Accelerating Molecular Graph Neural Networks via Knowledge Distillation

Filip Ekström Kelvinius[*]
Linköping University
filip.ekstrom@liu.se

Dimitar Georgiev[*]
Imperial College London
d.georgiev21@imperial.ac.uk

Artur Petrov Toshev[*]
Technical University of Munich
artur.toshev@tum.de

Johannes Gasteiger
Google Research
johannesg@google.com

Abstract

Recent advances in graph neural networks (GNNs) have enabled a more comprehensive representation of molecules and molecular systems, thereby enhancing the precision of molecular property prediction and molecular simulations. Nonetheless, as the field has been progressing to bigger and more complex architectures, state-of-the-art GNNs have become largely prohibitive for many large-scale applications. In this paper, we explore the utility of knowledge distillation (KD) for accelerating molecular GNNs. To this end, we devise KD strategies that facilitate the distillation of hidden representations in directional and equivariant GNNs, and evaluate their performance on the regression task of energy and force prediction. We validate our protocols across different teacher-student configurations and datasets, and demonstrate that they can consistently boost the predictive accuracy of student models without any modifications to their architecture. All in all, we manage to close the gap in predictive accuracy between teacher and student models by as much as 96.7% and 62.5% for energy and force prediction respectively, while fully preserving the inference throughput of the more lightweight models.

1 Introduction

In the last couple of years, the field of molecular simulations has undergone a rapid paradigm shift with the advent of new, powerful computational tools based on machine learning (ML) [1, 2]. At the forefront of this transformation have been recent advances in graph neural networks (GNNs), which have brought about architectures that more effectively capture geometric and structural information critical for the accurate representation of molecules and molecular systems [3–5]. Consequently, a multitude of GNNs have been developed, which now offer predictive performance approaching that of conventional gold-standard methods like density functional theory (DFT) at a fraction of the computational cost [6–10]. This has, in turn, significantly accelerated the modeling of molecular properties and the simulation of diverse molecular systems, bolstering new research developments in many scientific disciplines, including material sciences, drug discovery and catalysis.

Nonetheless, this progress - largely coinciding with the development of bigger and more complex models, has naturally come at the expense of increased complexity [8, 10–12]. This has gradually limited the utility of state-of-the-art GNNs for large-scale molecular simulation applications (e.g., molecular dynamics, high-throughput searches), where inference throughput (i.e., how many samples can be processed for a given time) is critical for making fast continual predictions about the evolution of a system. Hence, addressing the trade-off between accuracy and computational demand

[*]These authors contributed equally to this work. Order was determined by rolling a dice.

F. Ekström Kelvinius et al., Accelerating Molecular Graph Neural Networks via Knowledge Distillation (Extended Abstract). Presented at the Second Learning on Graphs Conference (LoG 2023), Virtual Event, November 27–30, 2023.

remains essential for creating more affordable tools for molecular simulations and expanding the transformational impact of GNN models in the area.

Motivated by that, in this work, we investigate the potential of knowledge distillation (KD) in enhancing the performance and scalability of state-of-the-art GNNs for molecular simulations.

2 Methods

Setup. We consider molecular systems at an atomic level, i.e., N atoms represented by their atomic number $\boldsymbol{z} = \{z_1, ..., z_N\} \in \mathbb{Z}^N$ and positions $\boldsymbol{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\} \in \mathbb{R}^{N\times 3}$. Given a system, we want a model that can predict in a multi-output fashion the energy $E \in \mathbb{R}$ of the system, and the forces $\boldsymbol{F} \in \mathbb{R}^{N\times 3}$ acting on each atom (see Appendix A.1 for more information).

In the context of this prediction task, we train molecular GNNs by enforcing a loss that combines both the energy and force prediction error as follows:

$$\mathcal{L}_0 = \alpha_{\mathrm{E}}\mathcal{L}_{\mathrm{E}}(\hat{E}, E) + \alpha_{\mathrm{F}}\mathcal{L}_{\mathrm{F}}(\hat{\boldsymbol{F}}, \boldsymbol{F}), \tag{1}$$

where E and $\boldsymbol{F}$ are the ground-truth energy and forces, $\hat{E}$ and $\hat{\boldsymbol{F}}$ are the predictions of the model of interest, and $\mathcal{L}_{\mathrm{E}}$ and $\mathcal{L}_{\mathrm{F}}$ are some loss functions weighted by $\alpha_{\mathrm{E}}, \alpha_{\mathrm{F}} \in \mathbb{R}$. To perform knowledge distillation, we augment this training process by defining an auxiliary KD loss term $\mathcal{L}_{\mathrm{KD}}$, which is added to $\mathcal{L}_0$ (with a factor $\lambda \in \mathbb{R}$) to derive a new training loss function $\mathcal{L}$ of the form:

$$\mathcal{L} = \mathcal{L}_0 + \lambda\mathcal{L}_{\mathrm{KD}}. \tag{2}$$

In this work, we focus on feature-based KD - an extension of vanilla output-based KD, concerned with the distillation of knowledge across the intermediate layers of models [13]. This allows us to overcome the limitations of vanilla KD in our context (see Appendices A.2 and A.3) and train more lightweight student models to mimic features that are easier to assimilate compared to the final output directly [14]. We perform knowledge distillation of intermediate representations by devising a loss on selected hidden features $H_{\mathrm{s}} \in U_{\mathrm{s}}$ and $H_{\mathrm{t}} \in U_{\mathrm{t}}$ in the student and teacher models respectively, which takes the form:

$$\mathcal{L}_{\mathrm{KD}} = \mathcal{L}_{\mathrm{feat}}(\mathcal{M}_{\mathrm{s}}(H_{\mathrm{s}}), \mathcal{M}_{\mathrm{t}}(H_{\mathrm{t}})), \tag{3}$$

where $\mathcal{M}_{\mathrm{s}} : U_{\mathrm{s}} \mapsto U$ and $\mathcal{M}_{\mathrm{t}} : U_{\mathrm{t}} \mapsto U$ are transformations that map the hidden features to a common feature space U, and $\mathcal{L}_{\mathrm{feat}} : U \times U \mapsto \mathbb{R}^+$ is some loss of choice. Possible options for the transformations $\mathcal{M}_{\mathrm{s}}, \mathcal{M}_{\mathrm{t}}$ include the identity transformation, linear projections and multilayer perceptron (MLP) projection heads; whereas for the distillation loss $\mathcal{L}_{\mathrm{feat}}$, typical functions are mean squared error (MSE) and mean absolute error (MAE).

