Graph Neural Networks with Adaptive Readouts

Anonymous Author(s)
Affiliation
Address
email

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., molecules, 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 [e.g., see 1–5]. Graph neural networks typically employ a permutation invariant neighborhood aggregation scheme (or a message passing algorithm) that is repeated for several iterations, where in each iteration node representations are updated by aggregating the feature vectors corresponding to their neighbors, with the latter features generated by the previous iteration of the algorithm. 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 the 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, max, and mean all satisfy this requirement [6, 7]. More specifically, Zaheer et al. [6] have studied the problem of learning on sets and shown 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

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

neural networks specifically has also been studied in [5], where it has been recommended to use an injective neighborhood aggregation scheme. 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 effective and computationally efficient for a large number of graph isomorphism classes [8].

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 its 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 our presentation with a review of graph neural networks and then focus on introducing different classes of adaptive and differentiable readout functions. The first class of readouts is based on set transformers [9] and it gives rise to permutation invariant hypotheses. We then slightly relax the constraint on permutation invariance of graph-level representations by introducing variations to the Janossy pooling [10] based on multi-layer perceptrons (MLP) and recurrent neural architectures known as GRUs. This type of readouts can be considered approximately permutation invariant. 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 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 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 the 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 graph neural networks 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.

Our empirical study concludes with an experiment involving a multi-million scale proprietary dataset from a pharma company that has been collected by primary screening assays. The empirical evidence again demonstrates that the avenue of neural readouts merits further exploration from both theoretical and empirical perspectives. 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 M. A time and memory analysis is available 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, node features X_G and 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:

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

where $h_u^{(k)}$ is a representation vector for node $u \in \mathcal{V}$ generated at the output of the kth iteration/layer. For example, in graph convolutional neural 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 kth iteration/layer. Alternatively, Xu et al. [5] perform neighborhood aggregation using a multi-layer perceptron:

$$h_v^{(k)} = \textsc{mlp}^{(k)}\left((1+\epsilon^{(k)})h_v^{(k-1)} + \sum_{u \in \mathcal{N}_v} h_u^{(k-1)}\right),$$

where $\epsilon^{(k)}$ is an optimization parameter/scalar learned jointly with the MLP weights. The neighborhood aggregation schemes are typically initialized by setting $h_v^{(0)} = x_v$ for all $v \in \mathcal{V}$.

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 $h \in \mathbb{R}^{MD}$ the flattened node feature matrix H.

Set Transformer Readouts. Recently an attention-based neural architecture for learning on sets has been proposed in [9]. The main difference compared to the classical attention model proposed by Vaswani et al. [11] 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, we provide a more detailed review of attention models in Appendix D and summarize here the adaptation to sets. The set transformers take as input matrices with items/nodes as rows and generate graph readouts by composing the attention-based encoder and decoder modules, i.e.,

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

where $[\cdot]_k$ refers to computation specific to head k. The encoder-decoder modules are given by [9]:

$$\begin{aligned}
&\text{ENCODER}\left(H\right) := \text{SAB}^n\left(H\right) \quad \text{and} \quad \text{DECODER}(Z) := \text{FF}\left(\text{SAB}^m\left(\text{PMA}(Z)\right)\right), \\
&\text{where} \quad \text{SAB}(H) := \text{MAB}(H, H) \quad \text{and} \quad \text{PMA}(Z) := \text{MAB}(s, \text{FF}(Z)), \\
&\text{with} \quad \text{MAB}(U, V) := A + \text{FF}(A) \quad \text{and} \quad A = U + \text{MULTI-HEAD}(U, V, V)\,.
\end{aligned} \tag{3}$$

Here, H denotes the node features after neighborhood aggregation and Z is the encoder output. The encoder is a chain of n self-attention blocks (SAB) that are based on 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 [10] 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:

$$\text{JANOSSY-DENSE}(H) := \frac{1}{p}\sum_{i=1}^{p}\text{DENSE}(h_{\pi_i}) \quad \text{or} \quad \text{JANOSSY-GRU}(H) := \frac{1}{p}\sum_{i=1}^{p}\text{GRU}(H_{\pi_i}), \tag{4}$$

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 DENSE 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:

