Supporting Information: Connectivity Optimized Nested Line Graph Networks for Crystal Structures

Robin Ruff
Institute of Theoretical Informatics
Karlsruhe Institute of Technology
Engler-Bunte-Ring 8,
76131 Karlsruhe, Germany

Patrick Reiser
Institute of Theoretical Informatics
Karlsruhe Institute of Technology
Engler-Bunte-Ring 8,
76131 Karlsruhe, Germany

Jan Stühmer
Institute for Anthropomatics and Robotics
Karlsruhe Institute of Technology
Engler-Bunte-Ring 8,
76131 Karlsruhe, Germany

Heidelberg Institute for Theoretical Studies
Schloß-Wolfsbrunnenweg 35,
69118 Heidelberg, Germany

Pascal Friederich
Institute of Theoretical Informatics
Karlsruhe Institute of Technology
Engler-Bunte-Ring 8,
76131 Karlsruhe, Germany

Institute of Nanotechnology
Karlsruhe Institute of Technology
Hermann-von-Helmholtz-Platz 1,
76344 Eggenstein-Leopoldshafen, Germany
pascal.friederich@kit.edu

37th Conference on Neural Information Processing Systems (NeurIPS 2023).

Supporting Information

1 Appendix

1.1 Crystal Preprocessing: Edge Selection

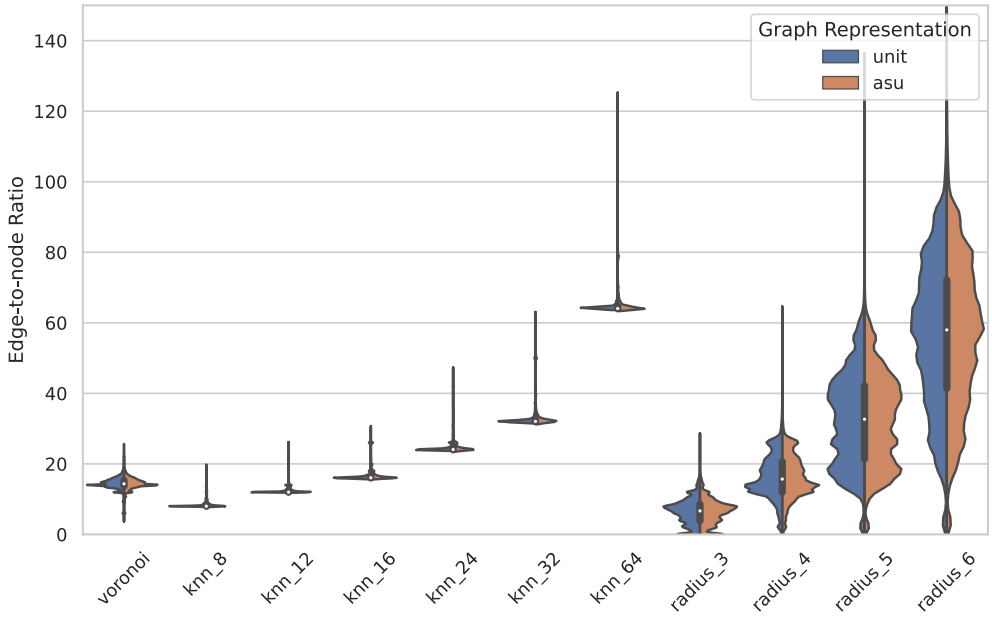


Figure 1: Distributions of edge-to-node ratios per graph (average degree) in the `mp_e_form` dataset for different preprocessing methods. Due to the symmetrical and regular structure of crystals, it often happens that there are many neighbors with equal distances. Therefore our kNN-based edge selection implementation, also allows to toggle whether edges with a ϵ-distance difference as the k-th nearest neighbor are also added as connections. We included an $\epsilon = 10^{-9}$ Å threshold to make up for numerical inaccuracies.

