SIMPLE GNN NOISE REGULARISATION FOR 3D MOLECULAR PROPERTY PREDICTION AND BEYOND

Anonymous authors
Paper under double-blind review

ABSTRACT

Graph Neural Networks (GNNs) have been proven effective across a wide range of molecular property prediction and structured learning problems. However, their efficiency is known to be hindered by practical challenges such as oversmoothing. We introduce "Noisy Nodes", a very simple technique for improved training of GNNs, in which we corrupt the input graph with noise, and add a noise correcting node-level loss. Adding noise helps overfitting, and the noise correction loss helps ameliorate oversmoothing by encouraging diverse node latents. Our regulariser applies well-studied methods in simple, straightforward ways which allows even generic architectures not designed for quantum chemistry to achieve state of the art results. We also demonstrate the effectiveness of Noisy Nodes with non-spatial architectures on Open Graph Benchmark (OGB) datasets. Our results suggest Noisy Nodes can serve as a complementary building block in the GNN toolkit for 3D molecular property prediction and beyond.

1 INTRODUCTION

While Graph Neural Networks have demonstrated success in a wide variety of tasks (Zhou et al., 2020a; Wu et al., 2020; Bapst et al., 2020; Schütt et al., 2017; Klicpera et al., 2020a), it has been proposed that in practice "oversmoothing" (Chen et al., 2019) limits their ability to benefit from overparametrization. Here we propose a simple noise regulariser, Noisy Nodes, and demonstrate how it overcomes these challenges across a range of datasets and architectures, achieving top results on OC20 IS2RS & IS2RE direct, QM9 and OGBG-PCQM4Mv1.

Graph Neural Networks (GNNs) are a family of neural networks that operate on graph structured data by iteratively passing learned messages over the graph's structure (Scarselli et al., 2009; Bronstein et al., 2017; Gilmer et al., 2017; Battaglia et al., 2018; Shlomi et al., 2021). Oversmoothing is a phenomenon where a GNN's latent node representations become increasing indistinguishable over successive steps of message passing (Chen et al., 2019). Once these representations are oversmoothed, further message-passing cannot improve expressive capacity, and so performance does not improve. We speculate that GNN architectures applied on the benchmarks we study here suffer from these effects and could have improved accuracy if ameliorated.

Our "Noisy Nodes" method is an elegant, i.e. extremely simple, technique for regularising GNNs and associated training procedures. During training, our noise regularisation approach corrupts the input graph's attributes with noise, and adds a per-node noise correction term. We posit that our Noisy Nodes approach is effective because the model is rewarded for maintaining and refining distinct node representations through message passing to the final output, which causes it to resist oversmoothing. Like denoising autoencoders, it may also encourage the model to explicitly learn the manifold on which the uncorrupted input graph's features lie, analogous to a form of representation learning. When applied to 3D molecular prediction tasks, it encourages the model to distinguish between low and high energy states. We find that applying Noisy Nodes reduces oversmoothing for shallower networks, and allows us to see improvements with added depth for some architectures, even on tasks for which depth was assumed to be unhelpful.

This study's approach is to investigate the combination of Noisy Nodes with generic, popular baseline GNN architectures. For 3D Molecular prediction we use a standard architecture working on 3D points developed for particle fluid simulations, the Graph Net Simulator (GNS) (Sanchez-Gonzalez* et al., 2020), which has also been used for molecular property prediction (Hu et al., 2021b). Without

using Noisy Nodes the GNS is not a competitive model, but using Noisy Nodes allows the GNS to achieve top performance on three 3D molecular property prediction tasks: the OC20 IS2RE direct task by 43% over previous work, 12% on OC20 IS2RS direct, and top results on 3 out of 12 of the QM9 tasks. For non-spatial GNN benchmarks we test a Message Passing Neural Network (MPNN) with Virtual Node (Gilmer et al., 2017) on OGBG-MOLPCBA and OGBG-PCQM4M (Hu et al., 2021a) and again see significant improvements. Finally, we applied Noisy Nodes to a GCN (Kipf & Welling, 2016), arguably the most popular and simple GNN, trained on OGBN-Arxiv and see similar results. These results suggest Noisy Nodes can serve as a complementary GNN building block for 3D molecular property prediction and beyond.

2 Related Work

Oversmoothing. Recent work has aimed to understand why it is challenging to realise the benefits of training deeper GNNs (Wu et al., 2021). A key contribution has been the analysis of "oversmoothing" which describes how all node features become almost identical in GCNs after a few layers. Since first being noted in (Li et al., 2018) oversmoothing has been studied extensively and regularisation techniques have been suggested to overcome it (Chen et al., 2019; Cai & Wang, 2020; Rong et al., 2019; Zhou et al., 2020b; Yang et al., 2020). A recent paper, (Li et al., 2021), finds, as in previous work, (Li et al., 2019; 2020), the optimal depth for some datasets they evaluate on to be far lower (5 for OGBN-Arxiv from the Open Graph Benchmark (Hu et al., 2020a), for example) than the 1000 layers possible.

Denoising Models. Training neural networks with noise has a long history (Sietsma & Dow, 1991; Bishop, 1995). Of particular relevance are Denoising Autoencoders (Vincent et al., 2008) in which an autoencoder is trained to map corrupted inputs $\tilde{\mathbf{x}}$ to uncorrupted inputs $\mathbf{x}$. Denoising Autoencoders have found particular success as a form of pre-training for representation learning (Vincent et al., 2010). More recently, in research applying GNNs to simulation (Sanchez-Gonzalez et al., 2018; Sanchez-Gonzalez* et al., 2020; Pfaff et al., 2020) Gaussian noise is added during training to input positions of a ground truth simulator to mimic the distribution of errors of the learned simulator. Pre-training methods (Devlin et al., 2019; You et al., 2020; Thakoor et al., 2021) are another similar approach; most similarly to our method Hu et al. (2020b) apply a reconstruction loss to graphs with masked nodes to generate graph embeddings for use in downstream tasks. FLAG (Kong et al., 2020) adds adversarial noise during to input node features as a form of data augmentation for GNNs that demonstrates improved performance for many tasks. It does not add an additional auxiliary loss, which we find is essential for addressing oversmoothing.

Machine Learning for 3D Molecular Property Prediction. One application of GNNs is to speed up quantum chemistry calculations which operate on 3D positions of a molecule (Duvenaud et al., 2015; Gilmer et al., 2017; Schütt et al., 2017; Hu et al., 2021b). In these graphs atoms are nodes and edges are constructed between close 3D neighbours. Common goals are the prediction of molecular properties (Ramakrishnan et al., 2014), forces (Chmiela et al., 2017), energies (Chanussot* et al., 2020) and charges (Unke & Meuwly, 2019). These datasets have spurred the development of GNNs that embed 3D physical symmetry inductive biases such as the rotation equivariance of forces. Such inductive biases typically improve performance and sample complexity.

A common approach to embed physical symmetries is to design a network that predicts a rotation and translation invariant energy (Schütt et al., 2017; Klicpera et al., 2020a; Liu et al., 2021). The input features of such models include distances (Schütt et al., 2017), angles (Klicpera et al., 2020b;a) or torsions and higher order terms (Liu et al., 2021). An alternative approach to embedding symmetries is to design a rotation equivariant neural network that use equivariant representations (Thomas et al., 2018; Köhler et al., 2019; Kondor et al., 2018; Fuchs et al., 2020; Batzner et al., 2021; Anderson et al., 2019; Satorras et al., 2021).

