DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Informatics

Self-Supervised learning for small Molecular Graphs

Hannes Stärk

DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Informatics

Self-Supervised learning for small Molecular Graphs

Self-Supervised learning für kleine Molekulare Graphen

Author:	Hannes Stärk
Supervisor:	Prof. Dr. Stephan Günnemann
Advisors:	Prof. Dr. Pietro Liò, Dr. Dominique Beaini, Dr. Prudencio Tossou
Submission Date:	15.09.2021

I confirm that this master's thesis in informatics is my own work and I have documented all sources and material used.

Munich, 15.09.2021 Hannes Stärk

Acknowledgments

I would like to thank my supervisors Pietro Liò and Stephan Günnemann, for their exceptional guidance and incredible support. Additionally, I feel fortunate to have been working in such a great team with Dominique Beaini, Gabriele Corso, Prudencio Tossou, and Christian Dallago, who provided valuable advice, feedback, discussions, ideas, and compute resources.

Apart from this core team, I want to express my gratitude to Aleksandar Bojchevski for his tips while writing the thesis and to all the researchers that discussed their papers, their research, my naive questions, and my research projects with me. Here I especially point out Simon Axelrod (GEOM Dataset and molecular physics advice); Limei Wang and Meng Liu (Spherical Message Passing and 3D GNNs); Adrien Bardes and Samuel Lavoie (Self-Supervised learning, VICReg, and BYOL); Octavian Ganea and Lagnajit Pattanaik (molecular geometry, GeoMol). Thank you for making me think and have ideas.

Moreover, I thank the other researchers who repeatedly took the time to discuss these topics with me to come up with great ideas together: Zekarias Kefato, Grégoire Mialon, Kenneth Atz, Minghao Xu, Muhammet Balcilar, Mozhi Zhang, Shantanu Thakoor, Minkai Xu, Devin Kreuzer, Shitong Luo, Yuning You, Prannay Khosla.

Besides that, I thank everyone else who agreed to discuss research with me. I wish to acknowledge that I would not have had most of my ideas and not nearly as much fun without these great conversations.

Abstract

Molecular property prediction is one of the fastest-growing applications of deep learning with some of the most important real-world impacts. While it is clear that 3D information is valuable to improve tasks on molecules, there are many molecules for which the geometry is not available. However, in principle, all information about a molecule's possible 3D configurations is contained in its representation as a molecular graph. Can a property prediction model be improved by understanding a molecule's geometry from only the molecular graph?

This thesis shows this to be the case. Using methods from self-supervised learning (SSL) we 3D pre-train a Graph Neural Network (GNN) to generate latent 3D information. During fine-tuning, the model generates implicit 3D representations and uses them to inform downstream molecular property predictions, improving their accuracy.

Multiple other methods to pre-train models for molecular tasks have been proposed previously. They rely on augmentations that significantly alter the molecules while assuming that their properties do not change. It is not clear whether the learned representations should be informative for downstream tasks, and it is questionable how well they generalize. Meanwhile, for 3D pre-training, we know that the features will contain general and helpful information.

In this thesis, we develop six methods for 3D pre-training, discuss their theoretical abilities to encode 3D information, and extend them to leverage multiple molecular conformers. We extensively empirically study variations of the approaches and analyze their representations.

The experimental results show that our 3D pre-training can provide large improvements for a wide range of molecular properties. Crucially, even in cases where increases are marginal, 3D pre-training always provides improvements or performs on par, unlike many prior SSL methods for molecules that suffer from negative transfer for some tasks. Lastly, the learned representations are very generalizable and can be transferred between datasets with vastly different molecules. All these qualities of our method are essential for real-world applications and make it highly interesting for practice.

Kurzfassung

Das Vorhersagen der Eigenschaften von Molekülen ist eine der am schnellsten wachsenden Anwendungen von Deep Learning und liefert einige der wichtigsten Errungenschaften des Feldes. Es ist klar, dass Informationen über die 3D-Struktur von Molekülen wertvoll sind und deutlich bessere Vorhersagen ermöglichen. Allerdings sind für viele Moleküle keine 3D Informationen vorhanden. Zusätzlich sind im Prinzip wiederum alle Informationen über die möglichen 3D Geometrien eines Moleküls in seiner Repräsentation als molekularer Graph enthalten. So stellt sich die Frage: Kann ein Neuronales Netz für Moleküle verbessert werden, wenn es nur von molekularen Graphen ein Verständnis über die Geometrie der Moleküle erlangen kann?

Diese Arbeit zeigt, dass dem so ist. Mit Self-Supervised Learning (SSL) Methoden trainieren wir ein Graph Neuronales Netzwerk (GNN), um latente 3D Informationen zu generieren. Wenn das Modell dann für andere Anwendungen weiter trainiert wird, kann es diese Informationen nutzenn um seine Vorhersagen zu verbessern.

Zuvor wurden schon einige andere Methoden für das Pre-training von GNNs für Moleküle vorgeschlagen. Diese nutzen immer Augmentationen, welche Moleküle signifikant verändern, aber gehen trotzdem davon aus, dass die Moleküle ihre Eigenschaften beibehalten. Es ist nicht klar warum die so gelernten Repräsentationen informativ für Endanwendungen sein sollten und es ist fragwürdig wie gut sie generalisieren. Mit dem 3D Pre-training Ansatz ist dies nicht der Fall und es ist evident, dass die Informationen in den gelernten Repräsentationen informativ sind.

In dieser Arbeit entwickeln wir sechs Methoden für 3D Pre-training, diskutieren ihre theoretischen Fähigkeiten 3D Informationen zu enkodieren, und erweitern sie, sodass sie mehrere molekulare Konformationen nutzen können. Mit verschiedenen Versionen der Methode führen wir extensive empirische Studien durch und analysieren ihre Repräsentationen.