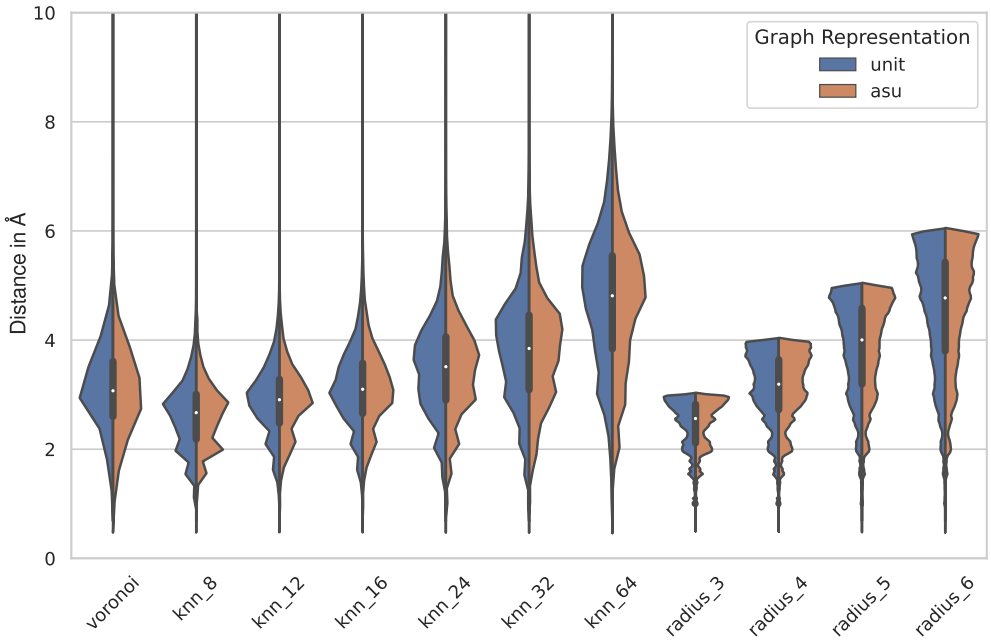


Figure 2: Distribution of edge distances in the `mp_e_form` dataset for different edge selection methods.

1.2 Crystal Preprocessing: Symmetries

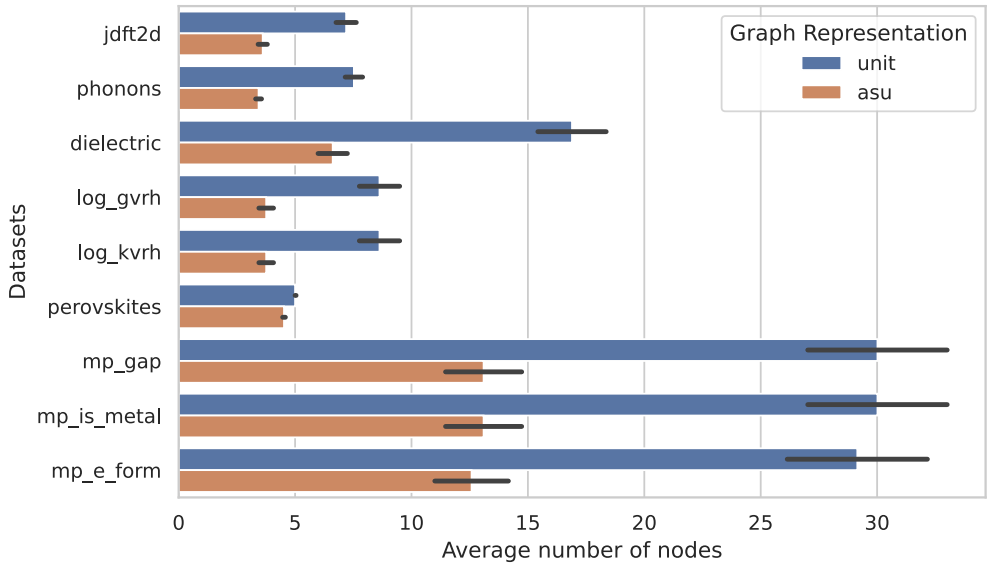


Figure 3: Average number of nodes in each dataset for unit cell graphs (unit) and asymmetric unit graphs (asu).

