3D Pre-training improves GNNs for Molecular Property Prediction

Anonymous Author(s)
Affiliation
Address
email

Abstract

Molecular property prediction is one of the fastest-growing applications of deep learning with critical real-world impacts. Including 3D molecular structure as input to learned models improves their predictions for many molecular properties. However, this information is infeasible to compute at the scale required by most real-world applications. We propose pre-training a model to understand the geometry of molecules given only their 2D molecular graph. Using methods from self-supervised learning, we maximize the mutual information between a 3D summary vector and the representations of a Graph Neural Network (GNN) such that they contain latent 3D information. During fine-tuning on molecules with unknown geometry, the GNN still generates implicit 3D information and can use it to inform downstream tasks. We show that 3D pre-training provides significant improvements for a wide range of molecular properties, such as a 22% average MAE reduction on eight quantum mechanical properties. Crucially, the learned representations can be effectively transferred between datasets with vastly different molecules.

1 Introduction

The understanding of molecular and quantum chemistry is a rapidly growing area for deep learning with models having direct real-world impacts in quantum chemistry (Dral 2020), protein structure prediction (Jumper et al. 2021), materials science (Schmidt et al. 2019), and drug discovery (Stokes et al. 2020). In particular, for the task of molecular property prediction, GNNs have had great success (Yang et al. 2019).

GNNs operate on the molecular graph by updating each atom's representation based on the atoms connected to it via covalent bonds. However, these models mostly cannot reason about other important interatomic forces that depend on the atoms' relative positions in space. Previous works have shown that using the atoms' 3D coordinates in space improves the accuracy of molecular property prediction (Schütt et al. 2017; Klicpera, Groß, et al. 2020; Liu et al. 2021; Klicpera, Becker, et al. 2021).

However, using classical methods to explicitly compute a molecule's geometry before predicting its properties is computationally infeasible for many real-world applications. Even recent machine learning methods for conformation generation (Xu et al. 2021; Shi et al. 2021; Ganea et al. 2021) are still too slow for large scale screening.

Our Solution We pre-train GNNs to capture implicit 3D information in their latent vectors using publicly available molecular structures. The GNN is pre-trained by maximizing the mutual information between its embedding of a molecular graph and a representation capturing the 3D information that is produced by a separate network. This forces the GNN to learn to generate latent 3D information using only the information given by the 2D molecular graphs. After pre-training, the weights can be transferred and fine-tuned on molecular datasets where no 3D information is available.

Submitted to NeurIPS 2021 AI for Science Workshop.

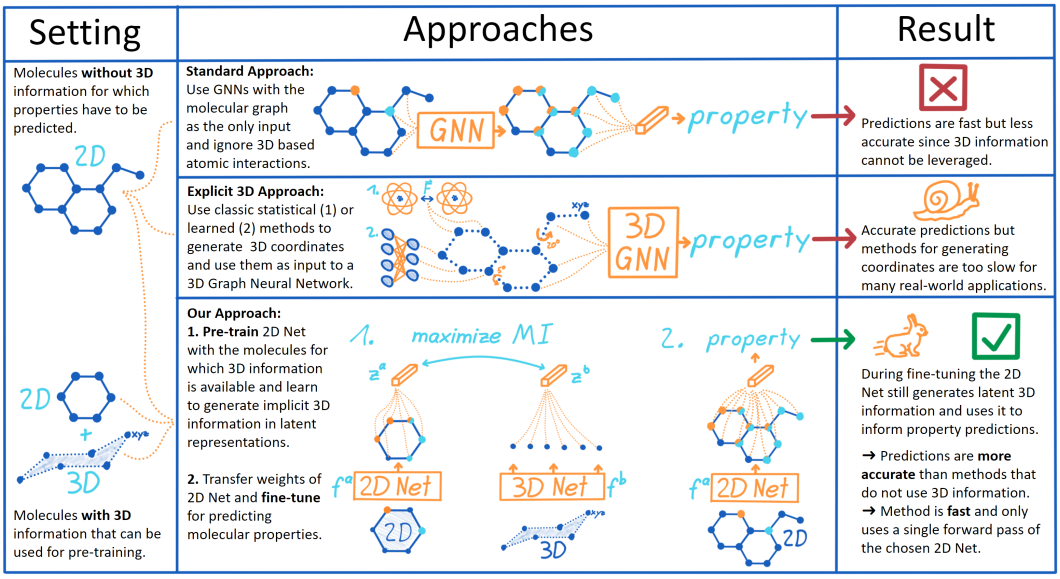

Figure 1: The considered problem setting and the motivation for our pre-training approach.

For those molecules, the GNN is still able to generate implicit 3D information and the network can use it to inform its property predictions.

We analyze our method's performance by pre-training with multiple 3D datasets before evaluating on 10 quantum mechanical molecular properties. 3D pre-training improves property predictions by large margins with a 22% average MAE reduction for the QM9 dataset's properties. Furthermore, the learned representations are highly generalizable. Indeed, significant improvements are obtained even when the molecular space of the pre-training dataset is vastly different (e.g., in size) from the kinds of molecules in the downstream tasks.