Die experimentellen Ergebnisse zeigen, dass unser 3D Pre-training große Verbesserungen für viele verschiedene molekulare Eigenschaften liefern kann. Auch in Fällen, in denen nur marginale Verbesserungen zu sehen sind, handelt es sich immer um Verbesserungen anders als bei vielen vorherigen SSL Methoden die für manche Aufgaben Verschlechterungen zeigten durch das Pre-training. Außerdem generalisieren die gelernten Repräsentationen äußerst stark und können zwischen Datensätzen mit sehr verschiedenen Molekülen transferiert werden. All diese Eigenschaften sind wichtig für tatsächliche Anwendungen und machen unsere Methode interessant für die Praxis.

Contents

1. Introduction

Advances in computer hardware and software have led to rapid progress in deep learning such that it heavily impacts and drives fields ranging from computer vision over natural language processing to healthcare. One area that has recently seen a rise in interest is applying deep learning research to molecules. Learned models for molecules are beyond the stage of only research and are having a real-world impact in quantum chemistry, protein prediction, materials science, or drug discovery (Dral 2020; Jumper, Evans, Pritzel, et al. 2021; Schmidt, M. R. G. Marques, Botti, and M. A. L. Marques 2019; Stokes, K. Yang, Swanson, et al. 2020).

In particular, associated with the advent of GNNs, there have been many successes in molecular property prediction (Walters and Barzilay 2021a). To use GNNs for molecules, they are represented as graphs with atoms as nodes and atomic bonds as edges. These methods yield strong results using the molecular graph since it provides a helpful locality bias - messages are passed between those atoms that probably influence each other the most.

However, how atoms and molecules actually physically interact does not only depend on our concept of atomic bonds. It is rather determined by the atoms' relative positions in space. Therefore, GNNs that use 3D graphs where each atom has an additional 3D coordinate, have often produced much better molecular property predictions - sometimes with errors that are orders of magnitude lower (Klicpera, Groß, and Günnemann 2020; Klicpera, Giri, Margraf, and Günnemann 2020; Y. Liu, L. Wang, M. Liu, et al. 2021; Satorras, Hoogeboom, and Welling 2021; Klicpera, Becker, and Günnemann 2021).

We know that 3D information is important for many molecular properties. However, for many molecules, it is not available. Often it is too expensive, time-consuming, or completely computationally infeasible to infer the geometry of a molecule with classical methods. This raises the question: Can we generate the 3D information with learned methods and then use it to improve downstream molecular property predictions? This way, we could leverage 3D datasets to make better predictions for other molecules with unknown geometry.

One idea to achieve this would be using learned methods for molecular conformation generation (M. Xu, S. Luo, Bengio, et al. 2021; Shi, S. Luo, M. Xu, and J. Tang 2021; Ganea, Pattanaik, Coley, et al. 2021) to produce explicit 3D coordinates that serve as additional input to a 3D GNN. While these learned approaches are rapidly developing, it stands to reason that their accuracy is high enough and often they are not fast either.

However, we do not have to produce explicit 3D information. Instead, a GNN could be pre-trained with SSL approaches to generate latent 3D information from molecular graphs. After pre-training, the weights can then be transferred and fine-tuned on the data where no 3D information is available. For those molecules, the GNN still generates the implicit 3D information and during fine-tuning, the network can use the latent information to inform its property predictions.

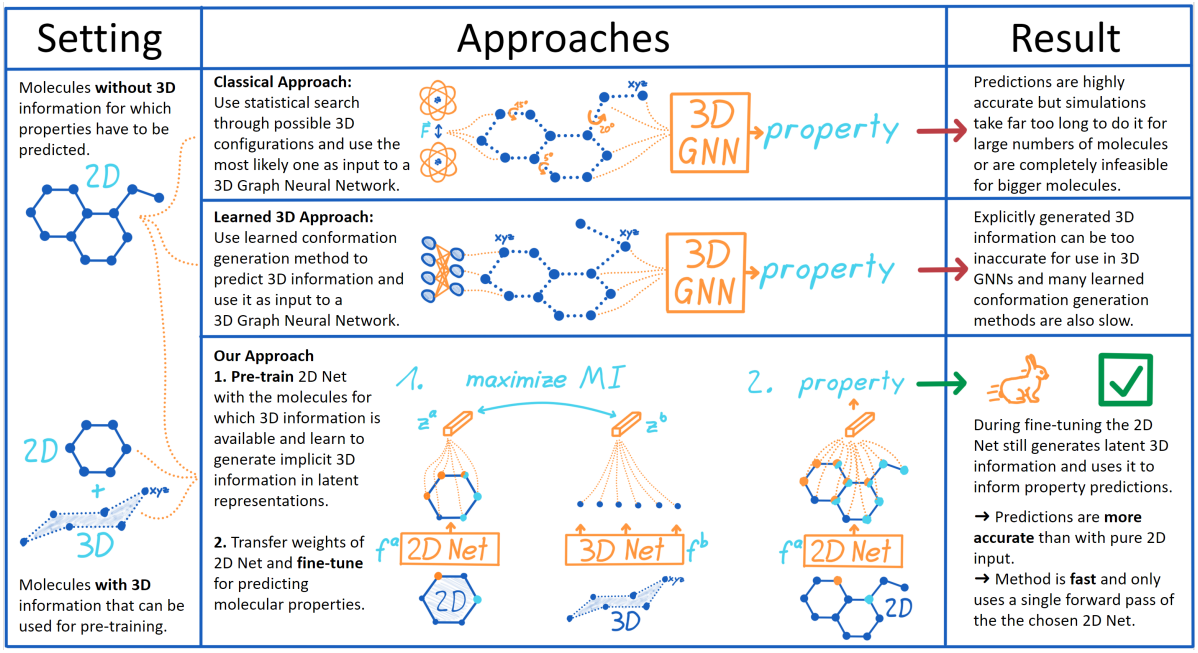


Figure 1.1.: The considered problem setting and the motivation for our pre-training approach.