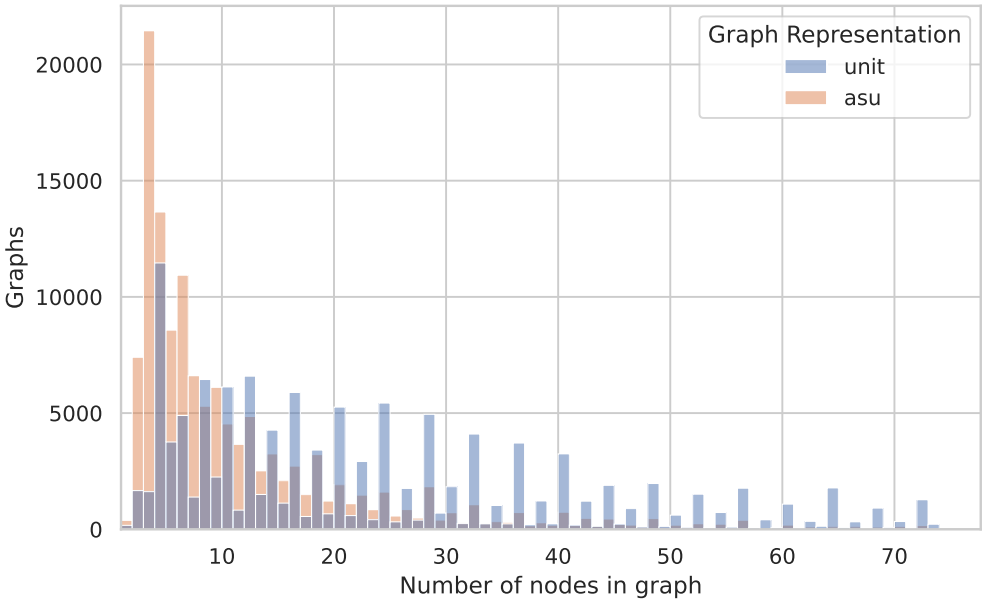


Figure 4: Histogram of number of nodes in `mp_e_form` dataset for unit cell graphs (unit) and asymmetric unit graphs (asu). The distribution for asymmetric unit graphs is skewed towards lower number of nodes. Clearly pronounced periodic peaks at even numbers for the unit cell graph distribution can be explained with symmetric atom pairs.

1.3 Nested Line Graph Network Algorithm

1.4 Line Graph Construction

Mathematically a line graph for a directed graph is defined as follows: An edge exists in $L(G)$ for each corresponding edge pair (e_{ij}, e_{jk}), which forms a path of length two in G (Figure 5a). This definition coincides with the angles $\measuredangle e_{ij}e_{jk}$ as used in in DimeNet [2], GemNet [3] and ALIGNN [4]. In this work, we propose a deviation from the original line graph definition and use angles $\measuredangle e_{ij}e_{kj}$

(a) Original line graph. (b) Our line graph variant.

Figure 5: Line graph variants.

between edges that have the same destination node, instead of edges that form a path of length two (see Figure 5b). The intuition behind this deviation is based on the interpretation of edge messages in the GNN as force contributions that act on an atom from different directions [5].

Note that we will use the term line graph very loosely in this work. Even though the variant is strictly speaking not a line graph we will still refer to it as such. In general, we will use the term line graph for every graph $L(G)$ that has a bijection between edges in G and nodes in $L(G)$ and a deterministic method for constructing edges in $L(G)$ based on the topology of G.

This generalized definition would also allow for incorporating dihedral angles by adding edges $\measuredangle e_{ij}e_{kl}$ to the line graph for every path (e_{ij}, e_{jk}, e_{kl}) of length three in the graph G [6–8]. Another

Function NGN-Block($\mathbf{X}_E, \mathbf{X}_V, \mathbf{x}_G, G; \theta$):

```
// X_E = {x_{e_1}, ..., x_{e_m}}
// X_V = {x_{v_1}, ..., x_{v_n}}
for e_ij ∈ E do
    x'_{e_ij} ← φ̂_E(x_{e_ij}, x_{v_i}, x_{v_j}, x_G)                    // Edge update
end
X_∠ ← get_line_graph_edge_features(X_E, X_V, x_G, G)                  // i.e. angles between edges
for t ← 1 to T^{L(G)} do
    _, X'_E, _, _ ← GN-Block(X_∠, X'_E, x_G, L(G); θ^t)               // Treat X'_E as node features of L(G)
end
for v_k ∈ V do
    x̂_{v_k} ← ρ_{E→V}({x'_{e_ij} | e_ij ∈ E ∧ j = k})                 // Local edge aggregation
    x'_{v_k} ← φ_V(x_{v_k}, x̂_{v_k}, x_G)                             // Node update
end
x̂_G ← ρ_{V→G}({x'_v | v ∈ V})                                         // Node aggregation
x̃_G ← ρ_{E→G}({x'_e | e ∈ E})                                         // Global edge aggregation
x'_G ← φ_G(x_G, x̂_G, x̃_G)                                             // Global update
// X'_E = {x'_{e_1}, ..., x'_{e_m}}
// X'_V = {x'_{v_1}, ..., x'_{v_n}}
return X'_E, X'_V, x'_G, G

// Iterate over sequentially composed NGN blocks
for t ← 1 to T do
    X_E, X_V, x_G, G ← NGN-Block(X_E, X_V, x_G, G; θ_t)
end
```

Algorithm 1: Graph network algorithm adapted from [1] with additional blue parts that indicate the modifications for the nested line graph networks. GN blocks are parameterized by parameters θ and receive edge features $\mathbf{X}_E$, node features $\mathbf{X}_V$, graph level features $\mathbf{x}_G$ as well as the graph topology G as input.

way to incorporate information of dihedral edges into the model would be to construct the second-order line graph $L(L(G))$ and add another layer of nesting to the model architecture.