2 Method

Our method consists of two steps: (1) A model is pre-trained using molecules with known geometry. (2) The pre-trained model is fine-tuned to predict properties from 2D molecular graphs. The goal during pre-training is that the 2D network learns to generate latent information about 3D structure and quantum mechanics, which can be used to improve molecular property predictions during fine-tuning.

The pre-training uses two different models, as visualized in the last row of Figure 1. Firstly, the model that should be pre-trained which we call *2D network* f^a since it takes 2D molecular graphs $G = (\mathcal{V}, \mathcal{E})$ with atoms $\mathcal{V}$ and bonds $\mathcal{E}$ to generate a latent representation of a molecule $f^a(G) = z^a \in \mathbb{R}^{d_z}$. Secondly, we define the *3D network* f^b as the model which encodes the atoms' 3D coordinates $R = \{r_v\}_{v \in \mathcal{V}}$ in a 3D representation $f^b(R) = z^b \in \mathbb{R}^{d_z}$.

Contrastive Learning To teach the 2D GNN f^a to generate 3D information from the 2D graph inputs, we maximize the mutual information between the latent 2D representations z^a and 3D representations z^b. Intuitively we wish to maximize the agreement between z^a and z^b if they come from the same molecule. For this purpose, we use contrastive learning (visualized in Figure 3) for which we consider a batch of N molecules and the networks produce multiple representations z_i^a and z_i^b. The first objective of contrastive learning is to maximize the representation's similarity if they are a positive pair, meaning that they come from the same molecule with the same index i. The second part is to enforces dissimilarity between negative pairs z_i^a and z_k^b where $i \neq k$. In this work, we use the cosine similarity $sim(z^a, z^b) = z^a \cdot z^b / (\|z^a\| \|z^b\|)$ and optimize the models using the popular normalized temperature-scaled cross entropy loss (NTXent) (Chen et al. 2020).

Multiple Conformers For most molecules, there are multiple plausible 3D configurations called *conformers*. Instead of only using the most probable conformer (with the lowest energy), we found

that leveraging structural information from multiple conformers provides significant benefits. To achieve this, we now consider the c many highest probability conformers $\{R_i^j\}_{j\in\{1\dots c\}}$ of the i-th molecule. For every molecule the 3D network now takes all conformers as input and produces their latent 3D representations $\{z_{i,j}^b\}_{j\in\{1\dots c\}}$. These conformers can be seen as additional positive samples. Thus, the objective is to maximize the similarity between z_i^a and all conformer representations $z_{i,j}^b$ that come from the same molecule. As such we modify the NTXent loss to sum over the similarities of all conformers and obtain our final objective with the temperature parameter τ:

$$\mathcal{L}_{NTXent}^{multi3D} = -\frac{1}{N}\sum_{i=1}^{N}\left[log\,\frac{\sum_{j=1}^{c} e^{sim(z_i^a,\, z_{i,j}^b)/\tau}}{\sum_{\substack{k=1\\k\neq i}}^{N}\sum_{j=1}^{c} e^{sim(z_i^a,\, z_{k,j}^b)/\tau}}\right]. \tag{1}$$

3D Network The 3D coordinates of molecules have a certain set of symmetries that the 3D network should capture. Under rotation or translation, a molecule's properties do not change. Therefore, the 3D representation z^b should not change either and be invariant to these transformations. Our 3D network achieves this as a GNN operating on the fully connected graph of a molecule with the atom distances as edge features. All node features in the first layer are the same learned vector which is initialized from a standard normal distribution. The message passing layers iteratively encode the 3D information into the node features which are pooled to produce the complete 3D representation z^b.

3 Experiments

Setup We choose Principal Neighborhood Aggregation (PNA) (Corso et al. 2020) as the GNN architecture to pre-train due to its simplicity and state-of-the-art performance for multiple molecular tasks. For pre-training, we use three datasets of molecules with 3D information: QM9 (Ramakrishnan et al. 2014), GEOM-Drugs (Axelrod and Gomez-Bombarelli 2020), and QMugs (Isert et al. 2021). More precisely, we pre-train three different instances of PNA (1) on $50\,000$ molecules from QM9 using a single conformer, (2) on $140\,000$ of GEOM-Drugs with 5 conformers and, (3) on $620\,000$ of QMugs using 3 conformers. For comparison, we pre-train baselines by either predicting the properties of GEOM-Drugs' pre-training subset (labeled *Prop Pred*) or by using Graph Contrastive Learning (GraphCL) (You et al. 2020), a strong conventional contrastive learning approach that uses the full GEOM-Drugs dataset. All pre-training methods use a batch size of 500.

After pre-training, the models are fine-tuned on $50\,000$ molecules from QM9 or $140\,000$ from GEOM-Drugs that have no overlap with the molecules from the pre-training data. With the same molecules, we also train PNA with random weight initialization (labeled *Rand Init*) to compare how much the downstream performance is improved by the different pre-training methods. Lastly, we train the 3D GNN Spherical Message Passing (SMP) (Liu et al. 2021) on the same subset as a method that uses 3D information explicitly. All experimental settings (including SMP and GraphCL) are detailed in Appendix C and we provide all code to reproduce the results: `https://anonymous.4open.science/r/3D-pretrain-3726`.