Defining feature distillation strategies for molecular GNNs. Unlike standard GNNs that often only consider scalar node features, molecular GNNs can contain diverse features (scalars, vectors and/or equivariant higher-order tensors based on spherical harmonics) organized across nodes and edges within a complex molecular graph. These are continually evolved by model-specific operators to infer molecular properties, such as energy and forces, in a multi-output prediction fashion. Therefore, features often represent different physical, geometric and/or topological information relevant to specific parts of the output. This significantly complicates the design of an effective KD strategy, especially when models differ architecturally, as one needs to extract and align representations corresponding to comparable features in both models.

Here, we set out to devise KD strategies that are representative and effective across various molecular GNNs. Hence, we consider GNNs that have diverse architectures and performance profiles, namely GemNet-OC [15], PaiNN [16], and SchNet [17] (see Appendices E and F). Unlike typical student-teacher configurations, these models are characterized by distinct types of features, including scalar node- and edge-features, and equivariant geometrical vectors (see Appendix B for more information). Here, we leverage these model dissimilarities and devise three distinct KD strategies:

- node-to-node (n2n): As all three models considered in this study contain scalar node features H_{node}, we can distill knowledge in between these directly by defining a loss $\mathcal{L}_{KD}$, such that

$$\mathcal{L}_{KD} = \mathcal{L}_{\mathrm{feat}}(\mathcal{M}_s(H_{\mathrm{node,s}}), \mathcal{M}_t(H_{\mathrm{node,t}})). \tag{4}$$

Note this is a general approach that utilizes node features only, making it applicable to standard GNNs. Here, we want to force the student to mimic the representations of the teacher for each

node (i.e. atom) independently, so we use a loss that directly penalizes the distance between the features in the two models, such as MSE (similar to the original formulation of feature-based KD in Romero *et al.* [13]). Other recently proposed losses $\mathcal{L}_{\text{feat}}$ for the distillation of node features in standard GNNs specifically include approaches based on contrastive learning [18–21] and adversarial training [22]. We do not focus on such methods as much since they are better suited for (node) classification tasks (e.g. contrasting different classes of nodes), and not for molecule-level predictions.

To take advantage of other types of features relevant to molecular GNNs, we further devise two additional protocols below.

- *edge-to-node (e2n):* The GemNet-OC model heavily relies on its edge features, which are a key component in the directional message passing defined in the architecture and can be useful as a KD resource. However, the other models considered here do not have similar edge features to distill to. To accommodate that, we propose a KD strategy where we transfer information from GemNet-OC's edge features $H_{\text{edge},(i,j)}$ by first aggregating them as follows:

$$H_{\text{edge2node},i} = \sum_{j \in \mathcal{N}(i)} H_{\text{edge},(i,j)}, \tag{5}$$

where i is the node index. The resulting vector $H_{\text{edge2node},i}$ is a scalar, node-level feature, and we can, therefore, use it to transfer knowledge to the student node features $H_{\text{node,s}}$ as in Eq. 4.

- *vector-to-vector (v2v):* Similarly, the PaiNN model defines custom vectorial node features, which differ substantially from the scalar (node and edge) features available in the other models. These are not scalar and invariant to rigid transformations of the atoms, but geometrical vectors that are equivariant with respect to rotations. As these carry important information about a given system, we also want to define a procedure to distill these. When we perform KD between two PaiNN models, we can directly distill information between these vectorial features just as in Eq. 4. However, when distilling knowledge into PaiNN from a teacher without such vectorial features, e.g., GemNet-OC, we do the following feature transformation. We transfer knowledge between (invariant) scalar *edge* features and (equivariant) vectorial node features by noting that scalar edge features sit on an equivariant 3D grid since they are associated with an edge between two atoms in 3D space. Hence, we can aggregate the edge features $\{H_{\text{edge},(i,j)}\}_{j \in \mathcal{N}}$ corresponding to a given node i into node-level equivariant vectorial features $H_{\text{vec},i}$ by considering the unit vector $\boldsymbol{u}_{ij} = \frac{1}{|\boldsymbol{x}_j - \boldsymbol{x}_i|}(\boldsymbol{x}_j - \boldsymbol{x}_i)$ that defines the direction of the edge (i, j), such that

$$H^{(k)}_{\text{vec},i} = \sum_{j \in \mathcal{N}(i)} \boldsymbol{u}_{i,j} H^{(k)}_{\text{edge},(i,j)}, \tag{6}$$

with the superscript k indicating the channel. Note $H^{(k)}_{\text{vec},i}$ is equivariant to rotations, as the vector $\boldsymbol{u}$ is equivariant to rotations, and $H^{(k)}_{\text{edge},(i,j)}$ is a scalar, not influencing its direction.

Additional KD strategies. We further evaluate two additional KD approaches inspired by the vanilla logit-based KD used in classification, which we augment to make suitable for regression tasks (see Appendix C).

3 Experimental results

To evaluate our proposed methods, we perform experiments using different teacher-student configurations on the OC20-2M [23] and COLL [24] datasets, with models as implemented in the OC20 codebase[2]. Experimental details can be found in Appendix D. In addition to regular PaiNN (referred to as PaiNN-big here), we also use a smaller version (PaiNN-small) where we reduce the number of hidden layers and their dimensionality.