Separate from 3D information, many other SSL methods have been proposed and evaluated to pre-train GNNs and obtain better property predictions after fine-tuning (W. Hu, B. Liu, Gomes, et al. 2020; You, T. Chen, Sui, et al. 2020; M. Xu, H. Wang, Ni, et al. 2021). They often rely 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 and it is questionable how well they generalize. Meanwhile, for 3D pre-training, we know that the features will contain generalizable and useful information if 3D knowledge is indeed captured in the representations.

The goal of this thesis is to successfully perform 3D pre-training for a GNN such that its weights can be fine-tuned on a variety of tasks to solve them with higher accuracy. We want to find the best approach to do so while evaluating how big of an improvement can be made in different settings and for what types of molecular property predictions it helps the most.

To do so we devise multiple 3D pre-training methods. In the first kind, the GNN predicts some explicit form of 3D information such as the pairwise distances of all atoms. Some amount of the GNN's layers are then transferred and fine-tuned for property prediction.

The second kind heavily draws from the field of SSL and contrastive learning. During pre-training, the GNN generates a latent representation of a molecular graph. At the same time, another network generates a representation of the 3D information. The training objective is to maximize the mutual information that is shared by both representations which can be achieved with methods from SSL. This approach forces the GNN to learn to generate latent 3D information only from molecular graphs. The 3D information can then be used during fine-tuning to inform property predictions.

We extensively study our methods by analyzing their latent representations and by pre-training them with three different 3D datasets before evaluating on 10 quantum mechanical molecular properties and 10 datasets with biophysical properties. We find that 3D pre-training can improve molecular property predictions by large margins, especially for quantum mechanical properties.

Furthermore, the learned representations are highly generalizable and big improvements are possible even if the 3D pre-training dataset's molecules are vastly different (e.g., in size) from the downstream molecules for which no 3D information is available. While conventional pre-training methods for molecules sometimes suffer from negative transfer (S. J. Pan and Q. Yang 2010) where performance decreases after pre-training, 3D pre-training does not have this issue in any of our evaluations. Another contribution of this work is a generalization of the SSL method Bootstrap your own latent (BYOL) (Grill, Strub, Altché, et al. 2020) to arbitrary multi-modal inputs.

The thesis is structured as follows: in Chapter 2 we provide the relevant background information about molecules and Machine Learning (ML) for molecules as well as a brief presentation of GNNs and mutual information. Next, we cover the multiple different areas of related work from which we draw our methods and contribute to. This shows in what context the used approaches were developed and we explain how our work distinguishes itself from prior work (see Chapter 3). In Chapter 4 we clearly define the type of data we consider and the mathematical framework into which our SSL methods fit. These methods are then presented, the reasoning behind them is explained, and their theoretical limitations are laid out in Chapter 5. We empirically evaluate them and explore their properties with different quantum mechanical and biophysical molecular property prediction datasets in Chapter 6. Lastly, we summarize our findings and propose future work that we deem important in Chapter 7.

2. Background

This section explains background information that is relevant for understanding this thesis. In Section 2.1 we first cover topics about ML for molecules and their 3D geometry. Next, Section 2.2 briefly explains GNNs and introduces the Message Passing Neural Network (MPNN) framework. Lastly, we provide a quick summary of mutual information and KL-Divergence in ML together with the intuition that is useful to ease the understanding of this thesis.

2.1. Chemistry

To use ML for molecules, they are most commonly represented as strings (SMILES) (Weininger 1988) or as molecular graphs. In this thesis we work with molecular graphs where each atom is seen as a vertex and each atomic bond corresponds to an edge in the graph. Such a graph can then be processed with a GNN to, e.g., produce molecular property predictions. However, to evaluate ML models, a conventional random dataset split is often not a realistic test scenario and a scaffold split may be preferred which Section 2.1.1 explains.

Furthermore, a molecule's 3D information (the relative positions of its atoms) can help for predicting its properties but a single molecule often has multiple 3D configurations which we cover in Section 2.1.2. Lastly, this section explains the symmetries of molecules which are important when using their 3D coordinates in ML (see Section 2.1.3).

2.1.1. Scaffold Split

For molecular data, scaffold splits are often used instead of random data splits to get a better estimate of the generalization capabilities of a method. Since scaffolds represent the core of the molecules to which functional groups are attached, such a split ensures that the training and testing sets have different families of molecules.

When training a machine learning model, we commonly try to estimate the generalization error of the method with a validation and test set that was split from the data. For molecular data, the generalization capability that we are interested in is often the performance when using a method on a different chemical space since this best reflects the situation that is present when deploying the method in practice such as in drug discovery. This means that if we generate our validation and test set via a random split, we are likely to overestimate the generalization power (K. Yang, Swanson, Jin, et al. 2019). With a scaffold split, we try to mimic the distribution shift between chemical spaces.

For instance, models can easily memorize and overfit on the Bemis-Murcko scaffold (Bemis and Murcko 1996) of a molecule, a backbone that is left after removing specific parts of a molecule (see Figure B.1 in Appendix B.2). To account for this and to avoid underestimating

the generalization error, a scaffold split is used to minimize the scaffold overlap between training and test data. In this approach, all molecules are clustered according to their Murcko scaffold and the clusters are randomly assigned to the different data splits until the desired ratio of molecules is reached.

In this thesis, we use both random and scaffold splits to be consistent with the evaluation protocols of previous work and for ease of comparison.