Results. Table 1 shows that 3D pre-training leads to large improvements over the randomly initialized baseline and over GraphCL with all three pre-training datasets. After 3D pre-training on one half of QM9, the average of the MAE decreases by 22%. Especially pre-training with GEOM-Drugs (Drugs 3D) and GraphCL is interesting to compare since GraphCL is pre-trained with two times as many molecules from the same dataset, but 3D pre-training is always better by a large margin.

Mostly pre-training with the disjoint half of QM9 (column QM9) performs the best since the pre-training and fine-tuning dataset are from the same molecular space. However, the learned representations also generalize well with pre-training on GEOM-Drugs and QMugs leading to average improvements of 19% and 20% respectively. This is the case even though QM9 contains much smaller molecules with on average 18 atoms compared to the 44.4 atoms for the drug-like molecules of GEOM-Drugs.

While 3D pre-training can yield large improvements, the MAE is still substantially higher than that of SMP where 3D information is assumed to be available for all molecules. For this comparison, we note that QM9's properties are conformer-specific. There might be a maximum accuracy that can be

Table 1: Shown is the MAE for predicting the properties of one half of QM9's molecules. We 3D pre-train either with GEOM-***Drugs***, ***QMugs*** or a disjoint half of ***QM9*** as 3D datasets. ***Rand Init*** refers to a randomly initialized baseline. The pre-trained baselines are ***GraphCL*** and predicting properties (***PropPred***). ***SMP*** is a 3D GNN using the test set's 3D coordinates which the other methods do not use. 3D pre-training yields large improvements over conventional SSL pre-training (GraphCL).

		pre-training baselines		our 3D pre-training			*full 3D*
target	Rand Init	GraphCL	PropPred	QM9	Drugs	QMugs	*SMP*
μ	0.4133±0.003	0.3937	0.3975	**0.3507**	**0.3512**	0.3668	0.0726
α	0.3972±0.014	0.3295	0.3732	0.3268	0.2959	**0.2807**	0.1542
$homo$	82.10±0.33	79.57	93.11	**68.96**	70.78	70.77	56.19
$lumo$	85.72±1.62	80.81	99.84	**69.51**	71.38	78.10	43.58
gap	123.08±3.98	120.08	131.99	**101.71**	102.59	103.85	85.10
$r2$	22.14±0.21	21.84	29.21	**17.39**	18.96	18.00	1.51
$ZPVE$	15.08±2.83	12.39	11.17	**7.966**	9.677	12.06	2.69
c_v	0.1670±0.004	0.1422	0.1795	0.1306	0.1409	**0.1208**	0.0498

achieved if only the molecule is known and not for which specific conformer the property should be predicted. Nevertheless, the performance gap suggests that there is still room for improvement and we discuss what we deem the most promising direction in the Conclusion.

The results in Table 2 further confirm that 3D pre-training substantially improves quantum property predictions. Our method wins out on GraphCL no matter what dataset is used for pre-training. This is the case even though GraphCL was pre-trained on all molecules of GEOM-Drugs while the 3D methods have never seen the fine-tuning data during pre-training.

Table 2: The MAE for GEOM-Drugs' properties. 3D pre-training on ***QM9***, GEOM-***Drugs***, or ***QMugs*** yields significant improvements.

Method	Gibbs	$\langle E \rangle$
Rand Init	.2035±.0011	.1026
GraphCL	.1941	.0995
QM9	.1852	.0968
Drugs	**.1811**	**.0952**
QMugs	.1835	.0965

The in-distribution transfer where we pre-train on one half of GEOM-Drugs and fine-tune on the other again leads to the lowest MAE. However, we see strong generalization in the improvement from .2035 MAE to .1852 when pre-training with QM9 and fine-tuning on GEOM-Drugs. In this case, the pre-training data only contains the atoms C, H, N, O, and F while the target data, to which the method generalizes, contains 11 additional atoms that are never seen during pre-training.

We explain these consistent improvements independent from the pre-training dataset with the type of information that is captured by 3D pre-training. Conventional pre-training relies on augmentations (such as removing atoms) that significantly alter the molecules while assuming that their properties do not change. It is not clear that the learned representations should be informative for downstream tasks. Meanwhile, 3D pre-training teaches the model an understanding of molecular dynamics and the underlying spatial relations in a molecule. Such knowledge is transferable across datasets since it is the foundation of how molecules and atoms interact.

4 Conclusion

Our 3D pre-training method enables a GNN to generate latent 3D and quantum information from 2D molecular graphs which can then be used during fine-tuning to improve molecular property predictions. We found consistently large improvements for quantum properties, overshadowing the gains possible with conventional SSL methods. Furthermore, the learned 3D knowledge can be transferred across highly different types of molecules since the representations capture a principled form of information that is known to be useful for molecular tasks.

Future Work Likely the main bottleneck of our method is that the mutual information is maximized partly by the 3D network encoding 2D information instead of the 2D network generating 3D information. To prevent this we are exploring regularization terms to guarantee that the representations mainly capture 3D information, e.g., via reconstructing all atom distances from them.

References

[1] S. Axelrod and R. Gomez-Bombarelli. "GEOM: Energy-annotated molecular conformations for property prediction and molecular generation". In: *arXiv preprint arXiv:2006.05531* (2020).

