NOISY NODES: SIMPLE GNN 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 and bottleneck effects. 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 denoising node-level loss. Adding noise helps overfitting, and the denoising loss helps ameliorate oversmoothing by encouraging diverse node latents. We demonstrate that Noisy Nodes is compatible with multiple architectures, particularly very deep GNNs, and achieve state of the art on several challenging molecular prediction tasks. While many of these tasks have recently seen exciting advances from incorporating sophisticated mathematical methods into GNNs, our regulariser applies well-studied methods in simple, straightforward ways and can be added to any GNN architecture with little overhead. Our results suggest it 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), it has been proposed that in practice "oversmoothing" (Chen et al., 2019) and "bottleneck effects'((Alon & Yahav, 2020) limit their expressivity and ability to benefit from overparametrization. Here we propose a simple denoising regulariser, Noisy Nodes, and demonstrate how it overcomes these challenges across a range of datasets, tasks, and architectures, achieving state of the art results on OC20 IS2RS & IS2RE and QM9 tasks.

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 the expressive capacity, and so performance does not improve. Bottleneck effects are thought to limit the ability of a GNNs to communicate information over long ranges, because as the number of steps increases and causes the receptive fields to grow, the intermediate nodes must propagate information between larger and larger sets of peripheral nodes, limiting the ability to precisely transmit individual pieces of information (Alon & Yahav, 2020). We speculate these are key reasons why strong GNN results on the benchmarks we study here, for example DimeNet++ (Klicpera et al., 2020a), reports a depth of only 4 despite broad hyperparameter searches.

Our "Noisy Nodes" approach is an elegant, somewhat embarrassingly simple, technique for regularising GNNs architecture and associated training procedures. During training, our noise regularisation approach corrupts the input graph's attributes with noise, and incorporates an denoising loss. 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 and overfitting. 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. We find that applying Noisy Nodes reduces oversmoothing for shallower networks, and allows us to see monotonic improvements with added depth, even on tasks for which depth was assumed to be unhelpful.

First we demonstrate the compatability of Noisy Nodes with sophisticated molecular property prediction architectures, E(n)GNN and Neuqip on MD17. Then, using a standard simulation architecture, the Graph Net Simulator (GNS) (Sanchez-Gonzalez* et al., 2020), we improve state-of-the-art (SOTA) 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. Moreover, we apply Noisy Nodes with a Message Passing Neural Network (MPNN) with Virtual Node (Gilmer et al., 2017) on 2D molecular property prediction tasks and achieve state-of-the-art on OGBG-PCQM4M. Finally, while we find Noisy Nodes is particularly useful for molecular property prediction, when applied to a GCN trained on OGBN-Arxiv we again observe improved performance.

2 Related Work

Machine Learning for Molecular Property Prediction. One application of GNNs is to speed up quantum chemistry calculations which operate on a graph structured representation of a molecule (Duvenaud et al., 2015; Gilmer et al., 2017; Schütt et al., 2017; Hu et al., 2021). In these graphs atoms are nodes and edges are constructed between close 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 derivatives of such a model with respect to its positions are energy conserving rotation equivariant forces. 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). As higher order terms such as angles are added the size of the input representation can increase significantly (Klicpera et al., 2020b), and therefore the memory and compute requirements of the model.

An alternative approach to embedding symmetries is to design a rotation equivariant neural network that predict quantities such as forces directly (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). However, the benefits of such models are typically seen on smaller data (e.g. Satorras et al. (2021)), and such approaches have yet to achieve SOTA performance on large quantum chemistry datasets. Not all competitive models embed rotation symmetries—notably ForceNet (Hu et al., 2021) does not enforce rotation equivariance.

Scaling Graph Neural Networks with Depth. 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), but these studies have not shown consistent improvements as depth increases. Indeed a recent paper, "Training Graph Neural Networks with 1000 Layers" (Li et al., 2021), finds, as in previous work, (Li et al., 2019; 2020), that find optimal depth for the datasets they evaluate on to be far lower (5 for ogbgn-arxiv, for example). Another insight has been the analysis of the bottleneck effect in Alon & Yahav (2020) which may be alleviated by using a fully connected graph. Using these techniques has not led to improvements from adding depth on molecular property prediction tasks.