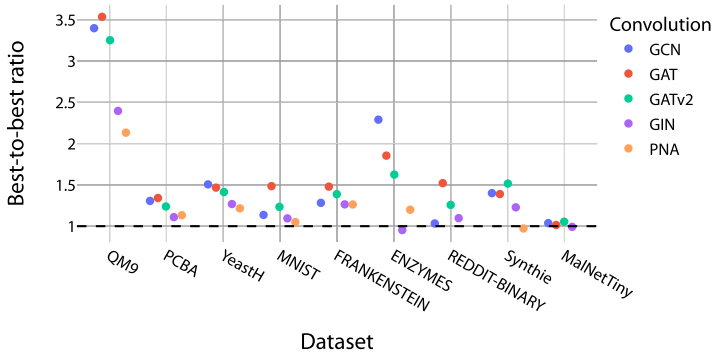
$$\text{DENSE}(H) := \text{RELU}(\text{BN}_2(W_2 z_1 + b_2)) \quad \text{with} \quad z_1 = \text{RELU}(\text{BN}_1(W_1 h + b_1)) \tag{5}$$

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 DENSE. The GRU neural readout is composed of a single-layer, unidirectional gated recurrent unit (GRU, [12]), 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.

3 Experiments

We perform a series of experiments to evaluate the effectiveness of adaptive and differentiable neural readouts presented in Section 2.1 relative to the standard pooling functions used for the aggregation of node features into a graph-level representation (i.e., sum, max, mean). We focus on quantifying the

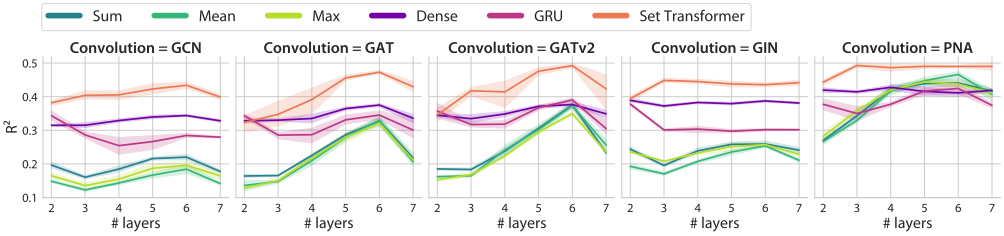
Figure 1: The performance of the best neural relative to the best standard readout on a collection of representative datasets from different domains. We use the ratio between the effectiveness scores (R^2 for QM9 and Matthew correlation coefficient otherwise), computed by averaging over five random splits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



difference in performance relative to different factors such as: *i*) most frequently used neighborhood aggregations schemes, *ii*) the number of neighborhood iterations that correspond to layers in graph neural networks, *iii*) convergence rates measured in the number of epochs required for training an effective model, and *iv*) different graph characteristics and domains from which the structured data originates. Moreover, we also evaluate a variety of neighborhood aggregation and readout schemes using a large scale proprietary dataset 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 small scale benchmarks typically used for evaluation of their effectiveness in drug design tasks. 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. In total, we use in excess of 40 datasets to showcase the potential of neural readouts for learning effective models over graph spaces. The datasets originate from different domains such as quantum mechanics, biophysics, bioinformatics, computer vision, social networks, synthetic graphs, and function call graphs. The benchmarks differ in the number of instances and graph structure (e.g., average number of nodes and edges per graph, node degree distributions, etc.).

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 (in total 39 of them, 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 [13], GIN [5], and PNA [14]. 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 nine 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 standard readouts. 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 (dataset, convolution) out of 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% relative). In classification tasks, we use the Matthew correlation coefficient (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 datasets ($\approx 63\%$ of time). We also observe that on 48 datasets (three models also failed to complete due to memory issues when using PNA) where standard readouts are more effective than the neural ones that the relative degradation is below 10% relative. 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 optimized in our experiments. Thus, it is likely that the performance can be further improved by hyperparameter tuning.

Figure 2: 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.



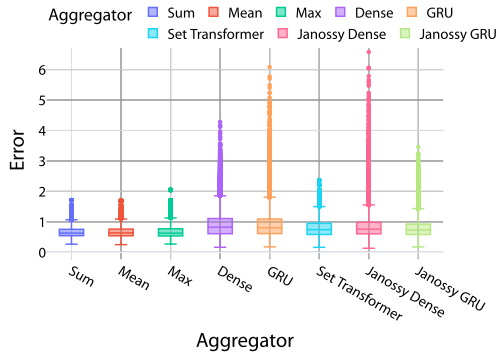
Neural vs standard readouts relative to the number of neighborhood aggregations iterations. The goal of this experiment is two-fold: *i*) assessment of the effectiveness of neural readouts relative to the depth of graph neural networks, and *ii*) validation of the observations in the experiments with 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 simulate the experiment with different graph convolutional operators using datasets with different number of instances, ranging from 600 to $132,480$ graphs on ENZYMES and QM9 datasets, respectively. Moreover, we also use a proprietary bio-affinity dataset with 1.5 million graphs (see Appendix O, Figure 19). Figure 2 summarizes the results of a representative experiment (more results can be found in Appendix H, Figure 4). 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 could 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 embedding (i.e., the first epoch) and all subsequent epochs (Appendix L, Figure 11), as well as between consecutive epochs (Appendix L, 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.