[2] T. Chen, S. Kornblith, M. Norouzi, and G. E. Hinton. "A Simple Framework for Contrastive Learning of Visual Representations". In: *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*. Vol. 119. Proceedings of Machine Learning Research. PMLR, 2020, pp. 1597–1607.

[3] G. Corso, L. Cavalleri, D. Beaini, P. Liò, and P. Veličković. "Principal Neighbourhood Aggregation for Graph Nets". In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin. Vol. 33. Curran Associates, Inc., 2020, pp. 13260–13271.

[4] P. O. Dral. "Quantum Chemistry in the Age of Machine Learning". In: *The Journal of Physical Chemistry Letters* 11.6 (2020). PMID: 32125858, pp. 2336–2347. DOI: 10.1021/acs.jpclett.9b03664. eprint: https://doi.org/10.1021/acs.jpclett.9b03664.

[5] M. Fey and J. E. Lenssen. "Fast Graph Representation Learning with PyTorch Geometric". In: *ICLR Workshop on Representation Learning on Graphs and Manifolds*. 2019.

[6] O.-E. Ganea, L. Pattanaik, C. W. Coley, R. Barzilay, K. F. Jensen, W. H. Green, and T. S. Jaakkola. *GeoMol: Torsional Geometric Generation of Molecular 3D Conformer Ensembles*. 2021. arXiv: 2106.07802 [physics.chem-ph].

[7] C. Isert, K. Atz, J. Jiménez-Luna, and G. Schneider. *QMugs: Quantum Mechanical Properties of Drug-like Molecules*. 2021. arXiv: 2107.00367 [physics.chem-ph].

[8] J. Jumper, R. Evans, A. Pritzel, T. Green, M. Figurnov, O. Ronneberger, K. Tunyasuvunakool, R. Bates, A. Žídek, A. Potapenko, A. Bridgland, C. Meyer, S. A. A. Kohl, A. J. Ballard, A. Cowie, B. Romera-Paredes, S. Nikolov, R. Jain, J. Adler, T. Back, S. Petersen, D. Reiman, E. Clancy, M. Zielinski, M. Steinegger, M. Pacholska, T. Berghammer, S. Bodenstein, D. Silver, O. Vinyals, A. W. Senior, K. Kavukcuoglu, P. Kohli, and D. Hassabis. "Highly accurate protein structure prediction with AlphaFold". In: *Nature* 596.7873 (Aug. 2021), pp. 583–589. ISSN: 1476-4687. DOI: 10.1038/s41586-021-03819-2.

[9] J. Klicpera, F. Becker, and S. Günnemann. *GemNet: Universal Directional Graph Neural Networks for Molecules*. 2021. arXiv: 2106.08903 [physics.comp-ph].

[10] J. Klicpera, J. Groß, and S. Günnemann. "Directional Message Passing for Molecular Graphs". In: *International Conference on Learning Representations (ICLR)*. 2020.

[11] Y. Liu, L. Wang, M. Liu, X. Zhang, B. Oztekin, and S. Ji. "Spherical Message Passing for 3D Graph Networks". In: *CoRR* abs/2102.05013 (2021). arXiv: 2102.05013.

[12] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng. "NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis". In: *ECCV*. 2020.

[13] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. "Automatic differentiation in PyTorch". In: (2017).

[14] N. Rahaman, A. Baratin, D. Arpit, F. Draxler, M. Lin, F. Hamprecht, Y. Bengio, and A. Courville. "On the Spectral Bias of Neural Networks". In: *Proceedings of the 36th International Conference on Machine Learning*. Ed. by K. Chaudhuri and R. Salakhutdinov. Vol. 97. Proceedings of Machine Learning Research. PMLR, June 2019, pp. 5301–5310.

[15] R. Ramakrishnan, P. O. Dral, M. Rupp, and O. A. von Lilienfeld. "Quantum chemistry structures and properties of 134 kilo molecules". In: *Scientific Data* 1.1 (Aug. 2014), p. 140022. ISSN: 2052-4463. DOI: 10.1038/sdata.2014.22.

[16] V. G. Satorras, E. Hoogeboom, and M. Welling. "E(n) Equivariant Graph Neural Networks". In: *CoRR* abs/2102.09844 (2021). arXiv: 2102.09844.

[17] J. Schmidt, M. R. G. Marques, S. Botti, and M. A. L. Marques. "Recent advances and applications of machine learning in solid-state materials science". In: *npj Computational Materials* 5.1 (Aug. 2019), p. 83. ISSN: 2057-3960. DOI: 10.1038/s41524-019-0221-0.

[18] K. Schütt, P.-J. Kindermans, H. E. S. Felix, S. Chmiela, A. Tkatchenko, and K.-R. Müller. "SchNet: A continuous-filter convolutional neural network for modeling quantum interactions". In: *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*. Ed. by I. Guyon, U. von Luxburg, S. Bengio, H. M. Wallach, R. Fergus, S. V. N. Vishwanathan, and R. Garnett. 2017, pp. 991–1001.