Machine Learning for Bond and Atom Molecular Graphs. Predicting properties from molecular graphs without 3D points, such as graphs of bonds and atoms, is studied separately and often used to benchmark generic graph property prediction models such as GCNs (Hu et al., 2020a) or GATs (Veličković et al., 2018). Models developed for 3D molecular property prediction cannot be applied to bond and atom graphs and vice versa. Common datasets that contain such data are OGBG-MOLPCBA and OGBG-MOLHIV.

3 PRELIMINARIES: GRAPH PREDICTION PROBLEM

Let $G = (V, E, g)$ be an input graph. The nodes are $V = \{v_1, \ldots, v_{|V|}\}$, where $v_i \in \mathbb{R}^{d_v}$. The directed, attributed edges are $E = \{e_1, \ldots, e_{|E|}\}$: each edge includes a sender node index, receiver node index, and edge attribute, $e_k = (s_k, r_k, e_k)$, respectively, where $s_k, r_k \in \{1, \ldots, |V|\}$ and $e_k \in \mathbb{R}^{d_e}$. The graph-level property is $g \in \mathbb{R}^{d_g}$.

The goal is to predict a target graph, G', with the same structure as G, but different node, edge, and/or graph-level attributes. We denote $\hat{G}'$ as a model's prediction of G'. Some error metric defines quality of $\hat{G}'$ with respect to the target G', Error$(\hat{G}', G')$, which the training loss terms are defined to optimize. In this paper the phrase "message passing steps" is synonymous with "GNN layers".

4 NOISY NODES

One way to incentivise diverse node representations is to have diverse node-level targets; in order to do well on such a task, the GNN must have diverse latents at the penultimate layer of the network. However, many problems are not node level prediction tasks, or do not contain sufficiently diverse node targets.

Noisy Nodes tackles this problem by adding a diverse noise correction auxiliary target. It modifies the original graph prediction problem definition in several ways. It introduces a graph corrupted by noise, $\tilde{G} = (\tilde{V}, \tilde{E}, \tilde{g})$, where $\tilde{v}_i \in \tilde{V}$ is constructed by adding noise, σ_i, to the input nodes, $\tilde{v}_i = v_i + \sigma_i$. The edges, $\tilde{E}$, and graph-level attribute, $\tilde{g}$, can either be uncorrupted by noise (i.e., $\tilde{E} = E$, $\tilde{g} = g$), calculated from the noisy nodes (for example in a nearest neighbors graph), or corrupted independent of the nodes—these are minor choices that can be informed by the specific problem setting.

Our method requires a noise correction target to prevent oversmoothing by enforcing diversity in the last layers of the GNN. For problems where the Error is defined with respect to graph-level predictions (e.g., predict the minimum energy value of some molecular system), a second output head can be added to the GNN architecture which requires denoising the inputs as targets, analogous to an auxiliary denoising autoencoder head and training objective. Because the $v'_i \in V'$ is not specified by the goal, we can set $v'_i = v_i$ and train the model to predict the uncorrupted input nodes.

The added noise prevents overfitting, and the denoising loss ensures a diverse node level target. Such a recipe also encourages the model to take advantage of message passing. For example, consider a group of three atoms corrupted by noise, leading to two atoms being very close together (and so with a very high interatomic force). To denoise the interatomic distances correctly the GNN must use message passing to triangulate between the three atoms.

In Figure 2 we illustrate the impact of Noisy Nodes on oversmoothing by plotting the Mean Absolute Distance (MAD) (Chen et al., 2020) of the residual updates of each layer of a GNN trained on the QM9 (Ramakrishnan et al., 2014) dataset. MAD is a measure of the diversity of graph node features, often used to quantify oversmoothing, the higher the number the more diverse the node features, the lower the number the less diverse. In this plot we can see that for Noisy Nodes the node updates remain diverse for all of the layers, whereas without Noisy Nodes diversity is lost after 3 layers.

The Graph Manifold Learning Perspective. By using an implicit mapping from corrupted data to clean data, the Noisy Nodes objective encourages the model to learn the manifold on which the clean data lies—the GNN learns to go from low probability graphs to high probability graphs. In the autoencoder case the GNN learns the manifold of the input data. When node targets are provided, the GNN learns the manifold of the target data (e.g. the manifold of atoms at equilibrium). Such a manifold may include commonly repeated substructures that are useful for downstream prediction tasks.

The Energy Perspective for Molecular Property Prediction. Local, random distortions of the geometry of a molecule at a local energy minimum are almost certainly higher energy configurations. As such, a task that maps from a noised molecule to a local energy minimum is learning a mapping from high energy to low energy. Data such as QM9 contains molecules at local minima.

Some problems have input data that is already high energy, and targets that are at equilibrium. For these datasets we can generate new high energy states by adding noise to the inputs but keeping the

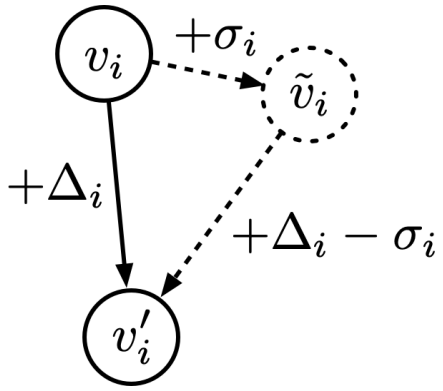

Figure 1: 3D Noisy Node mechanics during training. Input positions are corrupted with noise σ, and the training objective is to node-level difference between target positions and the noisy inputs. When the inputs are the targets ($\Delta = 0$), this is equivalent to a denoising auto-encoder.

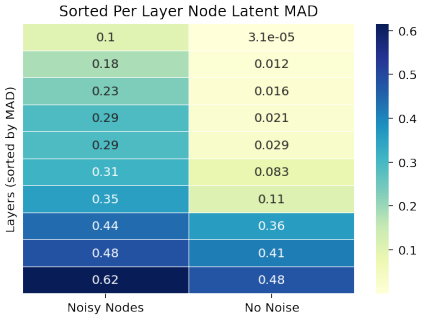

Figure 2: Per layer node latent diversity, measured by MAD on a 10 layer GNS. We can see that Noisy Nodes encourages the model to make diverse residual updates per node - ameliorating oversmoothing. For Noisy Nodes layer MAD remains much higher as depth increases.

equilibrium target the same, Figure 1 demonstrates this approach. To preserve translation invariance we use displacements between input and target Δ, the corrected target after noise is $\Delta - \sigma$.

5 3D Molecular Property Prediction Experiments and Results

In this section we evaluate how a popular, simple model, the GNS (Sanchez-Gonzalez* et al., 2020) performs on 3D molecular prediction tasks when combined with Noisy Nodes. The GNS was originally developed for particle fluid simulations, but has recently been adapted for molecular property prediction (Hu et al., 2021b). We find that Without Noisy Nodes the GNS architecture is not competitive, but by using Noisy Nodes we see improved performance comparable to the use of specialised architectures.

We made minor changes to the GNS architecture; we featurize the distance input features using radial basis functions, use the concept of "grouping" from Li et al. (2021), and finally we find that adding a loss after each group aids training stability.

We tested this architecture on three challenging molecular property prediction benchmarks: OC20 (Chanussot* et al., 2020) IS2RS & IS2RE, and QM9 (Ramakrishnan et al., 2014). These benchmarks are detailed below, but as general distinctions, OC20 tasks use graphs 2-20x larger than QM9. While QM9 always requires graph-level prediction, one of OC20's two tasks (IS2RS) requires node-level predictions while the other (IS2RE) requires graph-level predictions.