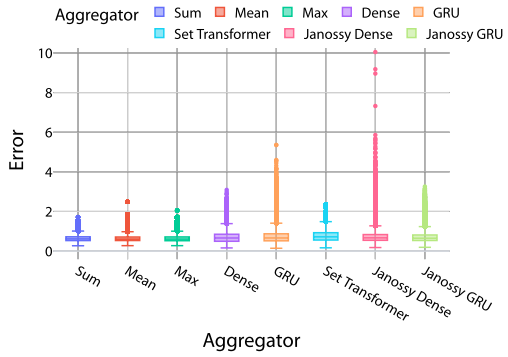
The importance of (approximate) permutation invariance in adaptive readouts. The goal of this experiment is to obtain an insight into the importance of highly expressive adaptive readouts that give rise to permutation invariant hypothesis spaces on the network's ability to learn a target concept. To this end, we exploit the modular design of SET TRANSFORMERS and consider architectures with 1 to 12 attention heads (Appendix J, Table 17). With a small number of attention heads (1 and 4), graph neural networks with SET TRANSFORMER readouts are able to outperform the standard pooling functions — sum, max, and mean. However, over different convolutions these models are outperformed by DENSE and GRU readouts that do not enforce permutation invariance. Increasing the number of heads to 8 or 12 leads to the best performance of SET TRANSFORMER readouts. However, the relative improvement that comes as a result of this compared to the setting with a small number of attention heads is minor, with the exception of GAT and GATV2 neighborhood aggregation schemes.

Figure 3: 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.



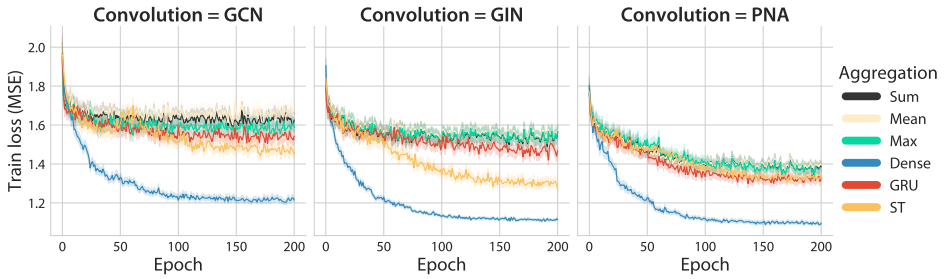
(b) For each molecule, we generate graphs corresponding to different non-canonical SMILES. The number varies from 20 to 1,000 (per-molecule).



Thus, we hypothesize that the network is able to extract better node features via DENSE and GRU readouts that do not enforce permutation invariance. We also evaluated the impact of enforcing approximate permutation invariance by Janossy readouts on the QM9 dataset (Appendix J, Table 16). The Janossy variants presented in Section 2.1 outperform the three standard readouts in both mean average error and R^2, but they score lower than the other neural readouts.

Robustness to node permutations for non-permutation invariant readouts. The goal of this experiment is to evaluate the stability of readouts that either do not enforce permutation invariance of the hypothesis space (i.e., DENSE and GRU readouts) or do that 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 layers (see above for details, neural vs standard readouts). In addition to this, we also generate permuted graphs by using non-canonical SMILES. For the latter, we have tried repeated sampling of permuted molecular graphs until the total number of them converged. 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 3a and 3b summarize the results of this experiment and show that the plain readouts (i.e., DENSE and GRU) can be negatively affected by some random permutations. Interestingly, the error for DENSE 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 K, Figure 9 for more details). The Janossy readouts are set to train not only on the original fixed-order graphs but also on 25 random node permutations computed during training. 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 to this, the JANOSSY DENSE 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 K, 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 DENSE readouts (see Appendix K, Figure 8).