2.1.2. Molecular Conformers

In this work, we pre-train models to learn 3D information and transfer it to downstream molecular property predictions tasks. However, this 3D information is not given by a single set of coordinates since for a given molecular graph there are often multiple conformers, i.e., plausible arrangements of the atoms (Figure 4.1 is illustrative) Different conformations can have different chemical properties.

For a fixed molecular graph, the 3D conformation of the atoms has one degree of freedom for each edge/bond that connects more than an individual atom given by rotating around that bond. As such, the number of conformations grows exponentially with the number of bonds.

Depending on their potential energy, the conformers have a higher probability (lower energy) or lower probability (higher energy) of occurring called the Boltzmann weight. These probabilities p_i are uniquely determined by the energies of the conformers through a Boltzmann distribution described below. Thus, the conformers used for training are those whose potential is a local minimum in the energy landscape of all conformations since they represent the molecular configurations that are most likely to be observed experimentally.

$$p_i = \frac{d_i e^{-E_i/k_B T}}{\sum_j d_j e^{-E_j/k_B T}} \tag{2.1}$$

In equation (2.1), the sum is over all conformers, E_i is the energy of a conformer, T is the temperature, k_B is the Boltzmann constant and d_i is the conformers degeneracy, i.e., the number of equivalent conformations that can be obtained by rotating around bonds (the number of rotamers).

For small molecules, there are many computational methods to determine the conformers that differ in speed and accuracy (Landrum 2016; Chan, Hutchison, and Morris 2019; Hawkins, Skillman, Warren, et al. 2010; Grimme 2019). With these tools, it is computationally feasible to generate sufficiently large datasets for pre-training our methods and in principle, arbitrarily more data could be generated. However, for direct use in a molecular property prediction pipeline, computational inference of the conformers is too slow for many applications (such as molecular search). In these settings our propsed 3D pre-training is especially valuable since inference is fast while 3D information is leveraged for better predictions.

2.1.3. SE(3) Symmetry and Chirality

There are multiple sets of 3D coordinates that describe the same conformer. For instance, if we rotate the molecule in space, the coordinates may change but the molecule and its properties are still the same. There is a set of symmetries that the 3D coordinates obey and we want our architecture to reflect these symmetries when operating on the 3D information. If this is not the case we would lose sample efficiency since our model has to learn to be invariant with respect to the symmetries instead of that prior knowledge being built into the architecture itself. We do not want our model outputs to depend on an arbitrary choice of coordinate systems.

The symmetry group of a molecule, that is the group of transformations under which a molecule is invariant, is the special Euclidean group in 3 dimensions $SE(3)$. Our model should obey these symmetries. More specifically, we have translation invariance: for a molecule of n atoms, translating our input coordinates $r_1, \ldots, r_n$ by $g \in \mathbb{R}^3$ to become $r_1 + g, \ldots, r_n + g$ should not change the output of the model. Additionally, $SE(3)$ also contains all rotations in 3 dimensions and we wish to be invariant to these: for any rotation matrix, $O \in \mathbb{R}^{3\times3}$ (matrices with determinant 1) the output of our model should be the same whether we use $Or_1, \ldots, Or_n$ or the original coordinates as input.

Note that $SE(3)$ does not include all orthogonal matrices (we are missing matrices with determinant -1) which would correspond to reflection and rotation. By adding these we would obtain the $E(3)$ symmetries. However, some molecules (like DNA) are not invariant with respect to reflection since they have a "handedness" called chirality. For many chiral drugs, their biological and pharmacological properties significantly change depending on their chirality (Nguyen, H. He, and Pham-Huy 2006). This is relevant in terms of this work since only two of our methods are non-invariant to reflection/chirality.

2.2. Graph Neural Networks

In our work, we make use of GNNs to predict molecular properties given a molecular graph. In this section, we briefly describe the very general framework of MPNNs (Gilmer, Schoenholz, Riley, et al. 2017) that fits well to describe the models which we employ.

The aim is to learn a representation of a graph $G = (\mathcal{V}, \mathcal{E})$ where $\mathcal{V} = \{1 \ldots n\}$ is the set of n vertices with features x_v and $\mathcal{E} \subset \{(u, v) \mid (u, v) \in \mathcal{V}^2 \wedge u \neq v\}$ is the set of edges with associated edge features e_{uv}. An MPNN learns a representation of G by iteratively applying message passing layers to end up with vertex representations which are then combined in a readout function. With h_v^l being the representation of vertex v in the l-th layer and $h_v^0 = x_v$, a message-passing layer can be described as first creating messages for each edge (Equation 2.2), then aggregating the messages from all connected edges (Equation 2.3), and combining the aggregated representation with the previous vertex: representation (Equation 2.4):

$$m_{uv} = MESSAGE^l(h_u^l, h_v^l, e_{uv}) \tag{2.2}$$

$$m_v = AGGREGATE^l(\{m_{uv} \mid (u, v) \in \mathcal{E}\}) \tag{2.3}$$

$$h_v^{l+1} = COMBINE^l(h_v^l, m_v). \tag{2.4}$$

The MESSAGE and COMBINE functions could be multi-layer perceptrons (MLPs) and common $AGGREGATE$ is a permutation invariant function such as taking the mean, the maximum, or the sum of the incoming messages. After this message passing phase, we have a readout phase to obtain a final graph level representation z:

$$z = READOUT(\{h_v^L \mid v \in \mathcal{V}\} \tag{2.5}$$

Where L is the index after the last message passing layer and $READOUT$ is another permutation invariant function similar to $AGGREGATE$.

