LARGE-SCALE GRAPH REGRESSION WITH VERY DEEP GNNS

OPEN GRAPH BENCHMARK LARGE-SCALE CHALLENGE ENTRY (PCQM4M)

Ravichandra Addanki[*]
DeepMind
ravichandraa@deepmind.com

Peter W. Battaglia[*]
DeepMind
peterbattaglia@deepmind.com

David Budden[*]
DeepMind
budden@deepmind.com

Andreea Deac[*]
DeepMind, Mila, Université de Montréal
adeac@deepmind.com

Jonathan Godwin[*]
DeepMind
jonathangodwin@deepmind.com

Wai Lok Sibon Li[*]
DeepMind
sibon@deepmind.com

Alvaro Sanchez-Gonzalez[*]
DeepMind
alvarosg@deepmind.com

Jacklynn Stott[*]
DeepMind
jacklynnstott@deepmind.com

Shantanu Thakoor[*]
DeepMind
thakoor@deepmind.com

Petar Veličković[*]
DeepMind
petarv@deepmind.com

ABSTRACT

Effectively and efficiently deploying graph neural networks (GNNs) at scale remains one of the most challenging aspects of graph representation learning. Many powerful solutions have only ever been validated on comparatively small datasets, often with counter-intuitive outcomes—a barrier which has recently been broken by the Open Graph Benchmark Large-Scale Challenge (OGB-LSC). Here we describe our OGB-LSC entry for the PCQM4M-LSC benchmark: a very deep (up to 50-layer) inductive graph regressor regularised by denoising objectives. Our model achieved an award-level (top-3) performance. In doing so, we demonstrate strong evidence of scalable self-supervised graph representation learning, and utility of very deep GNNs—both very important open issues. Our code is publicly available at: `https://github.com/deepmind/deepmind-research/tree/master/ogb_lsc/pcq`.

Keywords OGB-LSC · Graph Networks · Noisy Nodes

1 Introduction

Effective high-dimensional representation learning necessitates properly exploiting the geometry of data [Bronstein et al., 2021]—otherwise, it is a cursed estimation problem. Indeed, early success stories of deep learning relied on imposing strong geometric assumptions, primarily that the data lives on a grid domain; either spatial or temporal. In these two respective settings, convolutional neural networks (CNNs) [LeCun et al., 1998] and recurrent neural networks (RNNs) [Hochreiter and Schmidhuber, 1997] have traditionally dominated.

While both CNNs and RNNs are demonstrably powerful models, with many applications of high interest, it can be recognised that most data coming from nature cannot be natively represented on a grid. Recent years are marked with a gradual shift of attention towards models that admit a more generic class of geometric structures [Masci et al., 2015, Veličković et al., 2017, Cohen et al., 2018, Battaglia et al., 2018, de Haan et al., 2020, Satorras et al., 2021].

[*]All authors contributed equally.

In many ways, the most generic and versatile of these models are graph neural networks (GNNs). This is due to the fact that most discrete-domain inputs can be observed as instances of a graph structure. The corresponding area of graph representaton learning [Hamilton, 2020] has already seen immense success across industrial and scientific disciplines. GNNs have successfully been applied for drug screening [Stokes et al., 2020], modelling the dynamics of glass [Bapst et al., 2020], web-scale social network recommendations [Ying et al., 2018] and chip design [Mirhoseini et al., 2020].

While the above results are certainly impressive, they likely only scratch the surface of what is possible with well-tuned GNN models. Many problems of real-world interest require graph representation learning *at scale*: either in terms of the amount of graphs to process, or their sizes (in terms of numbers of nodes and edges). Perhaps the clearest motivation for this comes from the Transformer family of models [Vaswani et al., 2017]. Transformers operate a self-attention mechanism over a complete graph, and can hence be observed as a specific instance of GNNs [Joshi, 2020]. At very large scales of natural language data, Transformers have demonstrated significant returns with the increase in capacity, as exemplified by models such as GPT-3 [Brown et al., 2020]. Transformers enjoy favourable scalability properties at the expense of their functional complexity: each node's features are updated with *weighted sums* of neighbouring node features. In contrast, GNNs that rely on message passing [Gilmer et al., 2017]—passing vector signals across edges that are conditioned on *both* the sender and receiver nodes—are an empirically stronger class of models, especially on tasks requiring complex reasoning [Veličković et al., 2019] or simulations [Sanchez-Gonzalez et al., 2020, Pfaff et al., 2020].