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 methodHu et al. (2020) 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.

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.

4 Noisy Nodes

4.1 Motivation

Methods that incentivise a GNN to maintain diverse node latents should ameliorate oversmoothing. One obvious way to do this would be to have very diverse node-level targets; in order to do well on such a task, the GNN must have diverse node 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. Take for example the goal of molecular property prediction - the goal is to predict a single scalar quantity per atomic graph.

A partial solution to such a problem is adding noise to the input node features, and a node level denoising loss. 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. To see this, consider a group of three atoms at equilibrium corrupted by noise, leading to two atoms very close together (and so have a very high interatomic force). By using the distances represented by edges of the graph, the GNN would be able to identify that the atoms are too close together and predict the denoised equilibrium geometry.

4.2 Definition

These observations lead us naturally to Noisy Nodes. The Noisy Nodes method 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 node level 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 noisy inputs as node-level 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.

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 correct data, the Noisy Nodes objective encourages the model to learn the manifold on which the correct 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). For graphs constructed from points in Euclidean space, such as atoms (Ramakrishnan et al., 2014), the manifold may include commonly repeated substructures that are useful for downstream prediction tasks.

The Energy Perspective for Molecular Property Prediction. Molecules in equilibrium are at a local minimum in the energy landscape. Local, random distortions of the geometry of the molecule are almost certainly higher energy configurations. As such, a task that maps from a noised molecule to the equilibrium geometry is learning a mapping from high energy to low energy - or from low probability to high probability in the boltzmann distribution.

We can construct something similar for problems where inputs V are not at equilibrium, but where we have a target displacement Δ, where the energy of $\tilde{V} + \Delta$ is less than $\tilde{V}$. For our noised positions $\tilde{V}$ we know to a high degree of certainty that energy$(V + \Delta) <$ energy$(V) <$ energy$(\tilde{V})$. Given that it is a desirable property to map from high energy to low energy, we can set the target of our noised positions to $\Delta - \sigma$. Where Δ is the per atom forces, $\Delta - \sigma$ will likely be closer to the real forces of $\tilde{V}$ than σ. This is the denoising target we use for our 3D molecular prediction tasks with existing node displacement targets, and is illustrated in Figure 1.

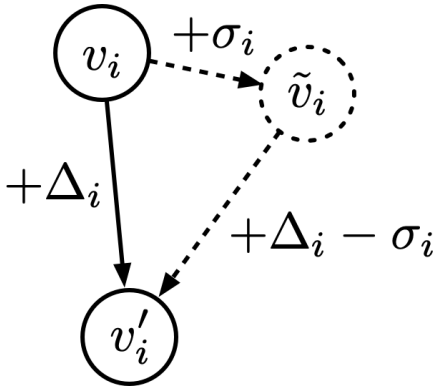

Figure 1: 3D Noisy Node mechanics during training. Input positions are corrupted with noise σ, and the training objective is to node-level difference between target positions and the noisy inputs.

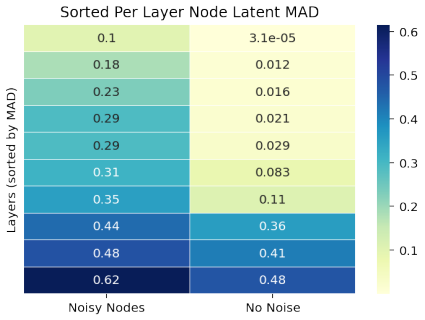

Figure 2: Residual Activation Per Layer, sorted by MAD. 10 Layer Model trained on QM9. We can see that Noisy Nodes encourages the model to make diverse residual updates per node - ameliorating oversmoothing. For Noisy Nodes all layers have a MAD above 0.1, compared to 4 for No Noise.