The results of our experiments on OC20 are summarized in Table 1 (see Appendix G for results on COLL), presenting a comparative analysis of the predictive performance of different student models with and without the implementation of knowledge distillation across four relevant metrics: energy MAE - the MAE between ground truth and predicted energies; force MAE; force cos - the cosine similarity between ground truth and predicted forces; and energy and forces within threshold (EFwT)

[2] https://github.com/Open-Catalyst-Project/ocp

Table 1: Evaluation of the performance of our KD strategies across teacher-student architectures on the OC20 S2EF task. All models are trained on the OC20-2M dataset. Numbers in brackets represent the proportion of the gap between the student and the teacher that has been closed by the respective KD strategy (in %). Best results are given in **bold**. Values represent the average across the 4 validation datasets available in OC20.

		OC20 S2EF Validation			
	Model	Energy MAE meV ↓	Force MAE meV/Å ↓	Force cos ↑	EFwT % ↑
same	Student (PaiNN-small)	489	47.1	0.345	0.085
	Teacher (PaiNN-big)	440	45.3	0.376	0.139
	Vanilla KD (1)	515(-52.4%)	48.5(-81.0%)	0.269(-237%)	0.07(-28%)
	Vanilla KD (2)	476(27.2%)	50.8(-215%)	0.307(-117%)	0.068(-32.6%)
	n2n	457(**64.8%**)	46.7(**20.5%**)	0.348(**9.3%**)	0.085(**0.5%**)
	v2v	459(60.8%)	47.2(-9.1%)	0.347(6.8%)	0.079(-11.9%)
similar	Student (SchNet)	1308	65.1	0.204	0
	Teacher (PaiNN-big)	440	45.3	0.376	0.139
	Vanilla KD (1)	1214(**10.8%**)	64.6(2.3%)	0.230(**15.2%**)	0.003(**1.8%**)
	Vanilla KD (2)	1216(10.5%)	64.6(**2.5%**)	0.229(14.5%)	0(0%)
	n2n	1251(6.6%)	65.2(-0.5%)	0.223(11.1%)	0(0%)
different	Student (PaiNN-big)	440	45.3	0.376	0.139
	Teacher (GemNet-OC)	286	25.7	0.598	1.063
	Vanilla (1)	440(0.0%)	43.9(7.1%)	0.378(0.8%)	0.14(0.4%)
	Vanilla (2)	419(13.6%)	114.8(-353%)	0.324(-23.8%)	0.127(-1.3%)
	n2n	**346 (60.8%)**	42.8(12.8%)	0.393(7.4%)	**0.262 (13.4%)**
	e2n	418(14.2%)	**41.3 (20.5%)**	**0.405 (12.8%)**	0.207(7.4%)
	v2v	437(1.8%)	42.9(17.1%)	0.397(9.4%)	0.124(-1.6%)

- the percentage of systems whose predicted energies and forces are within a specified threshold from the ground truth [23]. We observe that by using KD, we can significantly boost the performance of the student models, allowing us to substantially close the gap in predictive accuracy between teacher and student models - achieving results as high as 64.8% and 20.5% for energy and force predictions respectively on OC20, and 96.7% and 62.5% respectively on COLL. More importantly, our experiments demonstrate that the proposed procedures are robust and consistent, as we achieve improvements in accuracy across all teacher-student configurations on both datasets.

Notice that performance improvements are generally higher in energy predictions than force predictions. One possible explanation for this phenomenon could be attributed to the nature of the supervised task. In particular, there are substantially more force labels (i.e., one 3D vector per atom, which could be hundreds per sample) than energy labels (i.e., one per sample). Consequently, we hypothesize it is easier for models to learn to make accurate force predictions, and, therefore, there is more room for improvement in the energy predictions, which we can target with KD.

4 Conclusion

In this paper, we investigate the utility of knowledge distillation as a means of distilling larger, more computationally expensive GNNs for molecules into smaller, more efficient models. To this end, we propose three distinct feature-based KD strategies that allow the distillation of intermediate representations across diverse molecular GNN models. Our experiments demonstrate that knowledge distillation can consistently enhance the performance of student models across teacher-student configurations and datasets, confirming its effectiveness and robustness in the context of molecular GNNs. We reiterate that with our approach, no modifications to the student architectures are made, meaning we achieve a boost in accuracy without impacting inference throughput. Yet, we note that training times are affected, albeit not necessarily if a pre-trained teacher model is available (see Appendix H). With this work, we aim to elucidate the potential of KD in the domain of molecular simulations and stimulate future work in the area.

Acknowledgements

F.E.K. is financially supported by the Excellence Center at Linköping–Lund in Information Technology (ELLIIT). D.G. is supported by UK Research and Innovation [UKRI Centre for Doctoral Training in AI for Healthcare grant number EP/S023283/1]. Computing resources provided by: the Berzelius resource at the National Supercomputer Centre, provided by Knut and Alice Wallenberg Foundation; the Alvis resource provided by the National Academic Infrastructure for Supercomputing in Sweden (NAISS) at Chalmers e-Commons at Chalmers (C3SE) partially funded by the Swedish Research Council through grant agreement no. 2022-06725; the Chair of Aerodynamics and Fluid Mechanics at Technical University of Munich. This research project was initially conceived at the 2022 LOGML summer school, and we would like to thank Guocheng Qian and I-Ju Chen for their contribution during the early conceptualizing stages of this project during and in the first weeks following the summer school. We also thank the Open Catalyst team for their open-source codebase, support and discussions. In particular, Muhammed Shuaibi for providing the COLL dataset in LMDB format.

References

[1] Frank Noé, Alexandre Tkatchenko, Klaus-Robert Müller, and Cecilia Clementi. Machine learning for molecular simulation. *Annual review of physical chemistry*, 71:361–390, 2020. 1

[2] Julia Westermayr, Michael Gastegger, Kristof T Schütt, and Reinhard J Maurer. Perspective on integrating machine learning into computational chemistry and materials science. *The Journal of Chemical Physics*, 154(23):230903, 2021. 1

[3] Patrick Reiser, Marlen Neubert, André Eberhard, Luca Torresi, Chen Zhou, Chen Shao, Houssam Metni, Clint van Hoesel, Henrik Schopmans, Timo Sommer, et al. Graph neural networks for materials science and chemistry. *Communications Materials*, 3(1):93, 2022. 1

[4] Yuyang Wang, Zijie Li, and Amir Barati Farimani. Graph neural networks for molecules, 2023.