[19] C. Shi, S. Luo, M. Xu, and J. Tang. "Learning Gradient Fields for Molecular Conformation Generation". In: *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*. Ed. by M. Meila and T. Zhang. Vol. 139. Proceedings of Machine Learning Research. PMLR, 2021, pp. 9558–9568.

[20] J. M. Stokes, K. Yang, K. Swanson, W. Jin, A. Cubillos-Ruiz, N. M. Donghia, C. R. MacNair, S. French, L. A. Carfrae, Z. Bloom-Ackermann, V. M. Tran, A. Chiappino-Pepe, A. H. Badran, I. W. Andrews, E. J. Chory, G. M. Church, E. D. Brown, T. S. Jaakkola, R. Barzilay, and J. J. Collins. "A Deep Learning Approach to Antibiotic Discovery". In: *Cell* 180.4 (2020), 688–702.e13. ISSN: 0092-8674. DOI: `https://doi.org/10.1016/j.cell.2020.01.021`.

[21] M. Tancik, P. P. Srinivasan, B. Mildenhall, S. Fridovich-Keil, N. Raghavan, U. Singhal, R. Ramamoorthi, J. T. Barron, and R. Ng. "Fourier Features Let Networks Learn High Frequency Functions in Low Dimensional Domains". In: *NeurIPS* (2020).

[22] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. "Attention is All you Need". In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett. Vol. 30. Curran Associates, Inc., 2017.

[23] M. Wang, D. Zheng, Z. Ye, Q. Gan, M. Li, X. Song, J. Zhou, C. Ma, L. Yu, Y. Gai, T. Xiao, T. He, G. Karypis, J. Li, and Z. Zhang. "Deep Graph Library: A Graph-Centric, Highly-Performant Package for Graph Neural Networks". In: *arXiv preprint arXiv:1909.01315* (2019).

[24] M. Xu, S. Luo, Y. Bengio, J. Peng, and J. Tang. "Learning Neural Generative Dynamics for Molecular Conformation Generation". In: *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021.

[25] K. Yang, K. Swanson, W. Jin, C. Coley, P. Eiden, H. Gao, A. Guzman-Perez, T. Hopper, B. Kelley, M. Mathea, A. Palmer, V. Settels, T. Jaakkola, K. Jensen, and R. Barzilay. "Analyzing Learned Molecular Representations for Property Prediction". In: *Journal of Chemical Information and Modeling* 59.8 (2019). PMID: 31361484, pp. 3370–3388. DOI: `10.1021/acs.jcim.9b00237`. eprint: `https://doi.org/10.1021/acs.jcim.9b00237`.

[26] Y. You, T. Chen, Y. Sui, T. Chen, Z. Wang, and Y. Shen. "Graph Contrastive Learning with Augmentations". In: *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*. Ed. by H. Larochelle, M. Ranzato, R. Hadsell, M.-F. Balcan, and H.-T. Lin. 2020.

[27] E. D. Zhong, T. Bepler, J. H. Davis, and B. Berger. "Reconstructing continuous distributions of 3D protein structure from cryo-EM images". In: *International Conference on Learning Representations*. 2020.

A Method Details

A.1 Multiple Conformers

For each molecule $(G_i, \{R_i^j\}_{j\in\{1\dots c_i\}})$ we choose the c highest energy conformers to have a fixed number of them. If there are fewer than c conformers for a molecule ($c_i < c$) then the lowest energy conformer is repeated. For every molecule the 3D network takes all c conformers $\{R_i^j\}_{j\in\{1\dots c\}}$ as input and produces their latent 3D representations $\{z_{i,j}^b\}_{j\in\{1\dots c\}}$ which we can see as additional positive samples.

A.2 3D Network

Here we describe the 3D network in greater detail and call it *Net3D*. It encodes the 3D information given by the pairwise Euclidean distances of atoms. Therefore, the method is $E(3)$ invariant, and the positions of all atoms are uniquely defined up to $E(3)$ symmetry. Of course, using all pairwise ℓ_2-distances also means that the method's complexity is quadratic in the number of atoms in the molecule.

Net3D can be seen as a GNN operating on the complete graph of each molecular graph. Given a molecule (G, R), all the pairwise distances $\{d_{uv} = \|r_u - r_v\|_2 \mid u, v \in \mathcal{V} \wedge u \neq v\}$ are first mapped to a higher dimensional space using high-frequency functions and those encodings are used as input

to the network with learnable weights. This is motivated by (1) the fact that there are only small variations in distances for atoms that are connected by bonds and (2) the findings of Rahaman et al. 2019 deep neural networks optimized with stochastic gradient descent having a bias towards learning lower frequency functions. They show that using mappings as described leads to deep networks better fitting data with high-frequency variation which is further supported by Tancik et al. 2020 who show empirically and theoretically that multi-layer perceptrons (MLPs) fail to learn high frequencies. This scenario is present in our case with small differences in bond lengths. Additionally, these distances and their small variations might be the most important ones with the assumption that atom pairs that have small distances have the most relevant interactions.