5 3D Molecular Property Prediction Experiments and Results

We tested Noisy Nodes on four challenging molecular property prediction benchmarks: MD17, OC20 (Chanussot* et al., 2020) IS2RS & IS2RE, and QM9 (Ramakrishnan et al., 2014). These benchmarks are detailed below, but as general distinctions, OC20 uses graphs 2-20x larger than QM9 or MD17, and so we expected greater benefits on OC20 from deeper message passing. 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. MD17 requires node and graph level predictions.

Table 1: MD17 Forces, eV/Angstrom MAE, ↓

	Neuqip ($l = 1$) [1]	+ Noisy Nodes	% Improvement
Aspirin	25.1	**22.6**	**10.05%**
Ethanol	7.8	**7.5**	**4.64%**
Malonaldehyde	22.5	**19.4**	**16.2%**
Naphthalene	4.9	**3.9**	**13.43%**
Salicylic	12.9	**10.5**	**20.1%**
Toluene	6.9	**5.7**	**18.49%**
Uracil	9.9	**8.1**	**17.80%**

We first evaluate Noisy Nodes combined with Neuqip (Batzner et al., 2021), a recent equivariant architecture designed for molecular property prediction, on the MD17 dataset and find improvements from the use of Noisy Nodes.

Then, we evaluate Noisy Nodes on OC20 and QM9 using the GNS (Sanchez-Gonzalez* et al., 2020) architecture, a general purpose simulation architecture that has previoulsy been adapted for quantum modeling (Hu et al., 2021). We featurize the distance input features using radial basis functions. For this architecture, we use the concept of 'grouping' from (Li et al., 2021) to reduce the parameter count of deep networks. Additionally we find that adding a loss after each group aids training stability.

5.1 Training

We minimise the mean squared error loss on mean and standard deviation normalised targets in using the Adam (Kingma & Ba, 2015) optimiser with warmup and cosine decay. For 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. Neuqip with rotation order 1 was reimplemented and uses the same hyper parameters as the paper. OC20 & QM9 Models were trained on 8 TPU devices and evaluated on V100 GPUs. MD17 was trained on a single V100 GPU for 12 hours. QM9 models were trained for up to 24 hours, and OC models for up to 200 hours. We provide the full set of hyper-parameters and computational resources used separately for each dataset in the Appendix.

Our model 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.

5.2 MD17

Dataset.

Source of Noisy for Noisy Nodes.

Results.

5.3 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 of the community challenges; the Initial Structure to Resulting Energy (IS2RE) task which takes the initial structure of the simulation and predicts the energy at the relaxed state, and the Initial Structure to Resulting Structure (IS2RS) which takes the initial structure and predicts the relaxed state structure. These are both very challenging tasks with benchmark relative errors far higher than those seen on QM9.