[5] Xuan Zhang, Limei Wang, Jacob Helwig, Youzhi Luo, Cong Fu, Yaochen Xie, Meng Liu, Yuchao Lin, Zhao Xu, Keqiang Yan, Keir Adams, Maurice Weiler, Xiner Li, Tianfan Fu, Yucheng Wang, Haiyang Yu, YuQing Xie, Xiang Fu, Alex Strasser, Shenglong Xu, Yi Liu, Yuanqi Du, Alexandra Saxton, Hongyi Ling, Hannah Lawrence, Hannes Stärk, Shurui Gui, Carl Edwards, Nicholas Gao, Adriana Ladera, Tailin Wu, Elyssa F. Hofgard, Aria Mansouri Tehrani, Rui Wang, Ameya Daigavane, Montgomery Bohde, Jerry Kurtin, Qian Huang, Tuong Phung, Minkai Xu, Chaitanya K. Joshi, Simon V. Mathis, Kamyar Azizzadenesheli, Ada Fang, Alán Aspuru-Guzik, Erik Bekkers, Michael Bronstein, Marinka Zitnik, Anima Anandkumar, Stefano Ermon, Pietro Liò, Rose Yu, Stephan Günnemann, Jure Leskovec, Heng Ji, Jimeng Sun, Regina Barzilay, Tommi Jaakkola, Connor W. Coley, Xiaoning Qian, Xiaofeng Qian, Tess Smidt, and Shuiwang Ji. Artificial intelligence for science in quantum, atomistic, and continuum systems, 2023. 1

[6] Simon Batzner, Albert Musaelian, Lixin Sun, Mario Geiger, Jonathan P. Mailoa, Mordechai Kornbluth, Nicola Molinari, Tess E. Smidt, and Boris Kozinsky. E(3)-equivariant graph neural networks for data-efficient and accurate interatomic potentials. *Nature Communications*, 13(1): 2453, May 2022. ISSN 2041-1723. doi: 10.1038/s41467-022-29939-5. 1, 8

[7] Johannes Gasteiger, Janek Groß, and Stephan Günnemann. Directional Message Passing for Molecular Graphs. In *International Conference on Learning Representations*, 2020. 8

[8] Johannes Gasteiger, Florian Becker, and Stephan Günnemann. GemNet: Universal Directional Graph Neural Networks for Molecules. In *Advances in Neural Information Processing Systems*, volume 34, pages 6790–6802. Curran Associates, Inc., 2021. 1, 8

[9] Albert Musaelian, Simon Batzner, Anders Johansson, Lixin Sun, Cameron J. Owen, Mordechai Kornbluth, and Boris Kozinsky. Learning local equivariant representations for large-scale atomistic dynamics. *Nature Communications*, 14(1):579, February 2023. ISSN 2041-1723. doi: 10.1038/s41467-023-36329-y.

[10] Larry Zitnick, Abhishek Das, Adeesh Kolluru, Janice Lan, Muhammed Shuaibi, Anuroop Sriram, Zachary Ulissi, and Brandon Wood. Spherical channels for modeling atomic interactions. *Advances in Neural Information Processing Systems*, 35:8054–8067, 2022. 1, 8

[11] Anuroop Sriram, Abhishek Das, Brandon M Wood, Siddharth Goyal, and C Lawrence Zitnick. Towards training billion parameter graph neural networks for atomic simulations. *arXiv preprint arXiv:2203.09697*, 2022.

[12] Saro Passaro and C. Lawrence Zitnick. Reducing SO(3) convolutions to SO(2) for efficient equivariant GNNs. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 27420–27438. PMLR, 23–29 Jul 2023. URL https://proceedings.mlr.press/v202/passaro23a.html. 1

[13] Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. Fitnets: Hints for thin deep nets, 2015. 2, 3

[14] Gustavo Aguilar, Yuan Ling, Yu Zhang, Benjamin Yao, Xing Fan, and Chenlei Guo. Knowledge distillation from internal representations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 7350–7357, 2020. 2

[15] Johannes Gasteiger, Muhammed Shuaibi, Anuroop Sriram, Stephan Günnemann, Zachary Ward Ulissi, C. Lawrence Zitnick, and Abhishek Das. GemNet-OC: Developing Graph Neural Networks for Large and Diverse Molecular Simulation Datasets. *Transactions on Machine Learning Research*, October 2022. ISSN 2835-8856. 2, 8, 9

[16] Kristof Schütt, Oliver Unke, and Michael Gastegger. Equivariant message passing for the prediction of tensorial properties and molecular spectra. In *Proceedings of the 38th International Conference on Machine Learning*, pages 9377–9388. PMLR, July 2021. 2, 8, 9

[17] 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. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. 2, 8, 9

[18] Yiding Yang, Jiayan Qiu, Mingli Song, Dacheng Tao, and Xinchao Wang. Distilling knowledge from graph convolutional networks, 2021. 3

[19] Chaitanya K. Joshi, Fayao Liu, Xu Xun, Jie Lin, and Chuan Sheng Foo. On representation knowledge distillation for graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–12, 2022. doi: 10.1109/tnnls.2022.3223018. URL https://doi.org/10.1109%2Ftnnls.2022.3223018.

[20] Lu Yu, Shichao Pei, Lizhong Ding, Jun Zhou, Longfei Li, Chuxu Zhang, and Xiangliang Zhang. Sail: Self-augmented graph contrastive learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 8927–8935, 2022.