One reason why generic message-passing GNNs have not been scaled up as widely as Transformers is the lack of appropriate datasets. Only recently has the field advanced from simple transductive benchmarks of only few thousands of nodes [Sen et al., 2008, Shchur et al., 2018, Morris et al., 2020] towards larger-scale real-world and synthetic benchmarks [Dwivedi et al., 2020, Hu et al., 2020], but important issues still remain. For example, on many of these tasks, randomly-initialised GNNs [Veličković et al., 2018], shallow GNNs [Wu et al., 2019] or simple label propagation-inspired GNNs [Huang et al., 2020] can perform near the state-of-the-art level at only a fraction of the parameters. When most bleeding-edge expressive methods are unable to improve on the above, this can often lead to controversial discussion in the community. One common example is: *do we even need deep, expressive GNNs?*

Breakthroughs in deep learning research have typically been spearheaded by impactful large-scale competitions. For image recognition, the most famous example is the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) [Russakovsky et al., 2015]. In fact, the very "deep learning revolution" has partly been kickstarted by the success of the AlexNet CNN model of Krizhevsky et al. [2012] at the ILSVRC 2012, firmly establishing deep CNNs as the workhorse of image recognition for the forthcoming decade.

Accordingly, we have entered the recently proposed Open Graph Benchmark Large-Scale Challenge (OGB-LSC) [Hu et al., 2021]. OGB-LSC provides graph representation learning tasks at a previously unprecedented scale—millions of nodes, billions of edges, and/or millions of graphs. Further, the tasks have been designed with immediate practical relevance in mind, and it has been verified that expressive GNNs are likely to be necessary for strong performance. Here we detail our submitted model for the PCQM4M task, and our empirical observations while developing it. Namely, we find that the dataset's immense scale provides a great platform for demonstrating clear outperformance of very deep GNNs, powered by denoising objectives [Godwin et al., 2021]. In doing so, we have provided meaningful evidence towards a positive resolution to the above discussion: deep and expressive GNNs are, indeed, necessary at the right level of task scale and/or complexity. Our final model has achieved award-level (top-3) ranking on PCQM4M.

2 Dataset description

The **PCQM4M-LSC** dataset is an inductive graph regression dataset based on the PubChemQC project [Nakata and Shimazaki, 2017]. It consists of $\sim$4 million small molecules (described by their SMILES strings). The aim is to accelerate quantum-chemical computations: especially, to predict the HOMO-LUMO gap of each molecule. The HOMO-LUMO gap is one of the most important quantum-chemical properties, since it is related to the molecules' reactivity, photoexcitation, and charge transport. The ground-truth labels for every molecule were obtained through expensive DFT (density functional theory) calculations, which may take several hours per molecule. It is believed that machine learning models, such as GNNs over the molecular graph, may obtain useful approximations to the DFT at only a fraction of the computational cost, if provided with sufficient training data [Gilmer et al., 2017]. The molecules are split with a 80:10:10 ratio into training, validation and test sets, based on their PubChem ID.

3 GNN Architectures

We rely on a common *encode-process-decode* blueprint [Hamrick et al., 2018]. This implies that our input features are encoded into a latent space using node-, edge- and graph-wise *encoder* functions, and latent features are decoded to node-, edge- and graph- level predictions using appropriate *decoder* functions. The bulk of the computational processing is powered by a *processor* network, which performs multiple graph neural network layers over the encoded latents.

To formalise this, assume that our input graph, $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, has node features $\mathbf{x}_u \in \mathbb{R}^n$, edge features $\mathbf{x}_{uv} \in \mathbb{R}^m$ and graph-level features $\mathbf{x}_\mathcal{G} \in \mathbb{R}^l$, for nodes $u, v \in \mathcal{V}$ and edges $(u, v) \in \mathcal{E}$. Our encoder functions $f_n : \mathbb{R}^n \to \mathbb{R}^k$, $f_e : \mathbb{R}^m \to \mathbb{R}^k$ and $f_g : \mathbb{R}^l \to \mathbb{R}^k$ then transform these inputs into the latent space:

$$\mathbf{h}_u^{(0)} = f_n(\mathbf{x}_u) \qquad \mathbf{h}_{uv}^{(0)} = f_e(\mathbf{x}_{uv}) \qquad \mathbf{h}_\mathcal{G}^{(0)} = f_g(\mathbf{x}_\mathcal{G}) \tag{1}$$