5.1 Training

We minimise the mean squared error loss on mean and standard deviation normalised targets and use the Adam (Kingma & Ba, 2015) optimiser with warmup and cosine decay. For OC20 IS2RE energy prediction we subtract a learned reference energy, computed using an MLP with atom types as input.

For the GNS model the node and edge latents as well as MLP hidden layers were sized 512, with 3 layers per MLP and using shifted softplus activations throughout. OC20 & QM9 Models were trained on 8 TPU devices and evaluated on a single V100 GPUs. We provide the full set of hyper-parameters and computational resources used separately for each dataset in the Appendix. All noise levels were determined by sweeping a small range of of values (≈ 10) informed by the noised feature covariance.

Our code base is implemented in JAX using Haiku and Jraph for GNNs, and Optax for training (Bradbury et al., 2018; Babuschkin et al., 2020; Godwin* et al., 2020; Hennigan et al., 2020). Model selection used early stopping.

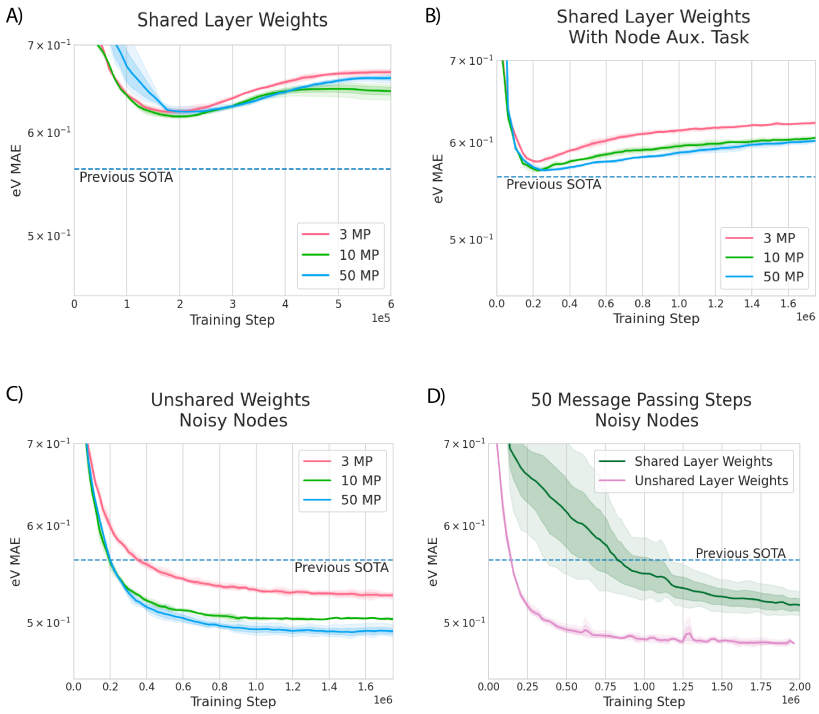

Figure 3: Validation curves, OC20 IS2RE ID. **A)** Without any node targets our model has poor performance and realises no benefit from depth. **B)** After adding a position node loss, performance improves as depth increases. **C)** As we add Noisy Nodes and parameters the model achieves SOTA, even with 3 layers, and stops overfitting. **D)** Adding Noisy Nodes allows a model with even fully shared weights to achieve SOTA.

5.2 Open Catalyst 2020

Dataset. The OC20 dataset (Chanussot* et al., 2020) (CC Attribution 4.0) describes the interaction of a small molecule (the adsorbate) and a large slab (the catalyst), with total systems consisting of 20-200 atoms simulated until equilibrium is reached.

We focus on two tasks; the Initial Structure to Resulting Energy (IS2RE) task which takes the initial structure of the simulation and predicts the final energy, and the Initial Structure to Resulting Structure (IS2RS) which takes the initial structure and predicts the relaxed structure. Note that we train the more common "direct" prediction task that map directly from initial positions to target in a single forward pass, and compare against other models trained for direct prediction. We do not train "relaxation" models, which have reported fewer results and use far more data.

Models are evaluated on 4 held out test sets; a set of In Distribution (ID) catalysts and adsorbates, Out of Distribution (OOD) catalysts and ID adsorbates, ID catalysts and OOD adsorbates and finally a set of both OOD catalysts and adsorbates. Four canonical validation datasets are also provided. Test sets are evaluated on a remote server hosted by the dataset authors with a very limited number of submissions per team.

During training we first sample uniformly from a point in the relaxation trajectory or interpolation, and then add I.I.D Gaussian noise with mean zero and $\sigma = 0.3$. The Noisy Node target is the relaxed structure.

We first convert to fractional coordinates (i.e. use the periodic unit cell as the basis) which render the predictions of our model invariant to rotations, and append the following rotation and translation invariant vector $(\alpha\beta^T, \beta\gamma^T, \alpha\gamma^T, |\alpha|, |\beta|, |\gamma|) \in \mathbb{R}^6$ to the edge features where α, β, γ are vectors of the unit cell. This additional vector provides rotation invariant angular and extent information to the GNN.

Table 1: OC20 ISRE Validation, eV MAE, ↓.
"GNS-Shared" indicates shared weights. "GNS-10" indicates a group size of 10.

Model	Layers	OOD Both	OOD Adsorbate	OOD Catalyst	ID
GNS	50	0.59 ±0.01	0.65 ±0.01	0.55 ±0.00	0.54 ±0.00
GNS-Shared + Noisy Nodes	50	0.49 ±0.00	0.54 ±0.00	0.51 ±0.01	0.51 ±0.01
GNS + Noisy Nodes	50	0.48 ±0.00	0.53 ±0.00	0.49 ±0.01	0.48 ±0.00
GNS-10 + Noisy Nodes	100	**0.46**±0.00	**0.51** ±0.00	**0.48** ±0.00	**0.47** ±0.00

Table 2: Results OC20 IS2RE Test

eV MAE ↓					
	SchNet	DimeNet++	SpinConv	SphereNet	GNS + Noisy Nodes
OOD Both	0.704	0.661	0.674	0.638	**0.485 (-24.0%)**
OOD Adsorbate	0.734	0.725	0.723	0.703	**0.543 (-22.8%)**
OOD Catalyst	0.662	0.576	0.569	0.571	**0.473 (-17.2%)**
ID	0.639	0.562	0.558	0.563	**0.457 (-18.8%)**
Average Energy within Threshold (ADwT) ↑					
	SchNet	DimeNet++	SpinConv	SphereNet	GNS + Noisy Nodes
OOD Both	0.0221	0.0241	0.0233	0.0241	**0.0392 (+61.8%)**
OOD Adsorbate	0.0233	0.0207	0.026	0.0229	**0.0434 (+89.5%)**
OOD Catalyst	0.0294	0.0410	0.0382	0.0409	**0.0565 (+38.1%)**
ID	0.0296	0.0425	0.0408	0.0447	**0.0648 (+45.0%)**

IS2RE Results. In Figure 3 we show how using Noisy Nodes allows the GNS to achieve state of the art performance. Figure 3 A shows that without any auxiliary node target, an IS2RE GNS achieves poor performance even with increased depth. As we add a node level position in B) target we see better performance, and improvement as depth increases, validating our hypothesis that node level targets are key to addressing oversmoothing. In C) we add noisy nodes and parameters, and see that the increased diversity of the node level predictions leads to very significant improvements and SOTA, even for a shallow 3 layer network. D) demonstrates this effect is not just due to increased parameters - SOTA can still be achieve with shared layer weights .