The mapping $\gamma : \mathbb{R} \mapsto \mathbb{R}^{2F+1}$ which we use before passing the 3D information to MLPs is defined as

$$\gamma(d_{uv}) = (d_{uv},\, sin(d_{uv}/2^0),\, cos(d_{uv}/2^0),\, \ldots,\, sin(d_{uv}/2^{F-1}),\, cos(d_{uv}/2^{F-1})), \tag{2}$$

where the number of frequencies F is a hyperparameter that is set to 4 in our experiments. This is inspired by the positional encoding as it is used in Transformers Vaswani et al. 2017 with the different purpose of providing an ordering in a set. With a similar purpose as in our application, this strategy was successfully previously used by Mildenhall et al. 2020 in image synthesis and by Zhong et al. 2020 in a method for inferring the 3D structure of proteins from projections.

The l-th layer of Net3D takes two sets as input. First, $n^2 - n$ edge representations $\{d^l_{uv} \in \mathbb{R}^{d_d} \mid u, v \in \mathcal{V} \wedge u \neq v\}$ (the edges of a complete graph without self-loops). In the first layer they are given by the encoded distances fed through an initial feed-forward network $MLP_{init} : \mathbb{R}^{2F+1} \mapsto \mathbb{R}^{d_d}$ which projects them to the hidden dimension of the edges $d^0_{uv} = MLP_{init}(\gamma(d_{uv}))$. The second input is a set of n atom representations $\{h^l_1, \ldots h^l_n\}$ with dimensionality $\mathbb{R}^{d_h}$. In the first layer, the atom representations are all set to the same learned vector that is initialized with a standard normal. With $\|$ meaning concatenation, every layer updates the edge and atom representations and iteratively encodes 3D information into them as follows:

$$m_{uv} = MLP_{edge}([h^l_u \parallel h^l_v \parallel d^l_{uv}]) \tag{3}$$

$$d^{l+1}_{uv} = d^l_{uv} + m_{uv} \tag{4}$$

$$h^{l+1}_u = MLP_h([h_u \parallel \sum_{\substack{v=1 \\ v \neq u}}^{n} m_{uv} * sigmoid(MLP_{softedge}(m_{uv})]). \tag{5}$$

The layer is parameterized by three MLPs where the first one updates the edges $MLP_{edge} : \mathbb{R}^{2d_h+d_d} \mapsto \mathbb{R}^{d_d}$. The second one updates the atom representations $MLP_h : \mathbb{R}^{d_h+d_d} \mapsto \mathbb{R}^{d_h}$. The third one $MLP_{softedge} : \mathbb{R}^{d_d} \mapsto \mathbb{R}$ is followed by the logistic sigmoid function to create a value between 0 and 1 that can be seen as a soft edge weight telling us how probable an edge is for each message m_{uv} as it is done by Satorras et al. 2021.

To produce the final 3D representation z^b, all atom representations are aggregated by concatenating their mean, their maximum, and their standard deviation and feeding this through a final feed-forward network $MLP : \mathbb{R}^{3d_h} \mapsto \mathbb{R}^{d_z}$.

B Explanatory Illustrations

In Figure 2 and Figure 3 we provide additional illustrations for our proposed method.

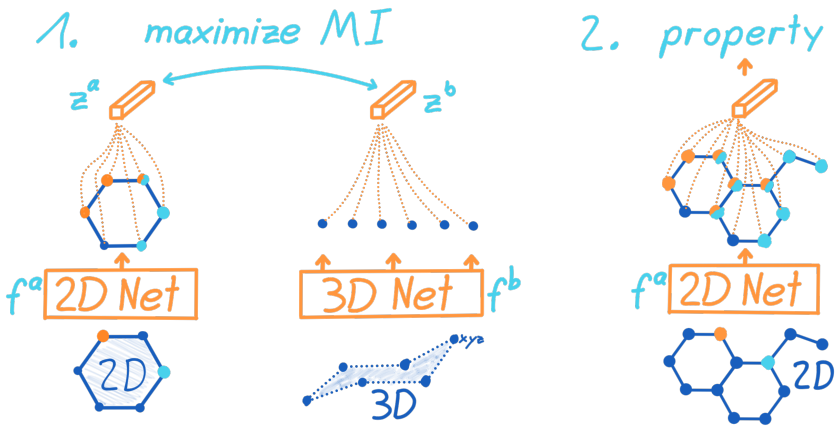

Figure 2: General overview of our latent space SSL methods. In step 1. we pre-train a 2D network by forcing it to generate a 2D representation z^a that has high mutual information with a 3D representation z^b coming from an encoder that has only 3D information as its input. After the 2D model has learned to generate latent 3D information through this pre-training, we transfer its weights and fine-tune it to predict molecular properties for molecules where we do not have the 3D information available.