[0]hello

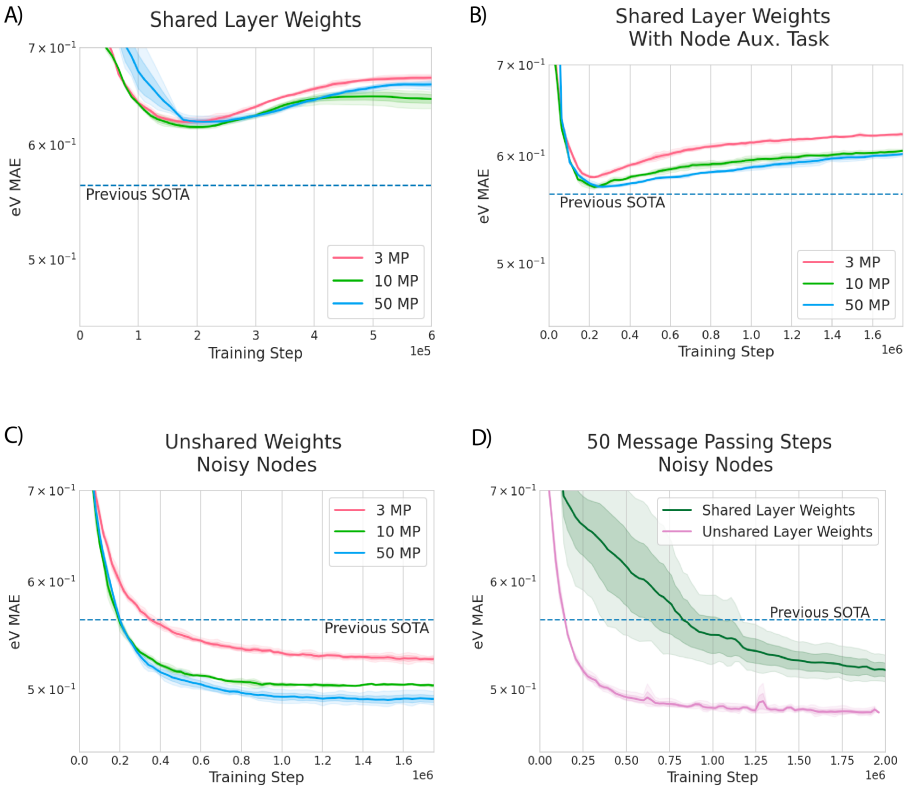

Figure 3: Validation curves on OC20 IS2RE ID. **A)** Without any node targets our model has poor performance and realises no benefit from depth, reflecting commonly held views about scaling GNNs for molecular property prediction. **B)** After adding a position node loss, we see performance improves as the number of message passing layers increases even with the same number of parameters. However, we still see overfitting. **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 fully shared weights to achieve SOTA.

Table 2: 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 $\pm$0.01	0.65 $\pm$0.01	0.55 $\pm$0.00	0.54 $\pm$0.00
GNS-Shared + Noisy Nodes	50	0.49 $\pm$0.00	0.54 $\pm$0.00	0.51 $\pm$0.01	0.51 $\pm$0.01
GNS + Noisy Nodes	50	0.48 $\pm$0.00	0.53 $\pm$0.00	0.49 $\pm$0.01	0.48 $\pm$0.00
GNS-10 + Noisy Nodes	100	**0.46**$\pm$0.00	**0.51** $\pm$0.00	**0.48** $\pm$0.00	**0.47** $\pm$0.00

Note that we train the more common "direct" prediction models that map directly from initial positions to target in a single forward pass, and compare against other models train for direct prediction. We do not train "relaxation" models, which have reported fewer results, use far more data and take longer to train.

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.

Source of Noise for Noisy Nodes. During training we first sample uniformly from a point in the relaxation trajectory (alternatively we can sample a point from the initial to final position interpolation), and then add I.I.D Gaussian noise with mean zero and $\sigma^2 = 0.3$. The Noisy Node target is the relaxed structure.

Additional Training Details. We first convert to fractional coordinates 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 transformation matrix. This additional vector provides rotation invariant angular and extent information to the GNN.

IS2RE Results. In Figure 3 we show that depth improves performance for IS2RE. Figure 3 A) shows that without using the any node auxiliary targets there is no benefit to be gained adding depth. However, once the Noisy Nodes objective is used, performance increases monotonically as depth increases. Furthermore, Figure 3 D) shows our model can achieve SOTA with a single message passing step recurrently applied 50 times.

In Table 2 we conduct an ablation on our hyperparameters and find again that using Noisy Nodes allows us to realise the benefits of depth. Interestingly we find that increased parameter counts do not necessarily improve performance—our best model a 100 layer model with a group size of 10. Results were averaged over 3 seeds and standard errors on the best obtained checkpoint show little sensitivity to initialisation. Further analysis of the effect of Noisy Nodes on GNN activations can be found in the Appendix.

Our best model on IS2RE was a 100 layer GNN with a group size of 10 which achieved a 43.3% relative performance improvement against SOTA results (Table 3). Due to limited permitted test submissions, results presented here were from one test upload of our best performing validation seed.

IS2RS. In Table 4 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 5 we see that our model is significantly better than the only other reported IS2RS direct result, ForceNet, which makes our results state of the art.