[21] Cuiying Huo, Di Jin, Yawen Li, Dongxiao He, Yu-Bin Yang, and Lingfei Wu. T2-gnn: Graph neural networks for graphs with incomplete features and structure via teacher-student distillation. *arXiv preprint arXiv:2212.12738*, 2022. 3

[22] Huarui He, Jie Wang, Zhanqiu Zhang, and Feng Wu. Compressing deep graph neural networks via adversarial knowledge distillation. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 534–544, 2022. 3

[23] 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. The Open Catalyst 2020 (OC20) Dataset and Community Challenges. *ACS Catalysis*, 11(10):6059–6072, May 2021. ISSN 2155-5435, 2155-5435. doi: 10.1021/acscatal.0c04525. 3, 4

[24] Johannes Gasteiger, Shankari Giri, Johannes T. Margraf, and Stephan Günnemann. Fast and uncertainty-aware directional message passing for non-equilibrium molecules. In *Machine Learning for Molecules Workshop, NeurIPS*, 2020. 3, 8

[25] Yu Cheng, Duo Wang, Pan Zhou, and Tao Zhang. Model compression and acceleration for deep neural networks: The principles, progress, and challenges. *IEEE Signal Processing Magazine*, 35(1):126–136, 2018. 8, 9

[26] Lin Wang and Kuk-Jin Yoon. Knowledge distillation and student-teacher learning for visual intelligence: A review and new outlooks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021. 8

[27] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*, 2019. 8

[28] Jianping Gou, Baosheng Yu, Stephen J Maybank, and Dacheng Tao. Knowledge distillation: A survey. *International Journal of Computer Vision*, 129:1789–1819, 2021. 8

[29] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the Knowledge in a Neural Network, March 2015. 8

[30] Chengming Hu, Xuan Li, Dan Liu, Xi Chen, Ju Wang, and Xue Liu. Teacher-student architecture for knowledge learning: A survey. *arXiv preprint arXiv:2210.17332*, 2022. 8

[31] Qing Xu, Zhenghua Chen, Mohamed Ragab, Chao Wang, Min Wu, and Xiaoli Li. Contrastive adversarial knowledge distillation for deep model compression in time-series regression tasks. *Neurocomputing*, 485:242–251, 2022. 8, 9

[32] Yijun Tian, Shichao Pei, Xiangliang Zhang, Chuxu Zhang, and Nitesh V. Chawla. Knowledge distillation on graphs: A survey, 2023. 8

[33] Jing Liu, Tongya Zheng, Guanzheng Zhang, and Qinfen Hao. Graph-based knowledge distillation: A survey and experimental evaluation. *arXiv preprint arXiv:2302.14643*, 2023. 8

[34] Muhamad Risqi U Saputra, Pedro PB De Gusmao, Yasin Almalioglu, Andrew Markham, and Niki Trigoni. Distilling knowledge from a deep pose regressor network. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 263–272, 2019. 8, 9

[35] Jang Hyun Cho and Bharath Hariharan. On the efficacy of knowledge distillation. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 4794–4802, 2019. 9

[36] Pengguang Chen, Shu Liu, Hengshuang Zhao, and Jiaya Jia. Distilling knowledge via knowledge review. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5008–5017, 2021. 10

[37] Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. Similarity of neural network representations revisited, 2019. 10

[38] Maithra Raghu, Thomas Unterthiner, Simon Kornblith, Chiyuan Zhang, and Alexey Dosovitskiy. Do vision transformers see like convolutional neural networks? 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=Gl8FHfMVTZu. 10

A Background

A.1 GNNs for molecular systems.

GNNs are a suitable framework for modeling molecular systems. Each molecular system $(\boldsymbol{X}, \boldsymbol{z})$ can be represented as a mathematical graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where the nodes $\mathcal{V}$ correspond to the set of atoms, and edges $\mathcal{E}$ are created between nodes by connecting the closest neighboring atoms (typically defined by a cutoff radius and/or a maximum number of neighbors). Hence, in the context of molecular simulations, we can create a GNN that operates on atomic graphs $\mathcal{G}$ by propagating information between the atoms and the edges, and makes predictions about the energy $E \in \mathbb{R}$ of the system, and the forces $\boldsymbol{F} \in \mathbb{R}^{N\times 3}$ acting on each atom in a multi-output manner - i.e., $\hat{E}, \hat{\boldsymbol{F}} = \text{GNN}(\boldsymbol{X}, \boldsymbol{z})$. Both these properties are of high interest when simulating molecular systems. The energy of a system is essential for the prediction of its stability, whereas the forces are important for molecular dynamics simulations, where computed forces are combined with the equations of motion to simulate the evolution of the system over time.

The main problem when modeling molecules and molecular properties is the number of underlying symmetries to account for, most importantly rigid transformations of the atoms. For instance, the total energy E of a system is not affected by (i.e., is *invariant* to) rotations and translations of the system. However, the forces $\boldsymbol{F}$ do change as we rotate a system - i.e., they are *equivariant* to rotations. Therefore, to make accurate predictions about molecular systems, it is crucial to devise models that respect these symmetries and other physical constraints. There is now a plethora of diverse molecular GNNs that achieve that, e.g., SchNet [17], DimeNet [7, 24], PaiNN [16], GemNet [8, 15], NequIP [6], and SCN [10], which have incrementally established a more holistic description of molecular systems by capturing advanced geometric features and physical symmetries. This has, however, come at the expense of computational efficiency, making state-of-the-art models orders of magnitude slower than more lightweight options.

A.2 Knowledge distillation.

Knowledge distillation is a technique for compressing and accelerating ML models [25], which has recently demonstrated significant potential in domains like computer vision [26] and natural language modeling [27]. The main objective of KD is to create more efficient models by means of transferring knowledge (e.g. model parameters and activations) from large, computationally expensive, more accurate models, often referred to as teacher models, to simpler, more efficient models called student models [28]. Since the seminal work of Hinton et al. [29], the field has drastically expanded methodologically, with the development of protocols that accommodate the distillation of "deeper" knowledge, more comprehensive transformation and fusion functions, as well as more robust distillation losses [28, 30]. Yet, these advances have mostly focused on classification, resulting in methods of limited utility in regression tasks [31]. Moreover, most research in the area has been confined to non-graph data (e.g., images, text, tabular data). Despite recent efforts to extend KD to graph data and GNNs, these have likewise only concentrated on classification tasks involving standard GNN architectures [32, 33]. And, in particular, the application of KD to large-scale regression problems in molecular simulations, which involve state-of-the-art molecular GNN architectures containing complex, geometric node- and edge-level features, is still unexplored.