In Table 1 we conduct an ablation on our hyperparameters, and again demonstrate the improved performance of using Noisy Nodes. Results were averaged over 3 seeds and standard errors on the best obtained checkpoint show little sensitivity to initialisation. We conducted an ablation on ID comparing sampling from a relaxation trajectory and interpolating between initial & final positions, which found that interpolation improved our score from 0.48 to 0.45.

Our best hyperparameter setting was 100 layers which achieved a 43.3% relative performance improvement against SOTA results (Table 2). Due to limited permitted test submissions, results presented here were from one test upload of our best performing validation seed.

IS2RS Results. In Table 3 we perform the same ablation studies on the IS2RS validation set using the Average Distance within Threshold (ADwT) metric and observe the same relative effects. In Table 4 we see that GNS + Noisy Nodes is significantly better than the only other reported IS2RS direct result, ForceNet, itself a GNS variant. Similarly, to IS2RE, we tried an ablation using random interpolated positions between input and target instead of sampling from mid trajectory, and noted an improvement from 54.3% to 54.5% on OOD both validation.

We note a drop between the validation and test sets, which we speculate is due to distribution shift. We also notice a consistent pattern that having energy predictions as an auxiliary loss inhibits both relaxation and direct IS2RE predictions, we speculate that adding more capacity to the positional decoder head may help ameliorate such effects.

Table 3: OC20 IS2RS Validation, ADwT, ↑

Model	Layers	Group Size	OOD Both	OOD Adsorbate	OOD Catalyst	ID
GNS	50	Unshared	43.0%±0.0	38.0%±0.0	37.5% 0.0	40.0%±0.0
+ Noisy Nodes	50	Shared	49.2%±0.0	42.6%±0.0	42.5%±0.0	43.6% ±0.01
+ Noisy Nodes	50	Unshared	50.1%±0.0	44.3%±0.0	44.1%±0.0	46.1% ±0.0
+ Noisy Nodes	50	10	52.0%±0.0	46.2%±0.0	46.1% ±0.0	48.3% ±0.0
++ Pos only	100	10	**54.3**%±0.0	**48.3**%±0.0	**48.2**% ±0.0	**50.0**% ±0.0

Table 4: OC20 IS2RS Test, ADwT, ↑

Model	OOD Both	OOD Adsorbate	OOD Catalyst	ID
ForceNet	46.9%	37.7%	43.7%	44.9%
GNS + Noisy Nodes	**52.7%**	**43.9%**	**48.4%**	**50.9%**
Relative Improvement	**+12.4%**	**+16.4%**	**+10.7%**	**+13.3%**

5.3 QM9

Dataset. The QM9 benchmark (Ramakrishnan et al., 2014) contains 134k molecules in equilibrium with up to 9 heavy C, O, N and F atoms, targeting 12 associated chemical properties (License: CCBY 4.0). We use 114k molecules for training, 10k for validation and 10k for test. All results are on the test set. We subtract a fixed per atom energy from the target values computed from linear regression to reduce variance. We perform training in eV units for energetic targets, and evaluate using MAE. We summarise the results across the targets using mean standardised MAE (std. MAE) in which MAEs are normalised by their standard deviation, and mean standardised logMAE. Std. MAE is dominated by targets with high relative error such as $\Delta\epsilon$, whereas logMAE is sensitive to outliers such as $\langle R^2 \rangle$. As is standard for this dataset, a model is trained separately for each target.

For this dataset we add I.I.D Gaussian noise with mean zero and $\sigma = 0.02$ to the input atom positions. A denoising autoencoder loss is used.

Results In Table 6 we can see that adding Noisy Nodes significantly improves results by 23.1% relative for GNS, making it competitive with specialised architectures. To understand the effect of adding a denoising loss, we tried just adding noise and found no where near the same improvement (Table 6).

A GNS + Noisy Nodes with 30 layers achieves top results on 3 of the 12 targets and comparable performance on the remainder (Table 6). On the std. MAE aggregate metric GNS + Noisy Nodes performs better than all other reported results, showing that Noisy Nodes can make even a generic model competitive with models hand-crafted for molecular property prediction. The same trend is repeated for an rotation invariant version of this network that uses the principle axes of inertia ordered by eigenvalue as the co-ordinate frame (Table 5).

$\langle R^2 \rangle$, the electronic spatial extent, is an outlier for GNS + Noisy Nodes. Interestingly, we found that without noise GNS + Noisy Nodes achieves 0.33 for this target. We speculate that this target is

Table 5: QM9, Impact of Noisy Nodes on GNS architecture.

	Layers	std. MAE	% Change	logMAE
GNS	10	1.17	-	-5.39
GNS + Noise But No Node Target	10	1.16	-0.9%	-5.32
GNS + Noisy Nodes	10	0.90	-23.1%	-5.58
GNS-10 + Noisy Nodes	20	0.89	-23.9%	-5.59
GNS-10 + Noisy Nodes + Invariance	30	0.92	-21.4%	-5.57
GNS-10 + Noisy Nodes	30	**0.88**	**-24.8%**	**-5.60**

Table 6: QM9, Test MAE, Mean & Standard Deviation of 3 Seeds Reported.

Target	Unit	SchNet	E(n)GNN	DimeNet++	SphereNet	PaiNN	**GNS + Noisy Nodes**
μ	D	0.033	0.029	0.030	0.027	**0.012**	0.025 $\pm$0.01
α	${a_0}^3$	0.235	0.071	**0.043**	0.047	0.045	0.052 $\pm$0.00
ϵ_{HOMO}	meV	41	29.0	24.6	23.6	27.6	**20.4** $\pm$0.2
ϵ_{LUMO}	meV	34	25.0	19.5	18.9	20.4	**18.6** $\pm$0.4
$\Delta\epsilon$	meV	63	48.0	32.6	32.3	45.7	**28.6** $\pm$0.1
$\langle R^2 \rangle$	${a_0}^2$	**0.07**	0.11	0.33	0.29	0.07	0.70 $\pm$0.01
ZPVE	meV	1.7	1.55	1.21	**1.12**	1.28	1.16 $\pm$0.01
U_0	meV	14.00	11.00	6.32	6.26	**5.85**	7.30 $\pm$0.12
U	meV	19.00	12.00	6.28	7.33	**5.83**	7.57 $\pm$0.03
H	meV	14.00	12.00	6.53	6.40	**5.98**	7.43$\pm$0.06
G	meV	14.00	12.00	7.56	8.0	**7.35**	8.30 $\pm$0.14
c_{v}	$\frac{\text{cal}}{\text{mol K}}$	0.033	0.031	0.023	**0.022**	0.024	0.025 $\pm$0.00
std. MAE	%	1.76	1.22	0.98	0.94	1.00	**0.88**
logMAE		-5.17	-5.43	-5.67	-5.68	**-5.85**	-5.60

Table 7: OGBG-PCQM4M Results

Model	Number of Layers	Using Noisy Nodes	MAE
MPNN + Virtual Node	16	Yes	0.1249 $\pm$ 0.0003
MPNN + Virtual Node	50	No	0.1236 $\pm$ 0.0001
Graphormer (Ying et al., 2021)	-	-	0.1234
MPNN + Virtual Node	50	Yes	**0.1218** $\pm$ **0.0001**

particularly sensitive to noise, and the best noise value for this target would be significantly lower than for the dataset as a whole.

6 Non-Spatial Tasks

The previous experiments use the 3D geometries of atoms, and models that operate on 3D points. However, the recipe adding a denoising autoencoder auxiliary loss can be applied to other graphs with different types of features. In this section we apply Noisy Nodes to additional datasets with no 3D points, using different GNNs, and show analagous effects to the 3D case. All results reported as an average of 10 random seeds. OGBG-PCQM4M & OGBG-MOLPCBA were trained with 16 TPUs and evaluated with a single V100 GPU. OGBN-Arxiv was trained and evalated with a single TPU.