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 position predictions, we speculate that adding more capacity to the positional decoder head may help ameliorate such effects but leave that to future work.

Table 3: Results OC20 IS2RE Test

eV MAE ↓					
	SchNet	DimeNet++	SpinConv	SphereNet	GNS + Noisy Nodes
OOD Both	0.704	0.661	0.674	0.638	**0.504 (-21.0%)**
OOD Adsorbate	0.734	0.725	0.723	0.703	**0.565 (-19.6%)**
OOD Catalyst	0.662	0.576	0.569	0.571	**0.493 (-13.7%)**
ID	0.639	0.562	0.558	0.563	**0.478 (-15.1%)**
Average Energy within Threshold (ADwT) ↑					
	SchNet	DimeNet++	SpinConv	SphereNet	GNS + Noisy Nodes
OOD Both	0.0221	0.0241	0.0233	0.0241	**0.0382 (+58.5%)**
OOD Adsorbate	0.0233	0.0207	0.026	0.0229	**0.0349 (+52.4%)**
OOD Catalyst	0.0294	0.0410	0.0382	0.0409	**0.0550 (+34.5%)**
ID	0.0296	0.0425	0.0408	0.0447	**0.0571 (+27.7%)**

Table 4: 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 5: OC20 IS2RS Test, ADwT, ↑

Model	OOD Both	OOD Adsorbate	OOD Catalyst	ID
ForceNet	46.9%	37.7%	43.7%	44.9%
GNS + Noisy Nodes	**52.5%**	**43.7%**	**47.8%**	**50.0%**
Relative Improvement	**+11.9%**	**+15.9%**	**+9.4%**	**+11.4%**

Table 6: QM9, Best Noisy Nodes, 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		**0.88**
logMAE		-5.17	-5.43	-5.67	**-5.68**		-5.60

5.4 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 but with identical hyper parameters.

Source of Noise for Noisy Nodes. 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 2 we can see that adding the denoising objective significantly improves results by 23.1% relative. Furthermore increased depth monotonically increases performance.

Our best model is a 30 layer network with a group size of 10 which achieves SOTA on 4 of the 12 targets and comparable performance on the remainder (Table 6). On the std. MAE aggregate metric our model performs better than all other reported results, and is thus SOTA. This is despite having no equivariance or invariance symmetries, or hand crafted molecular features.

$\langle R^2 \rangle$, the electronic spatial extent, is an outlier for our model. Interestingly, we found that without noise our model achieves 0.33 for this target. We speculate that this target is particularly sensitive to noise, and the best noise value would be significantly lower than for the dataset as a whole.

Since there are no fractional co-ordinates for QM9, the GNS implementation in not by default invariant to rotations. To show that Noisy Nodes does not rely on non-invariant features we evaluate an invariant version of the network that uses the principle axes of inertia ordered by eigenvalue as the co-ordinate frame (Table 7). Indeed, we see here that similar performance is achieved with rotation invariant features.

6 Other Tasks

6.1 OGBG-PCQM4M

To test whether the effects of noisy nodes are due to the use of 3D coordinates, we evaluate our model on 2D molecular graphs - consisting of bonds and atom types. To adapt Noisy Nodes to

Table 7: QM9 Ablation

	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 8: OGBG-PCQM4M Results

Model	Number of Layers	Using Noisy Nodes	MAE
MPNN + Virtual Node	16	Yes	0.1249 ± 0.0003
MPNN + Virtual Node	50	No	0.1236 ± 0.0001
Graphormer	-	-	0.1234
MPNN + Virtual Node	50	Yes	$\mathbf{0.1218 \pm 0.0001}$

this setting, we randomly flip node and edge features and add a reconstruction loss at a rate of 5%. Additionally, we must use a different model since GNS is defined on 3D Co-ordinates. We use the OGBG-PCQM4M dataset, consisting of 3.8m 2D graphs and associated HOMO-LUMO gaps. We use an MPNN with Virtual Node as our base model. The test set is not currently available for this dataset, but results will be updated when available.

