP	0.73 ± 0.02	0.48 ± 0.03	0.77 ± 0.02	0.53 ± 0.03	0.76 ± 0.01	0.52 ± 0.02	0.79 ± 0.01	0.57 ± 0.02	0.78 ± 0.01	0.56 ± 0.02
	GRU	0.73 ± 0.01	0.47 ± 0.02	0.75 ± 0.01	0.49 ± 0.02	0.74 ± 0.01	0.48 ± 0.03	0.81 ± 0.02	0.61 ± 0.04	0.73 ± 0.02	0.46 ± 0.04
	ST	0.80 ± 0.02	0.59 ± 0.03	0.75 ± 0.04	0.52 ± 0.06	0.78 ± 0.03	0.57 ± 0.05	0.81 ± 0.01	0.63 ± 0.02	0.82 ± 0.01	0.64 ± 0.02
YeastH	Sum	0.54 ± 0.01	0.20 ± 0.02	0.54 ± 0.01	0.20 ± 0.02	0.54 ± 0.00	0.21 ± 0.02	0.57 ± 0.01	0.27 ± 0.03	0.58 ± 0.01	0.29 ± 0.02
	Mean	0.52 ± 0.00	0.16 ± 0.01	0.52 ± 0.00	0.17 ± 0.01	0.53 ± 0.00	0.17 ± 0.01	0.55 ± 0.01	0.24 ± 0.02	0.56 ± 0.01	0.25 ± 0.01
	Max	0.53 ± 0.00	0.17 ± 0.01	0.51 ± 0.00	0.14 ± 0.02	0.52 ± 0.00	0.16 ± 0.01	0.57 ± 0.01	0.26 ± 0.03	0.57 ± 0.01	0.25 ± 0.03
	MLP	0.60 ± 0.00	0.24 ± 0.01	0.60 ± 0.01	0.24 ± 0.01	0.60 ± 0.01	0.25 ± 0.02	0.61 ± 0.01	0.26 ± 0.02	0.60 ± 0.01	0.24 ± 0.02
	GRU	0.54 ± 0.02	0.17 ± 0.02	0.55 ± 0.01	0.19 ± 0.01	0.54 ± 0.01	0.19 ± 0.02	0.58 ± 0.02	0.26 ± 0.02	0.55 ± 0.01	0.17 ± 0.01
	ST	0.62 ± 0.00	0.31 ± 0.01	0.59 ± 0.02	0.30 ± 0.01	0.60 ± 0.01	0.30 ± 0.02	0.64 ± 0.01	0.34 ± 0.02	0.64 ± 0.01	0.35 ± 0.02

Appendix Table 31: Detailed metrics (mean ± standard deviation) for the alchemy_full dataset.

		GCN		GAT		GATv2		GIN		PNA	
Dataset	Agg.	MAE	R^2	MAE	R^2	MAE	R^2	MAE	R^2	MAE	R^2
alchemy_full	Sum	17.78 ± 0.48	0.99 ± 0.00	46.80 ± 2.39	0.98 ± 0.00	46.00 ± 4.02	0.98 ± 0.00	17.49 ± 1.04	0.99 ± 0.00	10.30 ± 0.42	1.00 ± 0.00
	Mean	28.91 ± 0.74	0.97 ± 0.00	56.22 ± 2.65	0.97 ± 0.00	49.94 ± 1.14	0.97 ± 0.00	41.46 ± 4.19	0.97 ± 0.00	16.71 ± 0.50	0.99 ± 0.00
	Max	18.90 ± 0.24	0.99 ± 0.00	78.62 ± 9.31	0.99 ± 0.01	67.37 ± 3.42	0.99 ± 0.00	31.51 ± 9.16	0.99 ± 0.00	13.80 ± 0.99	1.00 ± 0.00
	MLP	12.75 ± 0.74	1.00 ± 0.00	15.87 ± 1.33	1.00 ± 0.00	17.76 ± 1.88	1.00 ± 0.00	20.24 ± 7.66	0.99 ± 0.00	10.14 ± 1.75	1.00 ± 0.00
	GRU	9.25 ± 0.58	1.00 ± 0.00	16.16 ± 1.07	1.00 ± 0.00	15.23 ± 1.54	1.00 ± 0.00	13.35 ± 0.61	0.99 ± 0.00	7.24 ± 0.40	1.00 ± 0.00
	ST	9.58 ± 0.38	1.00 ± 0.00	19.52 ± 8.84	0.99 ± 0.01	15.11 ± 1.27	1.00 ± 0.00	13.83 ± 1.62	1.00 ± 0.00	10.25 ± 1.19	1.00 ± 0.00