6.1 OGBG-PCQM4M

This dataset from the OGB benchmarks consists of molecular graphs which consist of bonds and atom types, and no 3D or 2D coordinates. To adapt Noisy Nodes to this setting, we randomly flip node and edge features at a rate of 5% and add a reconstruction loss. We evaluate Noisy Nodes using MPNN with Virtual Node as a base model. The test set is not currently available for this dataset. Version 2 of this dataset, with different splits, was released shortly before submission and we plan to update our results with the final test figures when we have the opportunity.

In Table 7 we see that for this task Noisy Nodes enables a 50 layer MPNN to reach state of the art results. Before adding Noisy Nodes, adding capacity beyond 16 layers did not improve results.

6.2 OGBG-MOLPCBA

The OGBG-MOLPCBA dataset contains molecular graphs with no 3D points, with the goal of predicting 128 biological activities. On the OGBG-MOLPCBA dataset we again use an MPNN with Virtual Node and random flipping noise. In Figure 4 we see that adding Noisy Nodes improves the

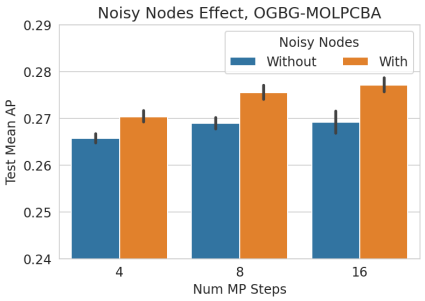

Figure 4: Adding Noisy Nodes with random flipping of input categories improves the performance of MPNNs, and we observe that performance improves with depth.

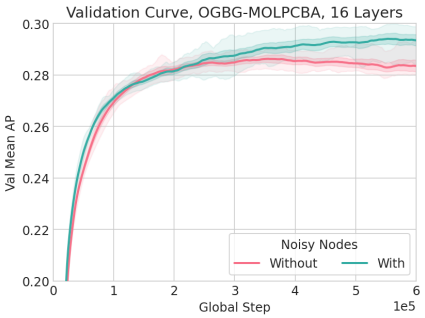

Figure 5: Validation curve comparing with and without noisy nodes. Using Noisy Nodes leads to a consistent improvement.

performance of the base model, and that the benefits are most visible on deep networks. Our 16 layer network improved from 26.9% $\pm$ 0.002 to 27.7% $\pm$ 0.002. Figure 5 demonstrates how Noisy Nodes improves performance during training. Of the reported leaderboard results, MPNNs are most similar to GCNs[1] + Virtual Node and GIN + Virtual Node (Xu et al., 2018) which report which report results of 24.2% $\pm$ 0.003 and 27.03% $\pm$ 0.003 respectively.

6.3 OGBN-ARXIV

The above results use models with explicit edge updates, and are reported for graph prediction. To test the effectiveness with Noisy Nodes with GCNs, arguably the simplest and most popular GNN, we use OGBN-ARXIV, a citation network with the goal of predicting the arxiv category of each paper. Adding Noisy Nodes, with noise as input dropout of 0.1, to 4 layer GCN with residual connections improves from 72.39% $\pm$ 0.002 accuracy to 72.52% $\pm$ 0.003 accuracy. A baseline 4 layer GCN on this dataset reports 71.71% $\pm$ 0.002. The SOTA for this dataset is 74.31% (Sun & Wu, 2020).

6.4 LIMITATIONS

We have not demonstrated the effectiveness of Noisy Nodes in small data regimes, which may be important for learning from experimental data. The representation learning perspective requires access to a local minimum configuration, which is not the case for all quantum modeling datasets. We have also not demonstrated the combination of Noisy Nodes with more sophisticated 3D molecular property prediction models such as DimeNet++(Klicpera et al., 2020a). We leave this to future work.

Noisy Nodes requires careful selection of the form of noise, and a balance between the auxiliary and primary losses. This can require hyper parameter tuning, and models can be sensitive to the choice of these parameters. Noisy Nodes has a particular effect for deep GNNs, but depth is not always an advantage. There are situations, for example molecular dynamics, which place a premium on very fast inference time. However even at 3 layers (a comparable depth to alternative architectures) the GNS architecture achieves state of the art validation OC20 IS2RE predictions (Figure 3).

7 CONCLUSIONS

In this work we present Noisy Nodes, a novel regularisation technique for GNNs with particular focus on 3D molecular property prediction. Noisy nodes helps address common challenges around oversmoothed node representations, shows benefits for GNNs of all depths, but in particular improves performance for deeper GNNs. We demonstrate results on challenging 3D molecular property prediction tasks, and some generic GNN benchmark datasets. We believe these results demonstrate Noisy Nodes could be a useful building block for GNNs for molecular property prediction and beyond.

[1]The GCN implemented in the official OGB code base has explicit edge updates, akin to the MPNN. The only difference between an MPNN and GCN in this case is the adjacency matrix normalization.

8 REPRODUCIBILITY STATEMENT

Code for reproducing OGB-PCQM4M results using Noisy Nodes is available on github, and was prepared as part of a leaderboard submission. We withhold the link here to preserve the double blind review process but will provide the link for camera ready.

We provide detailed hyper parameter settings for all our experiments in the appendix, in addition to formulae for computing the encoder and decoder stages of the GNS.

9 ETHICS STATEMENT

Who may benefit from this work? Molecular property prediction with GNNs is a fast-growing area with applications across domains such as drug design, catalyst discovery, synthetic biology, and chemical engineering. Noisy Nodes could aid models applied to these domains. We also demonstrate on OC20 that our direct state prediction approach is nearly as accurate as learned relaxed approaches at a small fraction of the computational cost, which may support material design which requires many predictions.

Finally, Noisy Nodes could be adapted and applied to many areas in which GNNs are used—for example, knowledge base completion, physical simulation or traffic prediction.

Potential negative impact and reflection. Noisy Nodes sees improved performance from depth, but the training of very deep GNNs could contribute to global warming. Care should be taken when utilising depth, and we note that Noisy Nodes settings can be calibrated at shallow depth.

REFERENCES

Brandon M. Anderson, T. Hy, and R. Kondor. Cormorant: Covariant molecular neural networks. In *NeurIPS*, 2019.

Igor Babuschkin, Kate Baumli, Alison Bell, Surya Bhupatiraju, Jake Bruce, Peter Buchlovsky, David Budden, Trevor Cai, Aidan Clark, Ivo Danihelka, Claudio Fantacci, Jonathan Godwin, Chris Jones, Tom Hennigan, Matteo Hessel, Steven Kapturowski, Thomas Keck, Iurii Kemaev, Michael King, Lena Martens, Vladimir Mikulik, Tamara Norman, John Quan, George Papamakarios, Roman Ring, Francisco Ruiz, Alvaro Sanchez, Rosalia Schneider, Eren Sezener, Stephen Spencer, Srivatsan Srinivasan, Wojciech Stokowiec, and Fabio Viola. The DeepMind JAX Ecosystem, 2020. URL `http://github.com/deepmind`.

V. Bapst, T. Keck, Agnieszka Grabska-Barwinska, C. Donner, E. D. Cubuk, S. Schoenholz, A. Obika, Alexander W. R. Nelson, T. Back, D. Hassabis, and P. Kohli. Unveiling the predictive power of static structure in glassy systems. *Nature Physics*, 16:448–454, 2020.