Our processor network then transforms these latents over several rounds of message passing:

$$\mathbf{H}^{(t+1)} = P_{t+1}(\mathbf{H}^{(t)}) \tag{2}$$

where $\mathbf{H}^{(t)} = \left(\left\{\mathbf{h}_u^{(t)}\right\}_{u\in\mathcal{V}}, \left\{\mathbf{h}_{uv}^{(t)}\right\}_{(u,v)\in\mathcal{E}}, \mathbf{h}_\mathcal{G}^{(t)}\right)$ contains all of the latents at a particular processing step $t \geq 0$.

The processor network P is iterated for T steps, recovering final latents $\mathbf{H}^{(T)}$. These can then be decoded into node-, edge-, and graph-level predictions (as required), using analogous decoder functions g_n, g_e and g_g:

$$\mathbf{y}_u = g_n(\mathbf{h}_u^{(T)}) \qquad \mathbf{y}_{uv} = g_e(\mathbf{h}_{uv}^{(T)}) \qquad \mathbf{y}_\mathcal{G} = g_g(\mathbf{h}_\mathcal{G}^{(T)}) \tag{3}$$

We will detail the specific design of f, P and g in the following sections. Generally, f and g are simple MLPs, whereas we use highly expressive GNNs for P in order to maximise the advantage of the large-scale datasets. Specifically, we use graph networks (GNs) [Battaglia et al., 2018]. All of our models have been implemented using the jraph library [Godwin et al., 2020].

4 PCQM4M-LSC setup

Input preprocessing For featurising our molecules within PCQM4M, we initially follow the baseline scripts provided by Hu et al. [2021] to convert SMILES strings into molecular graphs. Therein, every node is represented by a 9-dimensional feature vector, $\mathbf{x}_u$, including properties such as atomic number and chirality. Further, every edge is endowed with 3-dimensional features, $\mathbf{x}_{uv}$, including bond types and stereochemistry. Mirroring prior work with GNNs for quantum-chemical computations [Gilmer et al., 2017], we found it beneficial to maintain graph-level features (in the form of a "master node"), which we initialise at $\mathbf{x}_\mathcal{G} = \mathbf{0}$.

As will soon become apparent, our experiments on the PCQM4M benchmark leveraged GNNs that are substantially deeper than most previously studied GNNs for quantum-chemical tasks, or otherwise. While there is implicit expectation to compute useful "cheap" chemical features from the SMILES string, such as molecular fingerprints, partial charges, etc., our experiments clearly demonstrated that most of them do not meaningfully impact performance of our GNNs. This indicates that very deep GNNs are likely implicitly able to compute such features without additional guidance.

The exception to this have been *conformer features*, corresponding to approximated three-dimensional coordinates of every atom. These are very expensive to obtain accurately. However, using RDKit [Landrum, 2013], we have been able to obtain conformer estimates that allowed us to attain slightly improved performance with a (slightly) shallower GNN. Specifically, we use the experimental torsion knowledge distance geometry (ETKDGv3) algorithm [Wang et al., 2020] to recover conformers that satisfy essential geometric constraints, without violating our time limits.

Once conformers are obtained, we do not use their raw coordinates as features—these have many equivalent formulations that depend on the algorithm's initialisation. Instead, we encode their displacements (a 3-dimensional vector recording distances along each axis) and their distances (scalar norm of the displacement) as additional edge features concatenated with $\mathbf{x}_{uv}$. Note that RDKit's algorithm is not powerful enough to extract conformers for every molecule within PCQM4M; for about 0.1% of the dataset, the returned conformers will be NaN.

Lastly, we also attempted to use more computationally intensive forms of conformer generation—including energy optimisation using the universal force field (UFF) [Rappé et al., 1992] and the Merck molecular force field (MMFF) [Halgren, 1996]. In both cases, we did not observe significant returns compared to using rudimentary conformers.