Figure 4: Train loss for the VGAE models trained on proprietary datasets with ≈ 1 million molecular graphs.



The effectiveness of graph neural networks and readouts on multi-million graph datasets. In this experiment, we extend our empirical analysis with three proprietary high-throughput screening assays against varied protein targets, with approximately 1-2 million graphs per target (see Appendix A, Table 1). We trained both non-variational and guided variational graph autoencoder [15, 16] 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 4, with the other 5 models in Appendix N). Our analysis considers the performance relative to different neighborhood aggregation schemes or convolutional operators as well the total number of such iterations that correspond to depth in graph neural networks (separately, see Appendix O, Figure 19). Figure 4 summarizes the results of this experiment and indicates that the models using standard non-adaptive readouts (sum, 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 neighborhood aggregation scheme, the DENSE 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 offers only a slight improvement compared to the other two neural readout functions, while still outperforming the standard ones. 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 [16], 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 convolutional operators or neighborhood aggregation schemes. The results are further supported by the training metrics (MAE, R^2) after 200 epochs (see also Appendix P, Tables 24 to 29), 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 different 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 tuples 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 (DENSE and GRU) indicates that it might be possible to relax this constraint for certain tasks, unless the application explicitly demands this type of invariance. A primary candidate for relaxation are molecular datasets, representing 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 S), such that it is improbable that the graph neural networks are learning simple associations based on position or presence of certain atoms in this setup. 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 hypothesize that similar reasoning can be applied to the tasks from other domains. Due to the general-purpose nature of the proposed methods, we do not anticipate any specific negative societal impacts.

We have also discussed possible domain specific interpretations for the effectiveness of models 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 pooling 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 performance improvements with the presented adaptive readouts (see Appendix R). We were, however, unable to observe a similar pattern for the considered classification tasks. Due 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 complex 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 [17] 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 has been focused on neighborhood aggregation schemes. This step also requires permutation invariance and it is interesting that a related work by Hamilton et al. [18] 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. [10] have introduced Janossy pooling, a permutation-invariant pooling technique applicable to the local neighborhoods considered during message passing, designed to tackle node classification tasks. Perhaps the most related to our direction and readouts is the concurrently developed work by Baek et al. [19] 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, matching or improving the existing methods. For the purpose of measuring and fixing the over-squashing problem in graph neural networks, Alon and Yahav [20] proposed a fully-adjacent layer (each pair of nodes is connected by an edge), which greatly improved the performance and resembles our use of the DENSE neural readout. Prior work has also considered tunable ℓ_p pooling [21] and more restrictive universal readouts based on deep sets [22]. However, neither of these approaches offers a comprehensive empirical evaluation at the scale provided here.

References

[1] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *CoRR*, abs/1609.02907, 2016.

[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? *CoRR*, abs/1810.00826, 2018.

[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] Juho Lee, Yoonho Lee, Jungtaek Kim, Adam R. Kosiorek, Seungjin Choi, and Yee Whye Teh. Set transformer. *CoRR*, abs/1810.00825, 2018.

[10] Ryan L. Murphy, Balasubramaniam Srinivasan, Vinayak A. Rao, and Bruno Ribeiro. Janossy pooling: Learning deep permutation-invariant functions for variable-size inputs. *CoRR*, abs/1811.01900, 2018.

[11] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017.

[12] KyungHyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder-decoder approaches. *CoRR*, abs/1409.1259, 2014.

[13] Shaked Brody, Uri Alon, and Eran Yahav. How attentive are graph attention networks? *CoRR*, abs/2105.14491, 2021.

[14] Gabriele Corso, Luca Cavalleri, Dominique Beaini, Pietro Liò, and Petar Velickovic. Principal neighbourhood aggregation for graph nets. *CoRR*, abs/2004.05718, 2020.

[15] Thomas N. Kipf and Max Welling. Variational graph auto-encoders, 2016.

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

[17] 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.

[18] 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.

[19] Jinheon Baek, Minki Kang, and Sung Ju Hwang. Accurate learning of graph representations with graph multiset pooling. *CoRR*, abs/2102.11533, 2021.

[20] Uri Alon and Eran Yahav. On the bottleneck of graph neural networks and its practical implications. *CoRR*, abs/2006.05205, 2020.

[21] 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.

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

[23] 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.

[24] 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.

[25] 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.

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 will be available on GitHub, but are currently included in the supplementary materials archive to maintain double-blind confidentiality.
   (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 Q.
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 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]