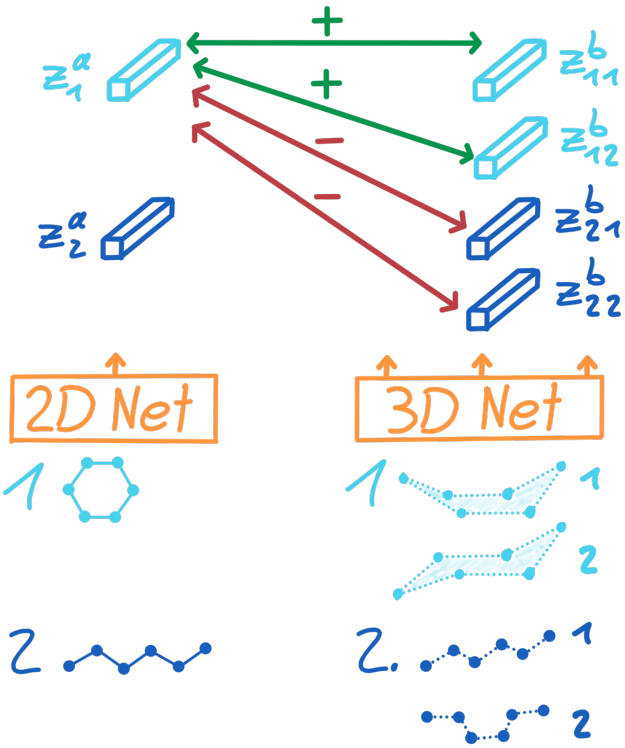

Figure 3: Explanatory illustration for contrastive learning with multiple conformers. This example displays a batch of two molecules with their 2D graphs on the left and the corresponding 3D conformers on the right where each molecule has two plausible conformers. To learn a joint embedding space the contrastive loss enforces high similarity between latent representations that come from the same molecule while encouraging dissimilarity if the latent representations belong to different molecules in the batch. Here this is depicted for the first molecule but the same loss is calculated for the second. The final loss is the average of those two.

C Experimental Details

The standard deviation of energy of the highest occupied molecular orbital (homo) and the Gibbs free energy is estimated with six seeds, the rest uses four.

C.1 Data Details

The pre-training datasets which we use are the following:

1. **QM9**[1] (Ramakrishnan et al. 2014) contains 134k stable small organic molecules of 5 atom types (CHONF). Every molecule has the 3D coordinates of one low-energy conformer and is annotated with 12 quantum mechanical properties as regression targets. The molecules are considered very small, with at most 9 heavy atoms.
2. **GEOM-Drugs**[2] (Axelrod and Gomez-Bombarelli 2020) consists of 304k realistically-sized biologically and pharmacologically relevant molecules of 16 atom types, annotated with multiple 3D conformers, the ensemble Gibbs free energy, and the ensemble energy as regression targets. For the average molecule, 70% of the Boltzmann weight is captured by just three conformers as can be seen in Figure 4 where we also provide a histogram for the number of molecules that have a certain amount of conformers5.
3. **QMugs**[3] (Isert et al. 2021) has 665k drug-like molecules with three diverse conformers each and multiple conformer specific quantum mechanical properties as regression tasks.

Table 3: Statistics of the used datasets. In the upper section are datasets with 3D information which we use for pre-training and the datasets in the bottom section do not contain additional 3D annotations.

Dataset	#Molecules	Avg. #Atoms	Avg. #Bonds	split
`QM9`	130 831	18.0	18.6	random
`GEOM-Drugs`	304 293	44.4	46.4	random
`QMugs`	665 911	30.6	33.4	random

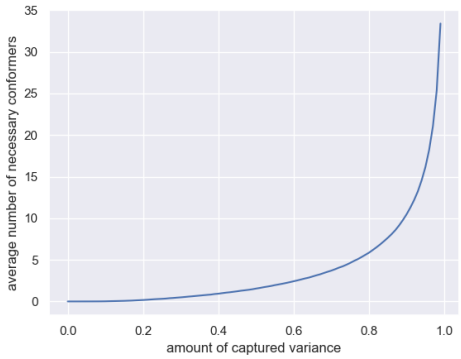

Figure 4: Average number of conformers necessary to cover a certain amount of Boltzmann weight in GEOM-Drugs. For a given amount of cumulative Boltzmannweight on the horizontal axis, the vertical axis shows the average number of conformers necessary to pass that threshold.

C.2 Units

For the GEOM-Drugs dataset, all reported numbers have the unit kcal/mol, Gibbs refers to the ensemble Gibbs free energy, and $\langle E \rangle$ to the ensemble energy.

C.3 Conventional pre-training Baseline GraphCL

The conventional pre-training method that we compare against is GraphCL (You et al. 2020) since it is the work with the strongest results in a consistent comparison for molecular tasks. GraphCL uses

[1] `https://github.com/klicperajo/dimenet/blob/master/data/qm9_eV.npz`
[2] `https://github.com/learningmatter-mit/geom`
[3] `https://www.research-collection.ethz.ch/handle/20.500.11850/482129`

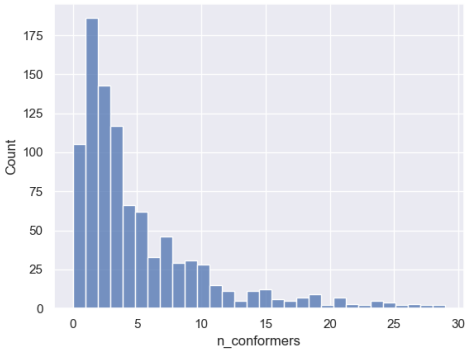

Figure 5: Histogram of how many molecules there are in GEOM-Drugs with a certain amount of conformers. The histogram is created for 1000 molecules of GEOM-Drugs.