Model architecture For the GNN architecture we have used on PCQM4M, our encoders and decoders are both three-layer MLPs, computing 512 features in every hidden layer. The node, edge and graph-level encoders' output layers compute 512 features, and we retain this dimensionality for $\mathbf{h}_u^{(t)}$, $\mathbf{h}_{uv}^{(t)}$ and $\mathbf{h}_\mathcal{G}^{(t)}$ across all steps t.

For our processor network, we use a very deep Graph Network (GN) [Battaglia et al., 2018]. Each GN block computes updated node, edge and graph latents, performing aggregations across them whenever appropriate. Fully expanded out, the computations of one GN block can be represented as follows:

$$\mathbf{h}_{uv}^{(t+1)} = \psi_{t+1}\left(\mathbf{h}_u^{(t)}, \mathbf{h}_v^{(t)}, \mathbf{h}_{uv}^{(t)}, \mathbf{h}_{\mathcal{G}}^{(t)}\right) \tag{4}$$

$$\mathbf{h}_u^{(t+1)} = \phi_{t+1}\left(\mathbf{h}_u^{(t)}, \sum_{u\in\mathcal{N}_v} \mathbf{h}_{vu}^{(t+1)}, \mathbf{h}_{\mathcal{G}}^{(t)}\right) \tag{5}$$

$$\mathbf{h}_{\mathcal{G}}^{(t+1)} = \rho_{t+1}\left(\sum_{u\in\mathcal{V}} \mathbf{h}_u^{(t+1)}, \sum_{(u,v)\in\mathcal{E}} \mathbf{h}_{uv}^{(t+1)}, \mathbf{h}_{\mathcal{G}}^{(t)}\right) \tag{6}$$

Taken together, Equations 4–6 fully specify the operations of the P_{t+1} network in Equation 2. The edge update function ψ_{t+1}, node update function ϕ_{t+1} and graph update function ρ_{t+1} are all three-layer MLPs, with identical hidden and output sizes to the encoder network.

The process is repeated for $T = 32$ message passing layers, after which the computed latents $\mathbf{h}_u^{(32)}$, $\mathbf{h}_{uv}^{(32)}$ and $\mathbf{h}_{\mathcal{G}}^{(32)}$ are sent to the decoder network for relevant predictions. Specifically, the global latent vector $\mathbf{h}_{\mathcal{G}}^{(32)}$ is used to predict the molecule's HOMO-LUMO gap. Our work thus constitutes a successful application of very deep GNNs, providing evidence towards ascertaining positive utility of such models. We note that, while most prior works on GNN modelling seldom use more than eight steps of message passing [Brockschmidt, 2020], we observe monotonic improvements of deeper GNNs on this task, all the way to 32 layers when the validation performance plateaus.

Non-conformer model Recalling our prior discussion about conformer features occasionally not being trivially computable, we also trained a GN which does not exploit conformer-based features. While we observe largely the same trends, we find that they tend to allow for even deeper and wider GNNs before plateauing. Namely, our optimised non-conformer GNN computes 1,024-dimensional hidden features in every MLP, and iterates Equations 4–6 for $T = 50$ message passing steps. Such a model performed marginally worse than the conformer GNN overall, while significantly improving the mean absolute error (MAE) on the 0.1% of validation molecules without conformers.

Denoising objective Our very deep GNNs have, in the first instance, been enabled by careful regularisation. By far, the most impactful method for our GNN regressor on PCQM4M has been Noisy Nodes [Godwin et al., 2021], and our results largely echo the findings therein.

The main observation of Noisy Nodes is that very deep GNNs can be strongly regularised by appropriate denoising objectives. Noisy Nodes perturbs the input node or edge features in a pre-specified way, then requires the decoder to reconstruct the un-perturbed information from the GNN's latent representations.

In the case of the flat input features, we have deployed a Noisy Nodes objective on both atom types and bond types: randomly replacing each atom and each bond type with a uniformly sampled one, with probability $p = 0.05$. The model then performs node/edge classification based on the final latents (e.g., $\mathbf{h}_u^{(32)}$, $\mathbf{h}_{uv}^{(32)}$ for the conformer GNN), to reconstruct the initial types. Requiring the model to correctly infer and rectify such noise is implicitly imbuing it with knowledge of chemical constraints, such as valence, and is a strong empirical regulariser. Note that, in this discrete-feature setting, Noisy Nodes can be seen as a more general case of the BERT-like objectives from Hu et al. [2019]. The main difference is that Noisy Nodes takes a more active role in requiring denoising—as opposed to unmasking, where it is known upfront which nodes have been noised, and the effects of noising are always predictable.