2.3. Mutual Information and KL-Divergence

Mutual information is a core quantity for specifying the relationship between random variables. It measures the dependence of two random variables X and Y with a joint distribution $\mathbb{P}_{XY}$. It has found a wide range of applications in machine learning and in this work, we use it for contrastive self-supervised learning to measure the information overlap between jointly learned embeddings.

For continuous variables, mutual information $\mathcal{I}$ is formalized as

$$\mathcal{I}(X;Y) = \int_Y \int_X p_{XY}(x,y) log \frac{p_{XY}(x,y)}{p_X(x)p_Y(y)}\, dx\, dy, \tag{2.6}$$

where p_{XY} is the probability density function of the joint distribution and p_X and p_Y are the density functions of the marginal distributions $\mathbb{P}_X$ and $\mathbb{P}_Y$. While correlation can only capture linear statistical dependencies between variables this does not hold for mutual information and it can be seen as an actual measure of dependence. Using the Shannon entropy $\mathcal{H}$ as a measure of uncertainty, another interpretation of mutual information between X and Y is to see it as the decrease in uncertainty about X given Y:

$$\mathcal{I}(X;Y) = \mathcal{H}(X) - \mathcal{H}(X \mid Y), \tag{2.7}$$

where $\mathcal{H}(X \mid Y)$ is the conditional entropy of X given Y.

We will also make use of the fact that the mutual information is the same as the Kullback-Leibler divergence (KL-Divergence) (Kullback 1959) between the joint $\mathbb{P}_{XY}$ and the product of the marginals $\mathbb{P}_X \otimes \mathbb{P}_Y$

$$\mathcal{I}(X;Y) = D_{KL}(\mathbb{P}_{XY} \parallel \mathbb{P}_X \otimes \mathbb{P}_Y) \tag{2.8}$$

with the KL-Divergence D_{KL} being

$$D_{KL}(\mathbb{P} \parallel \mathbb{Q}) = \int_X p(x)\, log \frac{p(x)}{q(x)}\, dx \tag{2.9}$$

where p and q denote the probability densities of $\mathbb{P}$ and $\mathbb{Q}$.

For our purposes, the most valuable interpretation of mutual information is as a measure of how much information X and Y share. This is because it is crucial for what we call latent space self-supervision where we aim to maximize the mutual information between two latent vectors that are seen as random variables.

3. Related Work

While this is the first work on pre-training molecular property prediction models with 3D information, it builds on and draws from multiple fields. We first lay out prior work on using learned methods for molecular property prediction in Section 3.1. This ranges from neural networks operating on molecular fingerprints over graphs processed by GNNs to geometry-informed methods that leverage the 3D structure of molecules as additional information. Here, we also briefly touch on learned methods for molecular conformation generation. Next, Section 3.2 provides a general overview of the literature on self-supervised learning before describing how it has been applied to tasks on graphs. Lastly, we present previous work on SSL specifically applied to pre-training models for better molecular property predictions.

3.1. Machine Learning for Molecules

Recent years have seen a large surge of deep learning in fields such as language processing or computer vision. Building on this success, deep learning is starting to find its way into real-world applications in different fields of natural sciences, including molecular chemistry. The space of applications of ML for molecules is too vast for a single account; it includes quantum chemistry, molecular search and simulation, property prediction, molecule generation and design, drug discovery, materials science, or protein structure prediction. This section focuses on quantum mechanical and biophysical property prediction (Section 3.1.1), the use of GNNs, and how 3D information comes into play (Section 3.1.2) since these are the most relevant aspects for this thesis.

3.1.1. Molecular Property Prediction and the Role of GNNs

Relevance of ML for Property Prediction and its Beginnings. One of the most significant advances of ML for molecules is the ability to predict their biological or physical properties. This allows inferring, for instance, the biological activity of a molecule, its toxicity, or its ability to bind to specific proteins. Areas, where such predictions have had a particularly substantial impact, are virtual screening for drug discovery (Kimber, Y. Chen, and Volkamer 2021; Stokes, K. Yang, Swanson, et al. 2020), where an ML model can search through millions of molecules, predict their properties, and potentially find ones with the attributes necessary for a certain drug or another application. In comparison, running *in vitro* experimentations at the same scale would require tremendous time and resources.

The first approaches for predicting molecular properties were Quantitative Structure-Activity Relationship (QSAR) models which relate a set of molecular descriptors to regression or classification values. The molecular descriptors (Mauri, Consonni, and Todeschini 2017;

Todeschini and Consonni 2009; Capecchi, Probst, and Reymond 2020) used in QSAR are generated by rule-based algorithms derived from expert knowledge and could be tailored to the specific task at hand. These methods have been successful and used since the 1930s (Walters and Barzilay 2021b) in real-world applications. Meanwhile, when it comes to quantum mechanical properties, non-statistical methods such as density functional theory are usually employed, coming with high accuracy but long computation times (Ramakrishnan, Dral, Rupp, and Lilienfeld 2014).

Deep Learning enters the Picture. Through the developments in deep learning, the field of learned molecular property predictions has recently undergone a revolution. Instead of using expert-derived molecular features, the models can directly operate on canonical representations leading to a higher degree of flexibility, and combined with the ability of neural networks to discover complex correlations they can outperform QSAR methods.

For deep learning approaches multiple different types of standard input types have been explored. These include representations such as strings, molecular graphs, 3-dimensional graphs, or molecular surfaces. When operating on strings such as SMILES (Weininger 1988) or the more deep learning oriented self-referencing embedded strings (SELFIES) (Krenn, Häse, Nigam, et al. 2020) common methods from natural language processing are often adopted such as recurrent neural networks or Transformers (Vaswani, Shazeer, Parmar, et al. 2017).