1.5 Graph Network Exploration

Graph Network Exploration We start with a simple non-nested architecture shown in Figure 6, which consists of sequentially connected GN blocks, which can be divided into three phases.

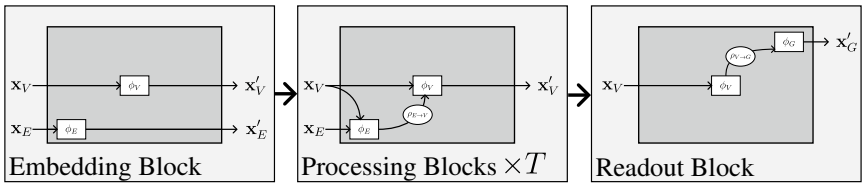


Figure 6: Basic graph network structure.

Moreover, we optimize the hyperparameter space on one dataset only, namely the `log_gvrh` MatBench dataset, and tested the model on all other datasets of MatBench without changing hyperparameters to verify that the found architecture is indeed a suitable candidate for multiple diverse tasks on crystal graphs.

The first block embeds atom/node features and bond/edge features independent from one another. In the second phase, we sequentially connect T processing blocks with identical architecture, but

independent learnable parameters. To restrict the search space we do not include global graph features and do not allow edge updates between blocks. This resembles a conventional message-passing architecture. In the third phase, we have a single readout block, which aggregates node features into graph-level features to make the crystal property prediction.

Figure 6 only specifies the high-level architecture of the GNN and not the concrete instantiations of update functions ϕ_E, ϕ_V, ϕ_G and aggregation functions $\rho_{E\to V}, \rho_{V\to G}$. To narrow down suitable concrete implementations, we conducted a two-part hyperparameter search on the `log_gvrh` MatBench dataset. First, we searched for categorical hyperparameters, in a greedy stepwise search. In the second step, we used the Optuna hyperparameter optimization framework [9] and its implementation of the Tree-structured Parzen Estimator (TPE) to find ordinal hyperparameters.[1]

We assume independence between categorical hyperparameters and optimize for each parameter individually, to keep runtimes within limits. We initially instantiated all update functions ϕ_E, ϕ_V, ϕ_G with MLPs of depth 3. Extending ϕ_V with residual $\phi_V(\mathbf{x}_v, \hat{\mathbf{x}}_v, \mathbf{x}_G) = \mathbf{x}_v + \text{MLP}_V(\hat{\mathbf{x}}_v)$ and gated node updates $\phi_V(\mathbf{x}_v, \hat{\mathbf{x}}_v, \mathbf{x}_G) = \text{GRU}(\mathbf{x}_v, \text{MLP}_V(\hat{\mathbf{x}}_v))$ both improve performance. For aggregation function ρ_V, ρ_G we tried mean, sum, and attention-based functions. Batch and graph normalization [10] increased training time significantly without considerable benefits for predictive performance. Prediction accuracy further increases when including atom features (atomic mass, radius, electronegativity, ionization, and oxidation states) in addition to the atomic number. The influence of the choice of edge selection methods shown in Figure 5 in the main article is discussed in detail in Section 4.2 in the main article.

For the kNN-based edge selection preprocessing with $k = 24$, we searched for ordinal hyperparameters with a hyperparameter optimization.

Embedding Block

ϕ_E: Gauss basis expansion of edge distances with 32 Gaussians evenly spread on the $[0, 8]$ Å interval. As only distance information is used, this embedding is E(3)-invariant. The initial representation is projected into a 128-dimensional embedding space with a single perceptron layer.

ϕ_V: Embedding of atom features (atomic number, atomic mass, atomic radius, electronegativity, ionization states, and oxidation states) into a 128-dimensional space.

Processing Blocks Five processing blocks are concatenated with identical configurations ($T = 5$).