When conformers or displacements are available, a richer class of denoising objectives may be imposed on the GNN. Namely, it is possible to perturb the individual nodes' coordinates slightly, and then require the network to reconstruct the original displacement and/or distances—this time using edge regression on the output latents of the processor GNN. The Noisy Nodes manuscript had shown that, under such perturbations, it is possible to achieve state-of-the-art results on quantum chemical calculations *without* requiring an explicitly equivariant architecture—only a very deep traditional GNN. Our preliminary results indicate a similar trend on the PCQM4M dataset.

Training and regularisation We train our GNN to minimise the mean absolute error (MAE) for predicting the DFT-simulated HOMO-LUMO gap based on the decoded global latent vectors. This objective is combined with any auxiliary tasks imposed by noisy nodes (e.g. cross-entropy on reconstructing atom and bond types, MAE on regressing denoised displacements). We use the Adam SGD optimiser [Kingma and Ba, 2014] with hyperparameters $\beta_1 = 0.9$, $\beta_2 = 0.95$. We use a cosine learning rate schedule with initial learning rate $\eta = 10^{-5}$ and $50,000$ warm-up steps, peaking at $\eta = 10^{-4}$, and decaying over $500,000$ training iterations. Optimisation is performed over dynamically-batched data: we fill up each training minibatch with molecules until any of the following limits are exceeded: $1,024$ atoms, $2,560$ bonds, or 64 molecules.

To regularise our model, we perform early stopping on the MAE over the validation dataset, and apply feature dropout [Srivastava et al., 2014] (with $p = 0.1$) and DropEdge [Rong et al., 2019] (with $p = 0.1$) at every message passing layer of the GNN.

Evaluation At evaluation time, we exploit several known facts about the HOMO-LUMO gap, and our conformer generation procedure, to achieve "free" reductions in MAE.

Firstly, it is known that the HOMO-LUMO gap cannot be negative, and that it is possible for our model to make (very rare) vastly inflated predictions on validation data if it encounters an out-of-distribution molecule. We ameliorate both of these issues by clipping the network's predictions in the $[0, 20]$ eV range.

Secondly, as discussed on several occasions, for a very small fraction (0.1% of molecules), RDKit was unable to compute conformers. We found that it was useful to fall back to the 50-layer non-conformer GNN in these cases, rather than assuming a default value. The observed reductions in MAE were significant across those specific validation molecules only.

Finally, we consistently track the exponential moving average (EMA) of our model's parameters (with decay rate $\epsilon = 0.9999$), and use it for evaluation. EMA parameters are generally known to be more stable than their online counterparts, an observation that held in our case as well.

5 Ensembling and training on validation

Once we established the top single-model architecture for our PCQM4M entry, we found it very important to perform two post-processing steps: (a) re-train on the validation set, (b) ensemble various models together.

Re-training on validation data offers a great additional wealth of learning signal, even just by the sheer volume of data available in the OGB-LSC.

However, training on the full validation set comes with a potentially harmful drawback: no held-out dataset would remain to early-stop on. In a setting where overfitting can easily occur, we found the risk to vastly outweigh the rewards. Instead, we decided to randomly partition the validation data into $k = 10$ equally-sized folds, and perform a cross-validation-style setup: we train k different models, each one observing the training set *and* $k - 1$ validation folds as its training data, validating and early stopping on the held-out fold. Each model holds out a different fold, allowing us to get an overall validation estimate over the entire dataset by combining their respective predictions.

While this approach may not correspond to the intended dataset splits, we have verified that the scores on individual held-out folds match the patterns observed on models that did not observe any validation data. This gave us further reassurance that no unintended strong overfitting had happened as a result.

Another useful outcome of our k-fold approach is that it allowed us a very natural way to perform ensembling as well: simply aggregating all of the k models' predictions would give us a certain mixture of experts, as each of the k models had been trained on a slightly modified training set. Our final ensembled models employ exactly this strategy, with the inclusion of two seeds per fold. This brings our overall number of ensembled models to 20, and these ensembles correspond to our final submission on PCQM4M.