In Table 8 we see that for this task, as in the 3D prediction tasks, Noisy Nodes enables at 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 OGBN-ARXIV

Whilst for this paper we focus on molecular property prediction task, the Noisy Nodes recipe could be applied to any type of graph. To test how far elements of the recipe generalise, we evaluate Noisy Nodes on OGBN-ARXIV. This is challenging task for Noisy Nodes to benefit - there is already a diverse node level target, and standard models already add input noise via input dropout.

Adding Noisy Nodes, with input dropout 0.1, to 4 layer GCN with residual connections improves from 72.09% accuracy to 72.51% accuracy. The best performing model, without the use of the 'Bag of Tricks' (cite) or self knowledge distillation (cite), is ADGN (cite) at 73.75%. This result demonstrates that there are benefits to using the Noisy Nodes even in the shallow networks applied to node level, non-molecular prediction tasks.

6.3 LIMITATIONS

Realising benefits from Noisy Nodes relies upon tuning two additional hyperparameters - the type of noise used, and the level of noise applied. This creates additional experimentation requirements.

Noisy Nodes has a particular effect on depth, but depth is not always an advantage. The GNS model we chose requires many layers to achieve state of the art performance - alternative architectures require far fewer. For situations in which it is not computationally feasible to use many layers using Noisy Nodes may not bring improvements, however we note that even at 3 layers the GNS architecture achieves state of the art direct validation energy predictions because of Noisy Nodes. We also note that some of the best performing alternative architectures such as DimeNet++ and GemNet can have worse time and memory complexity than very deep GNSs because the input representation is $\mathcal{O}(n^2)$ where n is the number of atoms. For example, for a system with 200 atoms a single layer of DimeNet is 200x more expensive than a layer with $\mathcal{O}(n)$ input representation. However, architectures such as PaiNN or SpinConv do not have this limitation.

In addition, Noisy Nodes has not been demonstrated in the small data regime on datasets such as MD17 (Chmiela et al., 2017), a regime in which models such as DimeNet++ perform well. This regime has particular importance for learning interatomic potentials since highly accurate experimental data is typically small.

7 CONCLUSIONS

In this work we present Noisy Nodes, a novel regularisation technique for graph neural networks. Noisy nodes helps address common challenges around oversmoothed node representations and in particular improves performance for deeper GNNs.

8 REPRODUCIBILITY STATEMENT

- Code?

- Comment on computational cost of main result?

- Comment on data preprocessing

9 ETHICS STATEMENT

Who may benefit from this work? Molecular property prediction with deep networks 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 is also a general technique. It could be adapted and applied to many areas in which GNNs are used—for example, knowledge base completion, physical simulation or traffic prediction. We hope that the benefits of training GNNs with depth could be realised in these domains.

Potential negative impact and reflection. Enabling 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. In future work we plan to further investigate depth-efficiency tradeoffs in GNNs.

REFERENCES

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

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

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, 2021.

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

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.

Yaping Liu, Limei Wang, Min Liu, Xuan Zhang, Bora Oztekin, and Shuiwang Ji. Spherical message passing for 3d graph networks. *ArXiv*, abs/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.

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`.

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.

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

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`.

A APPENDIX

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

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

Model	Method	OOD Both	OOD Adsorbate	OOD Catalyst	ID
Noisy Nodes	Direct	0.00%	0.00%	0.00%	0.63%

Table 10: 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.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.

However, force evaluation is arguably not a good metric of structural fidelity. Very small physical error at tiny inter-atomic distances can lead to an extremely large force error if atoms become too close together. Such a large force, caused by a very small positional error, would lead to failure on the force metrics even though the structures are very similar on distance measures. For this reason we do not report force metrics in the main body of the paper for our direct prediction results, since our positional model was not trained with any force data and we expect it to make such small positional errors. However, we present them here for completeness, since they are computed automatically upon submission.

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 9 and 10 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 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.3 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.

Table 11: 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

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 12 lists QM9 hyper-parameters which primarily reflect the smaller dataset and geometries with fewer long range interactions.

Table 12: 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