$\phi_E(\mathbf{x}_{e_{ij}}, \mathbf{x}_{v_i}, \mathbf{x}_{v_j}) = \text{MLP}_E(\mathbf{x}_{e_{ij}} || \mathbf{x}_{v_i} || \mathbf{x}_{v_j}) = \mathbf{x}'_{e_{ij}}$ The edge update function, which constructs the message between two nodes, is a five-layer MLP and takes the concatenation of edge, receiver, and sender node features.

$\rho_{E\to V}(\{\mathbf{x}'_{e_{ij}} | j = k\}) = \underset{j=k}{\text{sum}}(\mathbf{x}'_{e_{ij}}) = \hat{\mathbf{x}}_{v_k}$ For each node the incoming messages are sum-aggregated.

$\phi_V(\mathbf{x}_v, \hat{\mathbf{x}}_v) = \mathbf{x}_v + \text{MLP}_V(\hat{\mathbf{x}}_v)$ A residual node update function transforms the aggregated messages with a single-layer perceptron.

Readout Block

$\phi_V(\mathbf{x}_v) = \mathbf{x}_v$ The readout block does not update node features. Its node update function is the identity function.

$\rho_{V\to G}(\{\mathbf{x}'_v | v \in V\}) = \underset{v\in V}{\text{mean}}(\mathbf{x}'_v) = \hat{\mathbf{x}}_G$ The mean of the node features is aggregated to compute the graph-level prediction.

$\phi_G(\hat{\mathbf{x}}_G) = \text{MLP}_G(\hat{\mathbf{x}}_G) = \mathbf{x}'_G$ A single-layer MLP with a linear activation function creates the final prediction for the crystal property.

Below we give a complete description of our model, which we name coGN, after categorical and ordinal hyperparameter optimization:

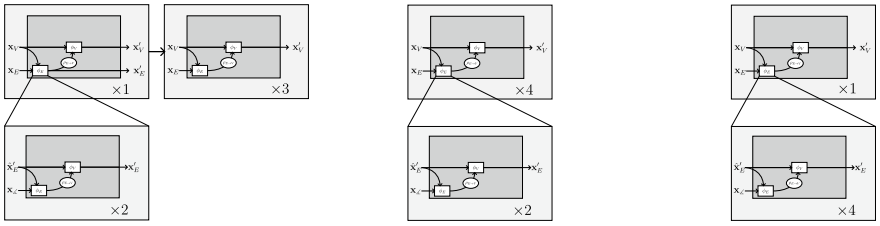
We used the same dimensionality (128) for all hidden representations of edges, nodes, and graphs. Unless mentioned otherwise we use the commonly used swish activation function in MLPs. The GNN is trained with an Adam optimizer with a linear learning rate scheduler for 800 epochs.

[1] Ordinal hyperparameters: depth of GNN (T), depth of MLPs $\text{MLP}_E, \text{MLP}_V, \text{MLP}_G$, dimensionality of features

Overall, coGN is comparably simple, as it only contains MLPs as update functions, mean or sum aggregation functions and no sophisticated message-passing scheme, such as edge-gated or attention-based message passing used in CGCNN, ALIGNN or GeoCGNN. In this regard, coGN also does not incorporate any domain-specific design decisions, which are not justified by the hyperparameter optimization.

When compared with previous models, we find that along with a connectivity optimization, discussed in section 4.2 in the main article, a much deeper edge update network ϕ_E, in our case five layers, yields better results. We attribute this observation to the increased complexity of edge information in a periodic multi-graph that in practice exhibits a large number of edges per node.

Nested Line Graph Network Exploration Based on the results of the first part of the hyperparameter search for parameters with categorical values, we augment the processing blocks with nested GN blocks shown in Figure 7.



(a) NLGN variant 1. (b) NLGN variant 2. (c) NLGN variant 3.

Figure 7: Nested line graph network variants.

The first variant (Figure 7a) has a single nested GN block at the beginning, which updates node and edge features, followed by non-nested GN blocks. The reasoning behind this architecture is that the Nested GN block might be able to encode geometrical constellations of edge angles into edge features. The second variant (Figure 7b) is a generalized version of ALIGNN-d [6] and DimeNet(++) [2, 7]. Each GN block on the graph level (G) has two GN blocks on the line graph level ($L(G)$). Only node features are updated between GN blocks. The third variant (Figure 7c) moves most of the computation to the line graph level ($L(G)$). It consists of only one GN block on the graph level with 4 consecutive nested GN blocks ($T^{L(G)} = 4$).