A.3 Output-based KD.

KD was originally proposed in the context of classification by leveraging the fact that the soft label predictions (i.e., the logits after softmax normalization) of a given (teacher) model carry valuable information that can complement the ground-truth labels in the training process of another (student) model [29]. Since then, this has become the standard KD approach - commonly referred to as vanilla KD in the literature, which is often the foundation of new KD protocols. The main idea of this technique is to employ a KD loss $\mathcal{L}_{\text{KD}}$ that forces the student to mimic the predictions of the teacher model. This is usually achieved by constructing a loss $\mathcal{L}_{\text{KD}} = \text{KL}(z_s, z_t)$ based on the Kullback–Leibler (KL) divergence between the soft logits of the student z_s and the teacher z_t.

However, this strategy - based on the distillation of the output of the teacher model only [28] - poses two significant limitations. First, it is by design exclusively applicable to classification tasks, since there are no outputs analogous to logits in regression setups [25, 34]. This has consequently limited the utility of most KD methods for regression tasks. Second, this approach forces the student to

emulate the final output of the teacher directly, which can be unattainable in regimes where the complexity gap between the two models is substantial, and thus detrimental to KD performance [35].

B Description of features

- *SchNet* [17]: A simple GNN model based on continuous-filter convolutional layers, which only contains scalar node features $\boldsymbol{s} \in \mathbb{R}^d$. These are used to predict the energy, $\hat{E}$. The force is then calculated as the negative gradient of the energy with respect to the atomic positions, i.e., $\hat{\boldsymbol{F}} = -\nabla \hat{E}$.
- *PaiNN* [16]: A GNN based on equivariant message passing, which contains scalar node features $\boldsymbol{x} \in \mathbb{R}^{d_1}$ - used for energy prediction; as well as geometric vectorial node features, $\boldsymbol{v} \in \mathbb{R}^{3\times d_2}$ that are equivariant to rotations and can thus be combined with the scalar features to make direct predictions of the forces (i.e., without computing gradients of the energy).
- *GemNet-OC* [15]: A GNN model that utilizes directional message passing. It contains scalar node features $\boldsymbol{h} \in \mathbb{R}^{d_\mathrm{h}}$ and scalar edges features $\boldsymbol{m} \in \mathbb{R}^{d_\mathrm{m}}$. After each block of layers, these are processed through an output block, resulting in scalar node features $\boldsymbol{x}_\mathrm{E}^{(i)}$ and edge features $\boldsymbol{x}_\mathrm{F}^{(i)}$, where i is the block number. The output features from each block are aggregated into output features $\boldsymbol{x}_E$ and $\boldsymbol{x}_F$, which are used to compute the energy and forces respectively.

Table 2: Molecular GNNs can have diverse features depending on their architecture. This is an overview of the types of features available in the three models we use in this study.

	SchNet	PaiNN	GemNet-OC
Scalar node features	✓	✓	✓
Scalar edge features			✓
Vectorial node features		✓	
Output blocks			✓

C Additional KD strategies

Vanilla (1): As mentioned, the main problem with using vanilla KD for regression is the lack of features analogous to logits. One way of adapting vanilla KD for regression is by steering the student to mimic the final output of the teacher directly [31]:

$$\mathcal{L}_\mathrm{KD} = \alpha_\mathrm{E}\mathcal{L}_\mathrm{E}(\hat{E}_\mathrm{s}, \hat{E}_\mathrm{t}) + \alpha_\mathrm{F}\mathcal{L}_\mathrm{F}(\hat{\boldsymbol{F}}_\mathrm{s}, \hat{\boldsymbol{F}}_\mathrm{t}), \tag{7}$$

where the subscripts $_\mathrm{s}$ and $_\mathrm{t}$ refer to the predictions of the student and teacher, respectively. Note that, unlike in classification, this approach does not provide much additional information in regression tasks, except for some limited signal about the error distribution of the teacher model [25, 34].

Vanilla (2): One way to enhance the teacher signal during training is to consider the fact that many GNNs for molecular simulations make separate atom- and edge-level predictions which are consequently aggregated into a final output. For instance, the total energy E of a system is usually defined as a sum of the predicted contributions from each atom $\hat{E} = \sum_i \hat{E}_i$. Hence, we note that we can extend the aforementioned vanilla KD approach by imposing a loss on these granular predictions instead. Following the energy definition above, the KD loss can be expressed as

$$\mathcal{L}_\mathrm{KD} = \frac{1}{N}\sum_{i=1}^{N} \mathcal{L}_\mathrm{E}(\hat{\mathrm{E}}_{i,\mathrm{s}}, \hat{E}_{i,\mathrm{t}}). \tag{8}$$

These individual energy contributions are not part of the labeled data, but, when injected during training, provide more fine-grained information than the aggregated prediction.

D Training and hyperparameters

We utilize the following setup: we use MSE as a distillation loss $\mathcal{L}_\text{feat}$; a learned linear layer as a transformation function $\mathcal{M}_s$ on the features of the student; and the identity transformation as $\mathcal{M}_t$.

When distilling knowledge from GemNet-OC, we use the aggregated output block features $\boldsymbol{x}_{\mathrm{E}}$ and $\boldsymbol{x}_{\mathrm{F}}$ as node features and edge features, respectively. This is similar to the review-based setup proposed in [36]. For PaiNN and SchNet, we use the final node features.

The models were trained on NVIDIA A100 40 GB and NVIDIA RTX A6000 48 GB GPUs. All models were trained on single GPUs, except for SchNet when trained on OC20, which required 3 GPUs. In Tables 3-5, hyperparameters for the models are presented. In Table 6, the weighting of the KD are presented.