6 Results and Discussion

Our final ensembled model achieved a *validation MAE of 0.110* on PCQM4M. Translated on the LSC test set, we recover **0.1205 test MAE** on PCQM4M. We incur a minimal amount of distribution shift, which is a testament to our principled ensembling and post-processing strategies, in spite of training on validation.

Our entry has been designated as an *awardee* (ranked **in the top-3**) on PCQM4M, solidifying the impact that very deep expressive graph neural networks can have on large scale datasets of scientific relevance. Further, we demonstrate how several recently proposed auxiliary objectives for GNN training, such as Noisy Nodes [Godwin et al., 2021] can be highly impactful at the right dataset scales. We hope that our work serves towards resolving several open disputes in the community, such as the utility of very deep GNNs, and the influence of self-supervision in this setting.

We would like to note that this is a *draft technical report* submitted to the OGB-LSC organisers shortly after the competition ended. We intend to release a substantially more detailed write-up in terms of experimental results (especially ablation studies demonstrating the influence of our various design choices) in the near-term.

In many ways, the OGB has been to graph representation learning what ImageNet has been to computer vision. We hope that OGB-LSC is only the first in a series of events designed to drive research on GNN architectures forward, and sincerely thank the OGB team for all their hard work and effort in making a contest of this scale possible and accessible.

References

Victor Bapst, Thomas Keck, A Grabska-Barwińska, Craig Donner, Ekin Dogus Cubuk, Samuel S Schoenholz, Annette Obika, Alexander WR Nelson, Trevor Back, Demis Hassabis, et al. Unveiling the predictive power of static structure in glassy systems. *Nature Physics*, 16(4):448–454, 2020.

Peter W Battaglia, Jessica B Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, et al. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*, 2018.

Marc Brockschmidt. Gnn-film: Graph neural networks with feature-wise linear modulation. In *International Conference on Machine Learning*, pages 1144–1152. PMLR, 2020.

Michael M Bronstein, Joan Bruna, Taco Cohen, and Petar Veličković. Geometric deep learning: Grids, groups, graphs, geodesics, and gauges. *arXiv preprint arXiv:2104.13478*, 2021.

Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020.

Taco S Cohen, Mario Geiger, Jonas Köhler, and Max Welling. Spherical cnns. *arXiv preprint arXiv:1801.10130*, 2018.

Pim de Haan, Maurice Weiler, Taco Cohen, and Max Welling. Gauge equivariant mesh cnns: Anisotropic convolutions on geometric graphs. *arXiv preprint arXiv:2003.05425*, 2020.

Vijay Prakash Dwivedi, Chaitanya K Joshi, Thomas Laurent, Yoshua Bengio, and Xavier Bresson. Benchmarking graph neural networks. *arXiv preprint arXiv:2003.00982*, 2020.

Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *International Conference on Machine Learning*, pages 1263–1272. PMLR, 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`.

Jonathan Godwin, Michael Schaarschmidt, Alexander L Gaunt, Alvaro Sanchez-Gonzalez, Petar Veličković, Yulia Rubanova, James Kirkpatrick, and Peter Battaglia. Very deep graph neural networks via noise regularisation. *Preprint*, 2021.

Thomas A Halgren. Merck molecular force field. i. basis, form, scope, parameterization, and performance of mmff94. *Journal of computational chemistry*, 17(5-6):490–519, 1996.

William L Hamilton. Graph representation learning. *Synthesis Lectures on Artifical Intelligence and Machine Learning*, 14(3):1–159, 2020.

Jessica B Hamrick, Kelsey R Allen, Victor Bapst, Tina Zhu, Kevin R McKee, Joshua B Tenenbaum, and Peter W Battaglia. Relational inductive bias for physical construction in humans and machines. *arXiv preprint arXiv:1806.01203*, 2018.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

Weihua Hu, Bowen Liu, Joseph Gomes, Marinka Zitnik, Percy Liang, Vijay Pande, and Jure Leskovec. Strategies for pre-training graph neural networks. *arXiv preprint arXiv:1905.12265*, 2019.

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 preprint arXiv:2005.00687*, 2020.

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*, 2021.