Table 4: Units of Quantum mechanical properties.

μ	α	$homo$	$lumo$	gap	$r2$	$ZPVE$	c_v
D	a_0^3	meV	meV	meV	a_0^2	meV	$\frac{cal}{molK}$

the usual self-supervised objective where the model has to learn to produce representations that are invariant with respect to augmentations. You et al. 2020 evaluated different such augmentations and in our baseline, we use randomly dropping nodes since they found strong results on molecular tasks with this augmentation. We use their node-drop ratio of 0.2 and the only difference of our setup is that we use a batch size of 500 which we found to work better in our setting instead of their 256.

C.4 Parameter Details

The hyperparameters for SMP are taken from the official repository[4] where Liu et al. 2021 provide their code and we predict gap even though it could be calculated as $|homo - lumo|$. The parameter search space and final parameters for the PNA architecture is specified in Table 5 and those of Net3D in Table 6.

Pre-training: We use Adam with a start learning rate of 8×10^{-5} and a batch size of 500. The learning rate schedule during pre-training starts with 700 optimization steps of linear warmup followed by the schedule given by the *ReduceLROnPlateau* scheduler by PyTorch[5] with reduction parameter 0.6, patience 25 and a cooldown of 20.

Fine-tuning quantum mechanical properties: We use Adam with a start learning rate of 7×10^{-5}, weight decay 1×10^{-11} and a batch size of 128. For the learning rate schedule, we first perform warmup as follows. We consider three different sets of learnable parameters: (1) the batch norm parameters, (2) all newly initialized parameters that were not transferred, and (3) all parameters. For these sets, we increase the learning rate in this order from 0 to the start learning rate with linear interpolation. For parameter group one we warm-up for 700 steps, 700 steps for group 2, and 350 steps for group 3. After that we use the schedule given by the *ReduceLROnPlateau* with reduction parameter 0.5, patience 25 and a cooldown of 20.

Fine-tuning biophysical properties: We use Adam with a start learning rate of 1×10^{-3} and a batch size of 32. The learning rate schedule is the same as for the quantum mechanical properties.

C.5 Implementation

All experiments were implemented in *PyTorch* (Paszke et al. 2017) using the deep learning libraries for processing graphs *Pytorch Geometric* (Fey and Lenssen 2019) and *Deep Graph Library* (Wang et al. 2019).

[4] `https://github.com/divelab/DIG`

[5] `https://pytorch.org/docs/stable/generated/torch.optim.lr_scheduler.ReduceLROnPlateau.html`

Table 5: Search space for the 2D network PNA through which we searched to obtain a strong baseline performance on the homo property of the QM9 dataset. The parameters were tuned in the order in which they are listed in this table from top to bottom. After this was completed for all parameters we performed a second round of tuning for a subset of them. The final parameters are marked in **bold**.

Parameter	Search Space
propagation depth	[4, 5, 6 ,**7**]
hidden dimension	[40, 50, 75, 90, 100, 150, **200** ,300]
message MLP layers	[1, **2**, 3]
update MLP layers	[**1**, 2, 3]
aggregators	[mean, max, min, std, sum], [mean, max, min], [mean, max, sum], **[mean, max, min, std]**, [max, sum], [sum]
scalers	[identity], **[identity, amplification, attenuation]**
readout aggregators	[mean], [sum], [mean, max, sum], **[mean, max, min, sum]**
dropout	[**0**, 0.05, 0.1, 0.2]
batchnorm after MLPs	True/False
batchnorm in MLPs	True/False
readout MLP layers	[1, **2**, 3]
batchnorm momentum	[**0.1**, 0.9, 0.93]

Table 6: Search space for the 3D network Net3D through which we searched to obtain a strong baseline performance on the homo property of the QM9 dataset and we considered the size of the network where parameters leading to less memory use are preferred. The parameters were tuned in the order in which they are listed in this table from top to bottom. After this was completed for all parameters we performed a second round of tuning for a subset of them. The final parameters are marked in **bold**.

Parameter	Search Space
propagation depth	[**1**, 3, 4, 5]
hidden dimension	[10, **20**, 40, 60, 80, 100]
F used in $\gamma : \mathbb{R} \mapsto \mathbb{R}^{2F+1}$	[0, 3, **4**, 8, 10, 50]
message MLP layers	[**1**, 2, 3]
update MLP layers	[**1**, 2, 3]
readout aggregators	[mean], [sum], **[mean, max, min]**, [mean, max, min, sum]
dropout	[**0**, 0.05, 0.1, 0.2, 0.5]
batchnorm after MLPs	True/False
readout MLP layers	[**1**, 2, 3]
batchnorm momentum	[0.1, 0.9, **0.93**]

The experiments were conducted on two different machines while the same system was always used in direct comparisons. The first machine has an AMD Ryzen 1700 CPU @ 3.70Ghz, 16GB of RAM, and an Nvidia GTX 1060 GPU with 6GB vRAM. The second system contains two Intel Xeon Gold 6248 CPUs @ 2.50GHz each with 20/40 cores, 400GB of RAM, and four Quadro RTX 8000 GPUs with 46GB vRAM of which only a single one was used for each experiment. All mentions and of training time refer to the second system.