E Defininig teacher-student configurations based on model similarity

We investigated 3 different teacher-student configurations with varying levels of architectural disparity as measured with central kernel alignment [37, 38] (see Figure 1):

- *same* architecture: distilling our default PaiNN model (PaiNN-big) to a smaller version with four instead of six layers, and 256 hidden dimensions instead of 512 (PaiNN-small);
- *similar* architecture: distilling PaiNN-big to SchNet;
- *different* architecture: distilling GemNet-OC to PaiNN-big.

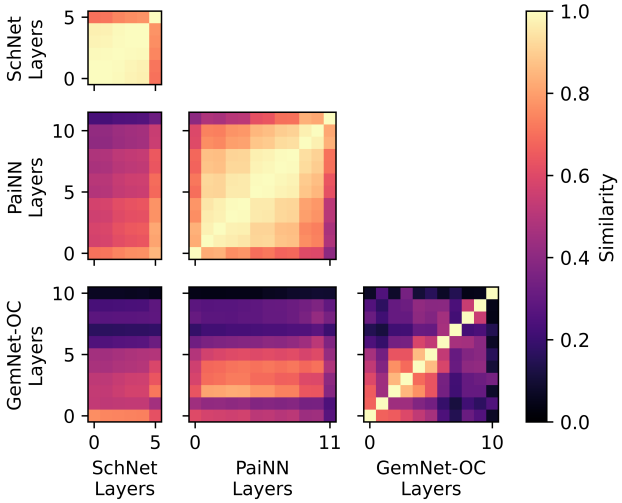

Figure 1: Similarity analysis between the node features of SchNet, PaiNN and GemNet-OC using CKA.

F Baseline results

Table 7 summarizes the performance of our models without any knowledge distillation on OC20 and COLL. Results on OC20 represent an average across the 4 validation datasets available in OC20. We train SchNet, PaiNN-small and PaiNN-big to convergence ourselves, whereas the GemNet-OC model we employ is the pre-trained model as available within the OC20 repository.

G COLL results

Table 8 summarizes our benchmarking results on the COLL dataset.

H Training times

One caveat of knowledge distillation is that it inherently increases the training time of the student model. In our offline KD setup, we need to perform additional forward passes through the teacher

Table 3: GemNet-OC hyperparameters. For the experiments we have conducted on OC20 data, we used the model pretrained on OC20-2M (July 2022), and we have hence not done any training ourselves.

Hyperparameter	OC20	COLL
No. spherical basis	7	7
No. radial basis	128	128
No. blocks	4	4
Atom embedding size	256	128
Edge embedding size	512	256
Triplet edge embedding input size	64	64
Triplet edge embedding output size	64	64
Quadruplet edge embedding input size	32	32
Quadruplet edge embedding output size	32	32
Atom interaction embedding input size	64	64
Atom interaction embedding output size	64	64
Radial basis embedding size	16	16
Circular basis embedding size	16	16
Spherical basis embedding size	32	32
No. residual blocks before skip connection	2	2
No. residual blocks after skip connection	2	2
No. residual blocks after concatenation	1	1
No. residual blocks in atom embedding blocks	3	3
No. atom embedding output layers	3	3
Cutoff	12.0	12.0
Quadruplet cutoff	12.0	12.0
Atom edge interaction cutoff	12.0	12.0
Atom interaction cutoff	12.0	12.0
Max interaction neighbors	30	30
Max quadruplet interaction neighbors	8	8
Max atom edge interaction neighbors	20	20
Max atom interaction neighbors	1000	1000
Radial basis function	Gaussian	Gaussian
Circular basis function	Spherical harmonics	Spherical Harmonics
Spherical basis function	Legendre Outer	Legendre Outer
Quadruplet interaction	True	True
Atom edge interaction	True	True
Edge atom interaction	True	True
Atom interaction	True	True
Direct forces	True	True
Activation	Silu	Silu
Optimizer	-	AdamW
Scheduler	-	LinearWarmupExponentialDecay
Force coefficient	-	100
Energy coefficient	-	1
EMA decay	-	0.999
Gradient clip norm threshold	-	10
Initial learning rate	-	10^{-3}

Table 4: PaiNN hyperparameters. The different number of layers and hidden channels indicate the values we have used for PaiNN-big and PaiNN-small, respectively.

Hyperparameter	OC20	COLL
Hidden channels	512/256	256/128
Number of layers	6/4	6/4
Number of RBFs	128	128
Cutoff	12.0	12.0
Max. num. neighbors	50	50
Direct Forces	True	True
Batch size	32	32
Optimizer	AdamW	AdamW
AMSGrad	True	True
Initial learning rate	10^{-4}	10^{-3}
Scheduler	LambdaLR	LinearWarmupExponentialDecay
Warmup steps	None	3750
Learning rate decay factor	0.45	0.01
Learning rate milestones (steps)	160000, 320000, 480000, 640000	-
Force coefficient	100	100
Energy coefficient	1	1
EMA decay	0.999	0.999
Gradient clip norm threshold	10	10

Table 5: SchNet hyperparameters. For the experiments we have conducted on OC20 data, we used the default configuration for the OCP-2M dataset as provided in the OCP repository.

Hyperparameter	OC20	COLL
Hidden channels	1024	128
Filters	256	128
Interaction blocks	5	6
Gaussians	200	50
Cutoff	6.0	12.0
Batch size	192	32
Initial learning rate	10^{-4}	10^{-3}
Optimizer	AdamW	AdamW
Scheduler	LambdaLR	LinearWarmupExponentialDecay
Learning rate decay factor	0.1	0.01
Learning rate milestones	52083, 83333, 104166	-
Warmup steps	31250	3750
Warmup factor	0.1	-
Force Coefficient	100	100
Energy Coefficient	1	1

Table 6: Choice of the weighting factor λ of the KD loss for the different teacher-student configurations and KD strategies.