E(3)-invariant angle features between two edges are encoded with a 16-dimensional Gauss basis expansion on the $[0, \pi]$ rad interval and attached to line graph edges.

Table 1: Results on the `log_gvrh` dataset with NLGN variants.

Nested GN	Line Graph	MAE (`log_gvrh`)
Variant 1	$\measuredangle e_{ij}e_{jk}$	0.0809 ± 0.0022
	$\measuredangle e_{ij}e_{kj}$	0.0787 ± 0.0019
Variant 2	$\measuredangle e_{ij}e_{jk}$	0.0801 ± 0.0020
	$\measuredangle e_{ij}e_{kj}$	0.0783 ± 0.0033
Variant 3	$\measuredangle e_{ij}e_{jk}$	0.0805 ± 0.0015
	$\measuredangle e_{ij}e_{kj}$	0.0799 ± 0.0016

Training on preprocessed crystals with kNN edge selection and $k = 24$ and both line graph variants from Section 1.4, we obtain the results displayed in Table 1. The results were obtained before the final ordinal hyperparameter optimization, which explains the discrepancy with Table 2 in the main article. For comparison, the MAE for the corresponding non-nested GN is 0.0788.

Despite the theoretically greater expressiveness of NLGNs, we do not achieve substantially better prediction results. The proposed line graph variant with angles between edges with the same target node ($\measuredangle e_{ij}e_{kj}$) leads to a small but consistent improvement across all line graph variants.

We optimized the ordinal hyperparameters of the DimeNet-like architecture (Variant 2) with TPE and reached a MAE of 0.0705 on the `log_gvrh` MatBench dataset which yields better performance than the current leader on MatBench [4], even though the graph preprocessing, i.e. the connectivity was not optimized yet. Hyperparameter optimization of plain GNs yields a similar error of 0.0693, making them comparable to nested GNs.

1.6 Open Catalyst Project: OC22

Table 2: Test error for the *initial structure to relaxed energies* (IS2RES) task of the OC22 challenge [11] (status 2023-08-08). Note that Models trained additionally on OC20 [12] and other data sources or indirect predictions using relaxations yield lower errors and better performance. Baseline models are SchNet [13], DimeNet++ [14], PaiNN [15] and GemNet [8]. The best Direct OC22-only predictions is marked by underscores.

Model	MAE (ID)	MAE (OOD)	Average
EquiformerV2 (122M, λ_E=1, λ_F=1, OC22-only)	**1.0837**	1.4443	**1.264**
EquiformerV2 (122M, λ_E=4, λ_F=100, OC22-only)	1.1181	**1.4398**	1.279
GemNet-OC (Relaxation OC20-All+OC22)	1.2007	1.5339	1.3673
GemNet-OC (Relaxation OC22-only)	1.3294	1.5841	1.4567
GemNet-dT (Relaxation OC22-only)	1.8129	2.0439	1.9284
coGN (Direct OC22-only, r=5.0)	<u>1.6183</u>	<u>2.8058</u>	<u>2.2121</u>
coGN (Direct OC22-only, k=32)	1.6278	2.9706	2.2992
GemNet-dT (Direct OC22-only)	1.6771	3.0837	2.3804
PaiNN (Direct OC22-only)	1.716	3.6835	2.6997
DimeNet++ (Direct OC22-only)	1.96	3.5186	2.7393
SchNet (Direct OC22-only)	2.0012	4.8468	3.424

1.7 JARVIS-Tools

Table 3: Test error for multiple tasks of the JARVIS benchmark [16]. Values are copied from the JARVIS leaderboard (status 2023-08-08). Models for comparison are DimeNet++ (DN++) [14] , ALIGNN [4], CGCNN [17], MatMiner (MM) [18, 19] and CFID [20].