P. Battaglia, Jessica B. Hamrick, V. Bapst, A. Sanchez-Gonzalez, V. Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, A. Santoro, R. Faulkner, Çaglar Gülçehre, H. Song, A. J. Ballard, J. Gilmer, George E. Dahl, Ashish Vaswani, Kelsey R. Allen, Charlie Nash, Victoria Langston, Chris Dyer, N. Heess, Daan Wierstra, P. Kohli, M. Botvinick, Oriol Vinyals, Y. Li, and Razvan Pascanu. Relational inductive biases, deep learning, and graph networks. *ArXiv*, abs/1806.01261, 2018.

Simon Batzner, T. Smidt, L. Sun, J. Mailoa, M. Kornbluth, N. Molinari, and B. Kozinsky. Se(3)-equivariant graph neural networks for data-efficient and accurate interatomic potentials. *ArXiv*, abs/2101.03164, 2021.

Charles M. Bishop. Training with noise is equivalent to tikhonov regularization. *Neural Computation*, 7:108–116, 1995.

James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018. URL `http://github.com/google/jax`.

Michael M Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. Geometric deep learning: going beyond euclidean data. *IEEE Signal Processing Magazine*, 34(4):18–42, 2017.

Chen Cai and Yusu Wang. A note on over-smoothing for graph neural networks. *CoRR*, abs/2006.13318, 2020. URL https://arxiv.org/abs/2006.13318.

Lowik Chanussot*, Abhishek Das*, Siddharth Goyal*, Thibaut Lavril*, Muhammed Shuaibi*, Morgane Riviere, Kevin Tran, Javier Heras-Domingo, Caleb Ho, Weihua Hu, Aini Palizhati, Anuroop Sriram, Brandon Wood, Junwoong Yoon, Devi Parikh, C. Lawrence Zitnick, and Zachary Ulissi. Open catalyst 2020 (oc20) dataset and community challenges. *ACS Catalysis*, 0(0): 6059–6072, 2020. doi: 10.1021/acscatal.0c04525. URL https://doi.org/10.1021/acscatal.0c04525.

Deli Chen, Yankai Lin, Wei Li, Peng Li, Jie Zhou, and Xu Sun. Measuring and relieving the over-smoothing problem for graph neural networks from the topological view. *CoRR*, abs/1909.03211, 2019. URL http://arxiv.org/abs/1909.03211.

Deli Chen, Yankai Lin, W. Li, Peng Li, J. Zhou, and Xu Sun. Measuring and relieving the over-smoothing problem for graph neural networks from the topological view. In *AAAI*, 2020.

Stefan Chmiela, A. Tkatchenko, H. E. Sauceda, I. Poltavsky, Kristof T. Schütt, and K. Müller. Machine learning of accurate energy-conserving molecular force fields. *Science Advances*, 3, 2017.

J. Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*, 2019.

David Duvenaud, Dougal Maclaurin, Jorge Aguilera-Iparraguirre, Rafael Gómez-Bombarelli, Timothy Hirzel, Alán Aspuru-Guzik, and Ryan P. Adams. Convolutional networks on graphs for learning molecular fingerprints. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'15, pp. 2224–2232, Cambridge, MA, USA, 2015. MIT Press.

F. Fuchs, Daniel E. Worrall, Volker Fischer, and M. Welling. Se(3)-transformers: 3d roto-translation equivariant attention networks. *ArXiv*, abs/2006.10503, 2020.

J. Gilmer, S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural message passing for quantum chemistry. *ArXiv*, abs/1704.01212, 2017.

Jonathan Godwin*, Thomas Keck*, Peter Battaglia, Victor Bapst, Thomas Kipf, Yujia Li, Kimberly Stachenfeld, Petar Veličković, and Alvaro Sanchez-Gonzalez. Jraph: A library for graph neural networks in jax., 2020. URL http://github.com/deepmind/jraph.

Tom Hennigan, Trevor Cai, Tamara Norman, and Igor Babuschkin. Haiku: Sonnet for JAX, 2020. URL http://github.com/deepmind/dm-haiku.

Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. Open graph benchmark: Datasets for machine learning on graphs. *ArXiv*, abs/2005.00687, 2020a.

Weihua Hu, Bowen Liu, Joseph Gomes, M. Zitnik, Percy Liang, V. Pande, and J. Leskovec. Strategies for pre-training graph neural networks. *arXiv: Learning*, 2020b.

Weihua Hu, Matthias Fey, Hongyu Ren, Maho Nakata, Yuxiao Dong, and Jure Leskovec. Ogb-lsc: A large-scale challenge for machine learning on graphs. *arXiv preprint arXiv:2103.09430*, 2021a.

Weihua Hu, Muhammed Shuaibi, Abhishek Das, Siddharth Goyal, Anuroop Sriram, J. Leskovec, Devi Parikh, and C. L. Zitnick. Forcenet: A graph neural network for large-scale quantum calculations. *ArXiv*, abs/2103.01436, 2021b.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2015.

Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *CoRR*, abs/1609.02907, 2016. URL http://arxiv.org/abs/1609.02907.

Johannes Klicpera, Shankari Giri, Johannes T. Margraf, and Stephan Günnemann. Fast and uncertainty-aware directional message passing for non-equilibrium molecules. *CoRR*, abs/2011.14115, 2020a. URL https://arxiv.org/abs/2011.14115.

Johannes Klicpera, Janek Groß, and Stephan Günnemann. Directional message passing for molecular graphs. *ArXiv*, abs/2003.03123, 2020b.

Risi Kondor, Hy Truong Son, Horace Pan, Brandon M. Anderson, and Shubhendu Trivedi. Covariant compositional networks for learning graphs. *CoRR*, abs/1801.02144, 2018. URL http://arxiv.org/abs/1801.02144.

Kezhi Kong, Guohao Li, Mucong Ding, Zuxuan Wu, Chen Zhu, Bernard Ghanem, G. Taylor, and T. Goldstein. Flag: Adversarial data augmentation for graph neural networks. *ArXiv*, abs/2010.09891, 2020.

Jonas Köhler, Leon Klein, and Frank Noé. Equivariant flows: sampling configurations for multi-body systems with symmetric energies, 2019.

G. Li, M. Müller, Ali K. Thabet, and Bernard Ghanem. Deepgcns: Can gcns go as deep as cnns? *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 9266–9275, 2019.

Guohao Li, C. Xiong, Ali K. Thabet, and Bernard Ghanem. Deepergcn: All you need to train deeper gcns. *ArXiv*, abs/2006.07739, 2020.

Guohao Li, Matthias Müller, Bernard Ghanem, and Vladlen Koltun. Training graph neural networks with 1000 layers. *CoRR*, abs/2106.07476, 2021. URL https://arxiv.org/abs/2106.07476.

Qimai Li, Zhichao Han, and Xiao-Ming Wu. Deeper insights into graph convolutional networks for semi-supervised learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

Yi Liu, Limei Wang, Meng Liu, Xuan Zhang, Bora Oztekin, and Shuiwang Ji. Spherical message passing for 3d graph networks. *arXiv preprint arXiv:2102.05013*, 2021.

T. Pfaff, Meire Fortunato, Alvaro Sanchez-Gonzalez, and P. Battaglia. Learning mesh-based simulation with graph networks. *ArXiv*, abs/2010.03409, 2020.

R. Ramakrishnan, Pavlo O. Dral, M. Rupp, and O. A. von Lilienfeld. Quantum chemistry structures and properties of 134 kilo molecules. *Scientific Data*, 1, 2014.