Teacher	Student	Loss	OC20	COLL
GemNet-OC	PaiNN-big	vanilla (1)	1.0	0.2
GemNet-OC	PaiNN-big	vanilla (2)	500	100
GemNet-OC	PaiNN-big	n2n	10000	1000
GemNet-OC	PaiNN-big	e2n	10	10
GemNet-OC	PaiNN-big	v2v	50000	100
PaiNN-big	PaiNN-small	vanilla (1)	1	1
PaiNN-big	PaiNN-small	vanilla (2)	200	100
PaiNN-big	PaiNN-small	n2n	100	100
PaiNN-big	PaiNN-small	v2v	1000	10000
PaiNN-big	SchNet	vanilla (1)	0.1	1
PaiNN-big	SchNet	vanilla (2)	0.1	100
PaiNN-big	SchNet	n2n	1000	100

Table 7: Evaluation of the performance of the four baseline GNN models considered in this study (i.e. no knowledge distillation) on: OC20 (*top*); and COLL (*bottom*). The results show a clear tradeoff between accuracy and computational cost.

	Inference Throughput	**OC20** S2EF Validation			
Model	Samples / GPU sec. ↑	Energy MAE meV ↓	Force MAE meV/Å ↓	Force cos ↑	EFwT % ↑
SchNet	788.2	1308	65.1	0.204	0
PaiNN-small	618.2	489	47.1	0.345	0.085
PaiNN-big	237.8	440	45.3	0.376	0.14
GemNet-OC	75.8	286	25.7	0.598	1.06

	Inference Throughput	**COLL** test set			
Model	Samples / GPU sec. ↑	Energy MAE meV ↓	Force MAE meV/Å ↓	Force cos ↑	EFwT % ↑
SchNet	35000	146.5	121.2	0.970	2.75
PaiNN-small	27000	104.0	80.9	0.984	5.4
PaiNN-big	12000	85.8	64.1	0.988	10.1
GemNet-OC	2600	44.8	38.2	0.994	20.2

to extract representations to distill to the student. However, it is important to note that, despite increasing the computational time per training step, we observed that models trained with KD consistently outperformed their baseline counterparts even when compared at the same training time point (Figure 2), despite the latter having been trained for more steps/epoch in total. This means that, all in all, we can use KD to enhance the predictive accuracy in models without necessarily impacting training times.

However, we make the following remark. In our experiments, we utilized publicly available pre-trained Gemnet-OC model weights, and therefore did not have to train the teacher model ourselves. However, when access to a pre-trained teacher model is not available, one should also account for the time required to train the teacher.

Table 8: Evaluation results on the COLL test set. Numbers in brackets represent the proportion of the gap between the student and the teacher that has been closed by the respective KD strategy (in %). Best results are given in **bold**.

		COLL test set			
	Model	Energy MAE meV ↓	Force MAE meV/Å ↓	Force cos ↑	EFwT % ↑
same	Student (PaiNN-small)	104.0	80.9	0.984	5.4
	Teacher (PaiNN-big)	85.8	64.1	0.988	10.1
	Vanilla KD (1)	106.1(-11.5%)	82.0(-6.5%)	0.984(2.3%)	4.46(-20.2%)
	Vanilla KD (2)	**86.4 (96.7%)**	80.9(0%)	0.983(-2.3%)	4.3(-23.7%)
	n2n	92.5(63.2%)	77.8(18.5%)	0.984(18.2%)	**6.63 (26.5%)**
	v2v	90.4(74.7%)	**70.4 (62.5%)**	**0.986 (45.5%)**	5.8(8.4%)
similar	Student (SchNet)	146.5	121.2	0.970	2.75
	Teacher (PaiNN-big)	85.8	64.1	0.988	10.1
	Vanilla KD (1)	146.1(0.7%)	120.8(0.7%)	0.970(1.1%)	2.54(-2.9%)
	Vanilla KD (2)	**104.1 (69.9%)**	120.9(0.5%)	0.970(1.1%)	**6.45 (50.7%)**
	n2n	141.6(8.1%)	**117.2 (7.0%)**	**0.971 (5.4%)**	2.63(-1.6%)
different	Student (PaiNN-big)	85.8	64.1	0.988	10.1
	Teacher (GemNet-OC)	44.8	38.2	0.994	20.2
	Vanilla KD (1)	86.2(-1.1%)	63.9(0.6%)	0.988(1.5%)	10.1(0.1%)
	Vanilla KD (2)	61.4(59.5%)	62.9(4.6%)	0.988(5.2%)	13.0(29.2%)
	n2n	**60.4 (62.0%)**	**61.2 (11.3%)**	**0.989 (14.9%)**	**13.6 (34.6%)**
	e2n	77.3(20.8%)	63.3(3.0%)	0.988(7.9%)	11.0(9.2%)
	v2v	81.2(11.2%)	63.3(3.1%)	0.988(3.4%)	10.5(4.6%)

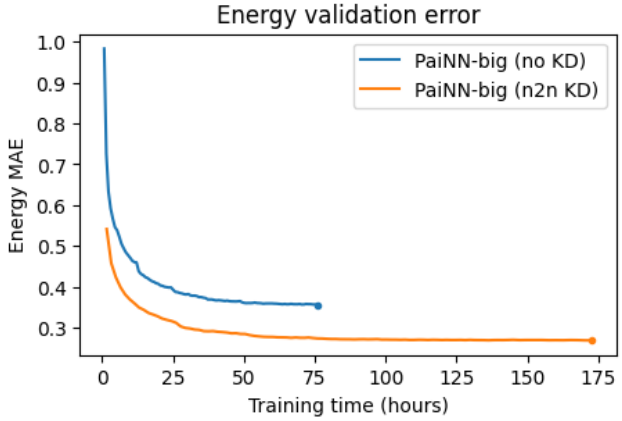

Figure 2: Energy validation error of PaiNN without (*blue*) and with (*orange*) knowledge distillation from GemNet-OC, trained for the same number of steps (1 million). Validation on a random sample of size 30k samples from the in-distribution OC20 validation set.