Task	coGN	coNGN	DN++	ALIGNN	CGCNN	MM	CFID
dft_3d_mepsz	24.1081	**22.842**	30.3644	23.7313	36.0538	24.6651	29.3445
dft_3d_exfoliation_energy	47.6979	46.272	46.1517	52.7033	52.7033	**40.887**	62.1169
dft_3d_shear_modulus_gv	8.6612	**8.4881**	26.0817	9.476	16.0459	10.5415	11.9164
dft_3d_spillage	0.3609	**0.3463**	0.4137	0.351	0.3965	0.3592	0.3867
dft_3d_optb88vdw_total_energy	**0.0262**	0.0273	0.051	0.0367	0.0815	0.0936	0.2436
dft_3d_mepsx	24.2289	**23.3801**	31.9568	24.0458	33.6597	25.2932	30.261
dft_3d_epsz	19.6192	**17.8104**	33.8379	19.5678	33.6597	25.2932	24.781
dft_3d_dfpt_piezo_max_dij	15.2235	**13.8868**	13.9889	20.5705	16.0135	21.5729	-
dft_3d_mepsy	24.1891	**23.3299**	31.0215	23.6482	32.4577	25.0706	30.0578
qe_tb_indir_gap	0.0474	-	-	0.1167	-	**0.0351**	-
dft_3d_kpoint_length_unit	9.5722	9.3459	11.8875	9.5146	13.2145	**9.047**	9.7085
dft_3d_n_powerfact	452.235	456.6118	568.8357	**442.2993**	-	469.6279	-
dft_3d_ph_heat_capacity	6.1125	7.8127	23.3618	9.6064	-	**5.2757**	-
dft_3d_formation_energy_peratom	**0.0271**	0.0291	0.0528	0.0331	0.0625	0.0734	0.1419
dft_3d_epsx	20.0004	**18.5738**	27.2511	20.3942	31.4744	21.2597	24.8408
dft_3d_optb88vdw_bandgap	**0.1219**	0.1267	0.2247	0.1423	0.1908	0.1873	0.299
qe_tb_energy_per_atom	**0.0636**	-	0.7515	-	-	1.5049	-
dft_3d_max_efg	20.4417	19.5495	26.9552	**19.1211**	24.6695	19.4382	-
dft_3d_epsy	24.1891	**23.3299**	31.0215	23.6482	32.4577	25.0706	30.0578
dft_3d_encut	133.8915	**129.8266**	164.315	133.7962	190.3857	138.2769	139.4357
dft_3d_n_Seebeck	**39.2692**	40.0977	54.2759	40.9214	49.3172	44.2229	-
dft_3d_ehull	**0.0466**	0.0485	0.3685	0.0763	0.173	0.0601	-
dft_3d_bulk_modulus_kv	8.992	**8.7022**	13.3743	10.3988	19.3028	12.7411	14.1999
dft_3d_avg_hole_mass	0.1372	0.1285	0.1709	**0.1239**	-	0.1529	-
dft_3d_avg_elec_mass	0.0917	0.0876	0.112	**0.0853**	-	0.107	-
dft_3d_mbj_bandgap	**0.264**	0.2719	0.4764	0.3104	0.4067	0.3392	0.5313
dft_3d_dfpt_piezo_max_dielectric	30.2923	**25.5553**	30.3358	28.1514	32.5589	36.6913	-
dft_3d_slme	4.4507	**4.4428**	5.6403	4.5207	5.6603	4.9255	6.2607
qe_tb_f_enp	**0.0956**	-	-	0.1016	-	0.3219	-
dft_3d_magmom_oszicar	0.2502	**0.2437**	0.3995	0.2574	0.3543	0.3645	0.4748
qe_tb_final_energy	**1.3185**	-	-	-	-	1.4714	-

References

[1] 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.

[2] Johannes Klicpera, Janek Groß, and Stephan Günnemann. Directional message passing for molecular graphs. *arXiv preprint arXiv:2003.03123*, 2020.

[3] Johannes Gasteiger, Florian Becker, and Stephan Günnemann. Gemnet: Universal directional graph neural networks for molecules. *Advances in Neural Information Processing Systems*, 34: 6790–6802, 2021.

[4] Kamal Choudhary and Brian DeCost. Atomistic line graph neural network for improved materials property predictions. *npj Computational Materials*, 7(1):1–8, 2021.

[5] Michaël Defferrard, Martino Milani, Frédérick Gusset, and Nathanaël Perraudin. Deepsphere: a graph-based spherical cnn. In *International Conference on Learning Representations*, 2020. URL `https://openreview.net/forum?id=B1e3OlStPB`.

[6] Tim Hsu, Nathan Keilbart, Stephen Weitzner, James Chapman, Penghao Xiao, Tuan Anh Pham, S Roger Qiu, Xiao Chen, and Brandon C Wood. Efficient, interpretable atomistic graph neural network representation for angle-dependent properties and its application to optical spectroscopy prediction. *arXiv preprint arXiv:2109.11576*, 2021.

[7] Johannes Klicpera, Shankari Giri, Johannes T Margraf, and Stephan Günnemann. Fast and uncertainty-aware directional message passing for non-equilibrium molecules. *arXiv preprint arXiv:2011.14115*, 2020.