Notably, for large molecules like proteins, using text representations to train language models has produced strong results (Rives, Meier, Sercu, et al. 2021) and even major scientific breakthroughs like with AlphaFold2 (Jumper, Evans, Pritzel, et al. 2021). However, for small molecules, string representations unnecessarily omit the graph structure of molecules which is a strong inductive bias.

Deep Learning on Molecular Graphs. Molecular graphs enable leveraging the valuable locality information of atoms. Using GNNs to process molecules induces an important inductive bias into the ML architecture: atoms that are connected by bonds are close in the graph, meaning that they have the greatest influence on each other. Therefore, GNNs have found tremendous success in processing molecules which has become one of their main applications (Zhou, Cui, S. Hu, et al. 2020). One of the first accounts being by Duvenaud, Maclaurin, Aguilera-Iparraguirre, et al. 2015 who demonstrated it as a promising approach.

Today, many of the state-of-the-art (SOTA) learned methods for molecules use GNNs (W. Hu, Fey, Zitnik, et al. 2020) and advances in graph representation learning research have big impacts on computational chemistry. Similarly, progress for learning on graphs has been driven from the application side as, for instance, MPNNs (explained in Section 2.2) (Gilmer, Schoenholz, Riley, et al. 2017) were first developed for molecules and are now a widely used framework for processing graphs (Battaglia, Hamrick, Bapst, et al. 2018; Geerts, Mazowiecki, and Perez 2021). Nowadays, many GNN methods even measure their strength with particular focus on molecular property prediction benchmarks (Bouritsas, Frasca, Zafeiriou, and Bronstein 2020; Corso, Cavalleri, Beaini, et al. 2020; Beaini, Passaro, Létourneau, et al. 2021; Bodnar, Frasca, Otter, et al. 2021).

After Gilmer, Schoenholz, Riley, et al. 2017 first introduced MPNNs to predict quantum mechanical properties at the accuracy of density functional theory (DFT) but orders of magnitude faster, GNNs became more and more widely and successfully applied in quantum chemistry (Brockschmidt 2020; B. Tang, Kramer, Fang, et al. 2020; Withnall, Lindelöf, Engkvist, and H. Chen 2020), drug discovery (J. Li, D. Cai, and X. He 2017; Stokes, K. Yang, Swanson, et al. 2020; Torng and Altman 2019), and molecular property prediction in general (Coley, Jin, Rogers, et al. 2019; Hy, Trivedi, H. Pan, et al. 2018; Unke and Meuwly 2019). There are rigorous evaluations of MPNNs for property prediction (K. Yang, Swanson, Jin, et al. 2019) showing the effectiveness of the approach and easily accessible molecular datasets drive progress in the established field (Z. Wu, Ramsundar, Feinberg, et al. 2017; W. Hu, Fey, Zitnik, et al. 2020).

GNNs are not the only way to leverage the inductive bias given by the molecular graph. After the first generalization of transformers to graphs (Dwivedi and Bresson 2020) multiple methods have been applied to molecular property prediction (Kreuzer, Beaini, Hamilton, et al. 2021; Mialon, D. Chen, Selosse, and Mairal 2021) with promising results. On a large scale quantum mechanical task, (Ying, T. Cai, S. Luo, et al. 2021) even achieve SOTA with a transformer and they further show benefits from pre-training their model on large datasets followed by fine-tuning on the target task.

The other two mentioned molecular representations, 3D graphs, and molecular surfaces, leverage additional 3D information in the form of molecular conformers (see Section 2.1.2). However, this information is not trivial to obtain which will be explained in the next section while presenting the most successful and promising approaches operating on 3D molecular data.

3.1.2. Leveraging 3D Information

Why 3D Information is valuable. The properties of a molecule are not only based on bonds and the information given in a molecular graph. Instead, it is determined by the underlying physical interactions of atoms which depend on their location in space. Two atoms in a molecule with a "bend" could be very far in the molecular graph while they are close in space and actually strongly influence each other, e.g., through electrostatic or van der Waals forces.

Furthermore, the interactions between molecules strongly depend on their geometry to the extent that different conformers of the same molecular graph have different properties. For instance, when determining whether a molecule binds to a protein, it is required to know if the molecule geometrically "fits" into the protein's binding pocket. As such, including 3D information as additional input to ML should enable them to drastically improve their performance and for many tasks, this has been confirmed as the following will describe.

Before discussing how the full 3D information in the form of atom coordinates is used when operating on 3D graphs, it is worth mentioning molecular surfaces as input representations which have recently found increasing popularity and success. A molecular surface is the surface enclosing the molecule with a certain distance from each atom. Represented as meshes, rotation-invariant 2D Convolutional Neural Network (CNN) can operate on the surfaces with

which Gainza, Sverrisson, Monti, et al. 2020 achieved impactful results for predicting binding sites. Additional methods successfully followed up on this (Mylonas, Axenopoulos, and Daras 2020; Q. Liu, P.-S. Wang, C. Zhu, et al. 2021) and the use of surface-based representations seems to grow.

3D GNNs for Molecules. In a complete 3D representation of a molecule via a 3D graph, each atom has an additional 3D coordinate. As input for an ML method, this 3D information is preferably used in a way that captures the $SE(3)$-invariance of the coordinates (see Section 2.1.3). Early on, various MPNNs started doing so by using bond distances as additional edge features (Gilmer, Schoenholz, Riley, et al. 2017) which is already able to significantly improve results for quantum mechanical property prediction (P. Chen, W. Liu, Hsieh, et al. 2020). One of the most prominent methods for using 3D information is SchNet (Schütt, Kindermans, Felix, et al. 2017) which was the SOTA on multiple tasks for a long time.