Yu Rong, Wenbing Huang, Tingyang Xu, and Junzhou Huang. The truly deep graph convolutional networks for node classification. *CoRR*, abs/1907.10903, 2019. URL http://arxiv.org/abs/1907.10903.

Alvaro Sanchez-Gonzalez, N. Heess, Jost Tobias Springenberg, J. Merel, Martin A. Riedmiller, R. Hadsell, and P. Battaglia. Graph networks as learnable physics engines for inference and control. *ArXiv*, abs/1806.01242, 2018.

Alvaro Sanchez-Gonzalez*, Jonathan Godwin*, Tobias Pfaff*, Rex Ying*, Jure Leskovec, and Peter Battaglia. Learning to simulate complex physics with graph networks. In Hal Daumé III and Aarti Singh (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 8459–8468. PMLR, 13–18 Jul 2020. URL http://proceedings.mlr.press/v119/sanchez-gonzalez20a.html.

Victor Garcia Satorras, Emiel Hoogeboom, and Max Welling. E(n) equivariant graph neural networks, 2021.

Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80, 2009. doi: 10.1109/TNN.2008.2005605.

Kristof Schütt, Pieter-Jan Kindermans, Huziel Enoc Sauceda Felix, Stefan Chmiela, A. Tkatchenko, and K. Müller. Schnet: A continuous-filter convolutional neural network for modeling quantum interactions. In *NIPS*, 2017.

Jonathan Shlomi, Peter Battaglia, and Jean-Roch Vlimant. Graph neural networks in particle physics. *Machine Learning: Science and Technology*, 2(2):021001, Jan 2021. ISSN 2632-2153. doi: 10.1088/2632-2153/abbf9a. URL http://dx.doi.org/10.1088/2632-2153/abbf9a.

J. Sietsma and Robert J. F. Dow. Creating artificial neural networks that generalize. *Neural Networks*, 4:67–79, 1991.

Chuxiong Sun and Guoshi Wu. Adaptive graph diffusion networks with hop-wise attention. *ArXiv*, abs/2012.15024, 2020.

Shantanu Thakoor, C. Tallec, M. G. Azar, R. Munos, Petar Velivckovi'c, and Michal Valko. Bootstrapped representation learning on graphs. *ArXiv*, abs/2102.06514, 2021.

Nathaniel Thomas, Tess Smidt, Steven M. Kearnes, Lusann Yang, Li Li, Kai Kohlhoff, and Patrick Riley. Tensor field networks: Rotation- and translation-equivariant neural networks for 3d point clouds. *CoRR*, abs/1802.08219, 2018. URL http://arxiv.org/abs/1802.08219.

Oliver T. Unke and Markus Meuwly. Physnet: A neural network for predicting energies, forces, dipole moments, and partial charges. *Journal of Chemical Theory and Computation*, 15(6):3678–3693, May 2019. ISSN 1549-9626. doi: 10.1021/acs.jctc.9b00181. URL http://dx.doi.org/10.1021/acs.jctc.9b00181.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks, 2018.

Pascal Vincent, H. Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *ICML '08*, 2008.

Pascal Vincent, H. Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *J. Mach. Learn. Res.*, 11:3371–3408, 2010.

Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems*, 2020.

Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, C. Zhang, and Philip S. Yu. A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 32:4–24, 2021.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? *CoRR*, abs/1810.00826, 2018. URL http://arxiv.org/abs/1810.00826.

Chaoqi Yang, Ruijie Wang, Shuochao Yao, Shengzhong Liu, and Tarek Abdelzaher. Revisiting" over-smoothing" in deep gcns. *arXiv preprint arXiv:2003.13663*, 2020.

Chengxuan Ying, Tianle Cai, Shengjie Luo, Shuxin Zheng, Guolin Ke, Di He, Yanming Shen, and Tie-Yan Liu. Do transformers really perform bad for graph representation? *ArXiv*, abs/2106.05234, 2021.

Yuning You, Tianlong Chen, Yongduo Sui, Ting Chen, Zhangyang Wang, and Yang Shen. Graph contrastive learning with augmentations. *ArXiv*, abs/2010.13902, 2020.

Jie Zhou, Ganqu Cui, Shengding Hu, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. Graph neural networks: A review of methods and applications. *AI Open*, 1:57–81, 2020a.

Kuangqi Zhou, Yanfei Dong, Wee Sun Lee, Bryan Hooi, Huan Xu, and Jiashi Feng. Effective training strategies for deep graph neural networks. *CoRR*, abs/2006.07107, 2020b. URL https://arxiv.org/abs/2006.07107.

Table 8: OC20 IS2RS Test, Average Force below Threshold %, ↑

Model	Method	OOD Both	OOD Adsorbate	OOD Catalyst	ID
Noisy Nodes	Direct	0.09%	0.00%	0.29%	0.54%

Table 9: OC20 IS2RS Test, Force below Threshold %, ↑

Model	Method	OOD Both	OOD Adsorbate	OOD Catalyst	ID
Noisy Nodes	Direct	0.0%	0.0%	0.0%	0.0%

A Appendix

The following sections include details on training setup, hyper-parameters, input processing, as well as additional experimental results.

A.1 Additional Metrics for Open Catalyst IS2RS Test Set

Relaxation approaches to IS2RS minimise forces with respect to positions, with the expectation that forces at the minimum are close to zero. One metric of such a model's success is to evaluate the forces at the converged structure using ground truth Density Functional Theory calculations and see how close they are to zero. Two metrics are provided by OC20 (Chanussot* et al., 2020) on the IS2RS test set: Force below Threshold (FbT), which is the percentage of structures that have forces below 0.05 eV/Angstrom, and Average Force below Threshold (AFbT) which is FbT calculated at multiple thresholds.

The OC20 project computes test DFT calculations on the evaluation server and presents a summary result for all IS2RS position predictions. Such calculations take 10-12 hours and they are not available for the validation set. Thus, we are not able to analyse the results in Tables 8 and 9 in any further detail. Before application to catalyst screening further work may be needed for direct approaches to ensure forces do not explode from atoms being too close together.

A.2 More details on GNS adaptations for molecular property prediction.

Encoder.

The node features are a learned embedding lookup of the atom type, and in the case of OC20 two additional binary features representing whether the atom is part of the adsorbate or catalyst and whether the atom remains fixed during the quantum chemistry simulation.

The edge features, e_k are the distances $|d|$ featurised using c Radial Bessel basis functions, $\tilde{e}_{RBF,c} = \sqrt{\frac{2}{R}}\frac{\sin(\frac{c\pi}{R}d)}{d}$, and the edge vector displacements, d, normalised by the edge distance:

$$e_k = \text{Concat}(\tilde{e}_{RBF,1}(|d|), ..., \tilde{e}_{RBF,c}(|d|), \frac{d}{|d|})$$

Decoder

The decoder consists of two parts, a *graph-level decoder* which predicts a single output for the input graph, and a *node-level decoder* which predicts individual outputs for each node. The graph-level decoder implements the following equation:

$$y = W^{\text{Proc}} \sum_{i=1}^{|V|} \text{MLP}_{\text{Proc}}(a_i^{\text{Proc}}) + b^{\text{Proc}} + W^{\text{Enc}} \sum_{i=1}^{|V|} \text{MLP}_{\text{Enc}}(a_i^{\text{Enc}}) + b^{\text{Enc}}$$

Where a_i^{Proc} are node latents from the Processor, a_i^{Enc} are node latents from the Encoder, W^{Enc} and W^{Proc} are linear layers, b^{Enc} and b^{Proc} are biases, and $|V|$ is the number of nodes. The node-level decoder is simply an MLP applied to each a_i^{Proc} which predicts a_i^{Δ}.