[8] Johannes Klicpera, Florian Becker, and Stephan Günnemann. Gemnet: Universal directional graph neural networks for molecules. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, 2021. URL `https://openreview.net/forum?id=HS_sOaxS9K-`.

[9] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2019.

[10] Tianle Cai, Shengjie Luo, Keyulu Xu, Di He, Tie-yan Liu, and Liwei Wang. Graphnorm: A principled approach to accelerating graph neural network training. In *International Conference on Machine Learning*, pages 1204–1215. PMLR, 2021.

[11] Richard Tran, Janice Lan, Muhammed Shuaibi, Brandon Wood, Siddharth Goyal, Abhishek Das, Javier Heras-Domingo, Adeesh Kolluru, Ammar Rizvi, Nima Shoghi, Anuroop Sriram, Zachary Ulissi, and C. Lawrence Zitnick. The open catalyst 2022 (oc22) dataset and challenges for oxide electrocatalysis. *arXiv preprint arXiv:2206.08917*, 2022.

[12] 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*, 2021. doi:10.1021/acscatal.0c04525.

[13] Kristof Schütt, Pieter-Jan Kindermans, Huziel Enoc Sauceda Felix, Stefan Chmiela, Alexandre Tkatchenko, and Klaus-Robert Müller. Schnet: A continuous-filter convolutional neural network for modeling quantum interactions. *Advances in neural information processing systems*, 30, 2017.

[14] Johannes Klicpera, Janek Groß, and Stephan Günnemann. Directional message passing for molecular graphs. *CoRR*, abs/2003.03123, 2020. URL `https://arxiv.org/abs/2003.03123`.

[15] Kristof Schütt, Oliver Unke, and Michael Gastegger. Equivariant message passing for the prediction of tensorial properties and molecular spectra. In *International Conference on Machine Learning*, pages 9377–9388. PMLR, 2021.

[16] Kamal Choudhary, Kevin F. Garrity, Andrew C. E. Reid, Brian DeCost, Adam J. Biacchi, Angela R. Hight Walker, Zachary Trautt, Jason Hattrick-Simpers, A. Gilad Kusne, Andrea Centrone, Albert Davydov, Jie Jiang, Ruth Pachter, Gowoon Cheon, Evan Reed, Ankit Agrawal, Xiaofeng Qian, Vinit Sharma, Houlong Zhuang, Sergei V. Kalinin, Bobby G. Sumpter, Ghanshyam Pilania, Pinar Acar, Subhasish Mandal, Kristjan Haule, David Vanderbilt, Karin Rabe, and Francesca Tavazza. The joint automated repository for various integrated simulations (JARVIS) for data-driven materials design. 6(1):173, 2020. ISSN 2057-3960. doi:10.1038/s41524-020-00440-1. URL `https://doi.org/10.1038/s41524-020-00440-1`.

[17] Tian Xie and Jeffrey C Grossman. Crystal graph convolutional neural networks for an accurate and interpretable prediction of material properties. *Physical review letters*, 120(14):145301, 2018.

[18] Kangming Li, Brian DeCost, Kamal Choudhary, Michael Greenwood, and Jason Hattrick-Simpers. A critical examination of robustness and generalizability of machine learning prediction of materials properties. 9(1):55. ISSN 2057-3960. doi:10.1038/s41524-023-01012-9. URL `https://doi.org/10.1038/s41524-023-01012-9`.

[19] Logan Ward, Alexander Dunn, Alireza Faghaninia, Nils E.R. Zimmermann, Saurabh Bajaj, Qi Wang, Joseph Montoya, Jiming Chen, Kyle Bystrom, Maxwell Dylla, Kyle Chard, Mark Asta, Kristin A. Persson, G. Jeffrey Snyder, Ian Foster, and Anubhav Jain. Matminer: An open source toolkit for materials data mining. *Computational Materials Science*, 152:60–69, September 2018. ISSN 0927-0256. doi:10.1016/j.commatsci.2018.05.018.

[20] Kamal Choudhary, Brian DeCost, and Francesca Tavazza. Machine learning with force-field-inspired descriptors for materials: Fast screening and mapping energy landscape. *Phys. Rev. Mater.*, 2:083801, Aug 2018. doi:10.1103/PhysRevMaterials.2.083801. URL `https://link.aps.org/doi/10.1103/PhysRevMaterials.2.083801`.