However, if the messages in MPNN only depend on bond distances, this fails to capture a large portion of the 3D information. Additional information can be used by including the angles between bonds when generating the messages which are another $SE(3)$ invariant quantity next to the distances. This idea is implemented in DimeNet and DimeNet++ by Klicpera, Groß, and Günnemann 2020; Klicpera, Giri, Margraf, and Günnemann 2020 to achieve a new level of accuracy for quantum mechanical property predictions.

Spherical Message Passing (SMP) (Y. Liu, L. Wang, M. Liu, et al. 2021) followed up on this by adding more angular information that is sufficient to completely define the positions of an atom and all its 1-hop neighbors. In a similar fashion, SMP was superseded by GemNet (Klicpera, Becker, and Günnemann 2021) that finally also captures torsion angles (the relative rotation around a bond of two substructures connected by that bond). With this inclusion, the whole geometry of a molecule is uniquely defined and the MPNN can leverage all of the 3D information. In this line of work, SE(3)-equivariant GNNs (Batzner, Smidt, L. Sun, et al. 2021) have to be mentioned as well which use the 3D structure in an equivariant architecture for simulating a molecule's dynamics.

Other Approaches for 3D Molecular Graphs. Another approach separate from the GNN paradigm is that of $SE(3)$-Transformer (F. Fuchs, Worrall, Fischer, and Welling 2020) which directly processes the 3D coordinates in an $SE(3)$ equivariant manner. Furthermore, there are Equivariant Graph Neural Networkss (EGNNs) (Satorras, Hoogeboom, and Welling 2021) and the work of Satorras, Hoogeboom, F. B. Fuchs, et al. 2021 where all the pairwise distances are used to completely specify the 3D geometry of small molecules (for their molecule benchmark EGNN is only a GNN in the sense that it operates on the complete graph).

The aforementioned approaches on 3D graphs use the 3D information to improve the accuracy of quantum mechanical tasks such as predicting quantum mechanical properties. For biophysical and pharmacological properties the use of 3D information is much less explored with initial steps taken by Axelrod and Gómez-Bombarelli 2020 who try to make more accurate predictions for COVID19 related tasks.

While leveraging 3D information is advantageous, it is not trivial to obtain. Thus, so far, it was mostly used for quantum mechanical tasks on smaller molecules, with rare use-cases for bigger drug-like molecules (Ying, T. Cai, S. Luo, et al. 2021). With increasing molecular sizes, there are exponentially more possible conformations - finding low-energy conformers becomes more expensive and time-consuming with classical conformation generation methods (see Section 2.1.2). The next section will touch on generating conformers with learned methods.

3.1.3. Conformation Generation

Previous sections explained the value of 3D information for molecular property prediction tasks, but they suffer a major computational bottleneck. In time-sensitive applications where the inference speed needs to be high, it is desirable to have a method that can directly go from molecular graphs to 3D information in a short time. This way, more accurate property predictions would become possible by taking the molecular graph as input, generating the 3D information, and using it with one of the 3D methods in section 3.1.2.

One idea to achieve this is to learn molecular conformation generation with a deep learning model. This trades the accuracy of classical methods for speed, but it allows to make the problem computationally tractable, especially for larger molecules such as proteins. This field has seen tremendous progress and even a scientific breakthrough in the case of protein structure predictions, namely AlphaFold2 (Jumper, Evans, Pritzel, et al. 2021).

For smaller molecules, one of the first methods used a variational autoencoder for graphs (Mansimov, Mahmood, Kang, and Cho 2019) that directly generates 3D coordinates of atoms but does not capture the spatial invariances of molecules. Improving on this, Simm and Hernández-Lobato 2020 proposed to generate atomic distances. Since then more $SE(3)$ invariant approaches have been proposed with ever increasing accuracy (M. Xu, S. Luo, Bengio, et al. 2021; Shi, S. Luo, M. Xu, and J. Tang 2021). The current SOTA for most types of drug-like molecules is held by GeoMol (Ganea, Pattanaik, Coley, et al. 2021) which first generates local structures of the 1-hop neighborhood around an atom and then fits these together by predicting the torsion angles between them.

This progress raises the question of whether the conformations generated by learned methods are accurate enough to inform property predictions given only a molecular graph, but there has been no work investigating this. Moreover, there is no prior work that investigates improving molecular predictions based on 3D information that can be generated with learned methods when only given the 2D molecular graph.

Thus this thesis proposes to train a model to generate implicit 3D information encoded in latent representations that can then be used by learned methods to inform downstream predictions. For this purpose, we draw from the field of self-supervised learning whose recent advances and related work are presented in the next section (Section 3.2).

3.2. Self-Supervised Learning

Another central topic of our work is the field of SSL from which we draw several methods. The aim of SSL is to train models with auxiliary tasks that can learn useful representations for general tasks without expensive labeling of the data. The interest is not in the final performance on the self-supervised task but rather in learning a representation that contains helpful information for downstream tasks.

We describe the progress of general SSL in Section 3.2 in a non-exhaustive manner, where the goal is not to provide a comprehensive review. Rather it is to provide an overview of the methods and principles that are used in our work, their origins, and what is our contribution to the field. Then, the last area of related work presents previous applications of SSL to graphs and molecules (see Section 3.2.1).

3.2.1. General Advances in Self-Supervised Learning

The main advances in SSL come from domains other than graph representation learning, such as computer vision or audio and text processing. One of the most popular and first approaches is denoising autoencoders (Vincent, Larochelle, Bengio, and Manzagol 2008) where parts of the input are corrupted and the pre-training task is to reconstruct the original input signal. Today this denoising pre-training task is finding wide applicability and huge successes for pre-training transformer language models (Devlin, Chang, Lee, and Toutanova 2019) or transformers for vision tasks (Dosovitskiy, Beyer, Kolesnikov, et al. 2021).