Qian Huang, Horace He, Abhay Singh, Ser-Nam Lim, and Austin R Benson. Combining label propagation and simple models out-performs graph neural networks. *arXiv preprint arXiv:2010.13993*, 2020.

Chaitanya Joshi. Transformers are graph neural networks. *The Gradient*, 2020.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25:1097–1105, 2012.

Greg Landrum. Rdkit: A software suite for cheminformatics, computational chemistry, and predictive modeling, 2013.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Jonathan Masci, Davide Boscaini, Michael Bronstein, and Pierre Vandergheynst. Geodesic convolutional neural networks on riemannian manifolds. In *Proceedings of the IEEE international conference on computer vision workshops*, pages 37–45, 2015.

Azalia Mirhoseini, Anna Goldie, Mustafa Yazgan, Joe Jiang, Ebrahim Songhori, Shen Wang, Young-Joon Lee, Eric Johnson, Omkar Pathak, Sungmin Bae, et al. Chip placement with deep reinforcement learning. *arXiv preprint arXiv:2004.10746*, 2020.

Christopher Morris, Nils M Kriege, Franka Bause, Kristian Kersting, Petra Mutzel, and Marion Neumann. Tudataset: A collection of benchmark datasets for learning with graphs. *arXiv preprint arXiv:2007.08663*, 2020.

Maho Nakata and Tomomi Shimazaki. Pubchemqc project: a large-scale first-principles electronic structure database for data-driven chemistry. *Journal of chemical information and modeling*, 57(6):1300–1308, 2017.

Tobias Pfaff, Meire Fortunato, Alvaro Sanchez-Gonzalez, and Peter W Battaglia. Learning mesh-based simulation with graph networks. *arXiv preprint arXiv:2010.03409*, 2020.

Anthony K Rappé, Carla J Casewit, KS Colwell, William A Goddard III, and W Mason Skiff. Uff, a full periodic table force field for molecular mechanics and molecular dynamics simulations. *Journal of the American chemical society*, 114(25):10024–10035, 1992.

Yu Rong, Wenbing Huang, Tingyang Xu, and Junzhou Huang. Dropedge: Towards deep graph convolutional networks on node classification. *arXiv preprint arXiv:1907.10903*, 2019.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015.

Alvaro Sanchez-Gonzalez, Jonathan Godwin, Tobias Pfaff, Rex Ying, Jure Leskovec, and Peter Battaglia. Learning to simulate complex physics with graph networks. In *International Conference on Machine Learning*, pages 8459–8468. PMLR, 2020.

Victor Garcia Satorras, Emiel Hoogeboom, and Max Welling. E (n) equivariant graph neural networks. *arXiv preprint arXiv:2102.09844*, 2021.

Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. Collective classification in network data. *AI magazine*, 29(3):93–93, 2008.

Oleksandr Shchur, Maximilian Mumme, Aleksandar Bojchevski, and Stephan Günnemann. Pitfalls of graph neural network evaluation. *arXiv preprint arXiv:1811.05868*, 2018.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

Jonathan M Stokes, Kevin Yang, Kyle Swanson, Wengong Jin, Andres Cubillos-Ruiz, Nina M Donghia, Craig R MacNair, Shawn French, Lindsey A Carfrae, Zohar Bloom-Ackermann, et al. A deep learning approach to antibiotic discovery. *Cell*, 180(4):688–702, 2020.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *arXiv preprint arXiv:1706.03762*, 2017.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

Petar Veličković, William Fedus, William L Hamilton, Pietro Liò, Yoshua Bengio, and R Devon Hjelm. Deep graph infomax. *arXiv preprint arXiv:1809.10341*, 2018.

Petar Veličković, Rex Ying, Matilde Padovano, Raia Hadsell, and Charles Blundell. Neural execution of graph algorithms. *arXiv preprint arXiv:1910.10593*, 2019.

Shuzhe Wang, Jagna Witek, Gregory A Landrum, and Sereina Riniker. Improving conformer generation for small rings and macrocycles based on distance geometry and experimental torsional-angle preferences. *Journal of chemical information and modeling*, 60(4):2044–2058, 2020.

Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. Simplifying graph convolutional networks. In *International conference on machine learning*, pages 6861–6871. PMLR, 2019.

Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L Hamilton, and Jure Leskovec. Graph convolutional neural networks for web-scale recommender systems. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 974–983, 2018.