A.3 Experiment setup

Open Catalyst. All training experiments were ran on a cluster of TPU devices. For the Open Catalyst experiments, each individual run (i.e. a single random seed) utilised 8 TPU devices on 2 hosts (4 per host) for training, and 4 V100 GPU devices for evaluation (1 per dataset).

Each Open Catalyst experiment was ran until convergence for up to 200 hours. Our best result, the large 100 layer model requires 7 days of training using the above setting. Each configuration was run at least 3 times in this hardware configuration, including all ablation settings.

We further note that making effective use of our regulariser requires sweeping noise values. These sweeps are dataset dependent and can be carried out using few message passing steps.

QM9. Experiments were also run on TPU devices. Each seed was run using 8 TPU devices on a single host for training, and 2 V100 GPU devices for evaluation. QM9 targets were trained between 12-24 hours per experiment.

Following Klicpera et al. (2020b) we define std. MAE as :

$$\text{std. MAE} = \frac{1}{M}\sum_{m=1}^{M}\left(\frac{1}{N}\sum_{i=1}^{N}\frac{|f_\theta^{(m)}(\boldsymbol{X}_i, \boldsymbol{z}_i) - \hat{t}_i^{(m)}|}{\sigma_m}\right)$$

and logMAE as:

$$\text{logMAE} = \frac{1}{M}\sum_{m=1}^{M}\log\left(\frac{1}{N}\sum_{i=1}^{N}\frac{|f_\theta^{(m)}(\boldsymbol{X}_i, \boldsymbol{z}_i) - \hat{t}_i^{(m)}|}{\sigma_m}\right)$$

with target index m, number of targets $M = 12$, dataset size N, ground truth values $\hat{t}^{(m)}$, model $f_\theta^{(m)}$, inputs $\boldsymbol{X}_i$ and z_i, and standard deviation σ_m of $\hat{t}^{(m)}$.

A.4 Hyper-parameters

Open Catalyst. We list the hyper-parameters used to train the default Open Catalyst experiment. If not specified otherwise (e.g. in ablations of these parameters), experiments were ran with this configuration.

Dynamic batch sizes refers to constructing batches by specifying maximum node, edge and graph counts (as opposed to only graph counts) to better balance computational load. Batches are constructed until one of the limits is reached.

Parameter updates were smoothed using an EMA for the current training step with the current decay value computed through $decay = min(decay, (1.0 + step)/(10.0 + step)$. As discussed in the evaluation, best results on Open Catalyst were obtained by utilising a 100 layer network with group size 10.

QM9 Table 11 lists QM9 hyper-parameters which primarily reflect the smaller dataset and geometries with fewer long range interactions. For U_0, U, H and G we use a slightly larger number of graphs per batch - 16 - and a smaller position loss co-efficient of 0.01.

OGBG-PCQM4M Table 12 provides the hyper parameters for OGBG-PCQM4M.

OGBG-MOLPCBA Table 13 provides the hyper parameters for the OGBG-MOLPCBA experiments.

OGBN-ARXIV Table 14 provides the hyper parameters for the OGBN-Arxiv experiments.

Table 10: Open Catalyst training parameters.

Parameter	Value or description
Optimiser	Adam with warm up and cosine cycling
β_1	0.9
β_2	0.95
Warm up steps	$5e5$
Warm up start learning rate	$1e-5$
Warm up/cosine max learning rate	$1e-4$
Cosine cycle length	$5e6$
Loss type	Mean squared error
Batch size	Dynamic to max edge/node/graph count
Max nodes in batch	1024
Max edges in batch	12800
Max graphs in batch	10
MLP number of layers	3
MLP hidden sizes	512
Number Bessel Functions	512
Activation	shifted softplus
message passing layers	50
Group size	10
Node/Edge latent vector sizes	512
Position noise	Gaussian ($\mu = 0, \sigma = 0.3$)
Parameter update	Exponentially moving average (EMA) smoothing
EMA decay	0.9999
Position Loss Co-efficient	1.0

Table 11: QM9 training parameters.

Parameter	Value or description
Optimiser	Adam with warm up and cosine cycling
β_1	0.9
β_2	0.95
Warm up steps	$1e4$
Warm up start learning rate	$3e-7$
Warm up/cosine max learning rate	$1e-4$
Cosine cycle length	$2e6$
Loss type	Mean squared error
Batch size	Dynamic to max edge/node/graph count
Max nodes in batch	256
Max edges in batch	4096
Max graphs in batch	8
MLP number of layers	3
MLP hidden sizes	1024
Number Bessel Funtions	512
Activation	shifted softplus
message passing layers	10
Group Size	10
Node/Edge latent vector sizes	512
Position noise	Gaussian ($\mu = 0, \sigma = 0.02$)
Parameter update	Exponentially moving average (EMA) smoothing
EMA decay	0.9999
Position Loss Coefficient	0.1

Table 12: OGBG-PCQM4M Training Parameters.

Parameter	Value or description
Optimiser	Adam with warm up and cosine cycling
β_1	0.9
β_2	0.95
Warm up steps	$5e4$
Warm up start learning rate	$1e-5$
Warm up/cosine max learning rate	$1e-4$
Cosine cycle length	$5e5$
Loss type	Mean absolute error
Reconstruction type	Softmax Cross Entropy
Batch size	Dynamic to max edge/node/graph count
Max nodes in batch	20,480
Max edges in batch	8,192
Max graphs in batch	512
MLP number of layers	2
MLP hidden sizes	512
Activation	relu
Node/Edge latent vector sizes	512
Dropnode Rate	0.1
Noisy Nodes Category Flip Fate	0.05
Parameter update	Exponentially moving average (EMA) smoothing
EMA decay	0.999
Reconstruction Loss Coefficient	0.1

Table 13: OGBG-PCQM4M Training Parameters.

Parameter	Value or description
Optimiser	Adam with warm up and cosine cycling
β_1	0.9
β_2	0.95
Warm up steps	$1e4$
Warm up start learning rate	$1e-5$
Warm up/cosine max learning rate	$1e-4$
Cosine cycle length	$1e5$
Loss type	Softmax Cross Entropy
Reconstruction loss type	Softmax Cross Entropy
Batch size	Dynamic to max edge/node/graph count
Max nodes in batch	20,480
Max edges in batch	8,192
Max graphs in batch	512
MLP number of layers	2
MLP hidden sizes	512
Activation	relu
Batch Normalization	Yes, after every hidden layer
Node/Edge latent vector sizes	512
Dropnode Rate	0.1
Dropout Rate	0.1
Noisy Nodes Category Flip Fate	0.05
Parameter update	Exponentially moving average (EMA) smoothing
EMA decay	0.999
Reconstruction Loss Coefficient	0.1

Table 14: OGBG-PCQM4M Training Parameters.

Parameter	Value or description
Optimiser	Adam with warm up and cosine cycling
β_1	0.9
β_2	0.95
Warm up steps	50
Warm up start learning rate	$1e-5$
Warm up/cosine max learning rate	$1e-3$
Cosine cycle length	$12,000$
Loss type	Softmax Cross Entropy
Reconstruction loss type	Mean Squared Error
Batch size	Full graph
MLP number of layers	1
Activation	relu
Batch Normalization	Yes, after every hidden layer
Node/Edge latent vector sizes	256
Dropout Rate	0.5
Noisy Nodes Input Dropout	0.05
Reconstruction Loss Coefficient	0.1