Closer to our work is SSL with Siamese Neural Networks (two similar "twin" networks producing latent representations for different inputs) and contrastive learning objectives. In contrastive learning, the goal is to learn a representation space in which semantically similar data points are close while dissimilar points are mapped far apart. An early contrastive loss that achieves this is by Chopra, Hadsell, and LeCun 2005 for learning face similarity. Similarly, the Triplet Loss (Schroff, Kalenichenko, and Philbin 2015) is widely used for the same application of matching faces.

An important milestone was met when the notion that we wish to maximize the mutual information between two representations entered the picture. The idea is to interpret the output representations of Siamese Networks as distributions between which the mutual information is maximized. With Contrastive Predictive Coding Oord, Y. Li, and Vinyals 2018 introduced the noise contrastive estimation of mutual information (InfoNCE) loss. It maximizes a lower bound on the mutual information via neural estimation of mutual information (Belghazi, Baratin, Rajeswar, et al. 2018) which itself is based on noise contrastive estimation of mutual information (Gutmann and Hyvärinen 2010). The mutual information maximization perspective has been further developed and has gained a firm footing through Deep Infomax (DIM) (Hjelm, Fedorov, Lavoie-Marchildon, et al. 2019) who evaluate different mutual information estimators including InfoNCE.

InfoNCE is now one of the most widely used objectives for contrastive learning used by influential works of the field such as momentum contrast (MoCo) (K. He, Fan, Y. Wu, et al. 2020) which strongly popularized SSL on images and showed it as a very promising direction

together with the follow-up work SimCLR T. Chen, Kornblith, Norouzi, and Hinton 2020 and SwAV (Caron, Misra, Mairal, et al. 2020). In these frameworks, semantically similar data points (positive pairs) are obtained via augmentations that change a datapoint while leaving its meaning as intact as possible (e.g., rotating an image). Especially SimCLR showed how to make contrastive learning work reliably and the importance of using the right augmentations, large batch sizes, and using hard negative pairs (e.g., images that look very similar but have different labels).

However, contrastive learning comes with drawbacks such as the reliance on large batch sizes. As such, new Siamese Network based methods have been introduced that do not use contrastive loss terms. The similarity to contrastive learning is that they also try to learn a latent space where semantically similar data points are mapped to closeby vectors in embedding space. However, this could be achieved by mapping all data to the same point in the latent space, and collapse to this trivial solution needs to be prevented. While in contrastive learning this is done with the constraint of mapping negative pairs far apart, other tricks can be applied to avoid collapse.

One of the first methods to choose this path is BYOL (Grill, Strub, Altché, et al. 2020). BYOL prevents collapse with asymmetry between the twin networks and a stop gradient operation where only one of the twins is updated with the gradients of a similarity loss while the second twin updates its weights as an average of its own weights and those of the first twin. It is not entirely understood why this avoids collapse but there is a lot of research on the topic (Tian, X. Chen, and Ganguli 2021; Richemond, Grill, Altché, et al. 2020) with SimSiam (X. Chen and K. He 2020) simplifying the approach and finding the stop gradient operation as the most significant contributor. BYOL is now also an established approach with impressive applications such as the completely self-supervised semantic segmentation model DINO (Caron, Touvron, Misra, et al. 2021) and the ability to outperform SOTA contrastive methods (Assran, Caron, Misra, et al. 2021).

Recently Recasens, Luc, Alayrac, et al. 2021 already proposed a setup similar to SimSiam for SSL in a multi-modal setting with both video and audio data. However, our work proposes the only complete generalization of BYOL to arbitrary multi-modal inputs. While we apply it to molecular graphs and 3D information, the method could be used to learn similar representations between any type of data.

Following BYOL, more principled approaches for avoiding collapse have been proposed with Barlow Twins (Zbontar, Jing, Misra, et al. 2021) and Variance-Invariance-Covariance Regularization (VICReg) (Bardes, Ponce, and LeCun 2021). Their idea is an additional loss term that explicitly forces the data points in a batch to have different representations.

What all of the mentioned Siamese Network-based methods have in common is that they have been applied to graph data, except for VICReg. We are the first to consider and evaluate SSL on graphs using VICReg. The previous applications of SSL for graphs are covered in the next Section 3.2.2 with a focus on molecular graphs.

3.2.2. Self-Supervised Learning for Graphs and Molecules

Applying SSL methods to graphs has garnered a lot of attention recently and many were adapted to graph-structured data in very short time frames. A complete review (up to the publication date) is provided by Xie, Z. Xu, Z. Wang, and Ji 2021. Similar to the previous Section 3.2.1 we focus on Siamese Network-based methods. However, for predictive methods, the popular Variational Graph Autoencoders (Kipf and Welling 2016) have to be mentioned.

Deep Infomax (DIM) (Hjelm, Fedorov, Lavoie-Marchildon, et al. 2019) was applied to graphs on the node-level by Deep Graph Infomax (DGI) (Velickovic, Fedus, Hamilton, et al. 2019) and extended to graph level representations by InfoGraph (F.-Y. Sun, Hoffmann, Verma, and J. Tang 2020). After their success, many works followed with different graph augmentation strategies and evaluating different mutual information estimators (W. Hu, B. Liu, Gomes, et al. 2020; Hassani and Ahmadi 2020; Peng, Huang, M. Luo, et al. 2020; Y. Zhu, Y. Xu, Yu, et al. 2021).

One of the main difficulties for SSL on graphs is the question of what augmentations to use. For images, there are augmentations that work reliably across multiple tasks and for which it is clear that they do not change the semantics of a data point for the given task. Meanwhile, augmentations on graphs are not so straightforward. For instance, considering a molecular graph, it is not clear whether the molecule will still have the same properties after a common augmentation such as dropping a node. In regard to this, Graph Contrastive Learning (GraphCL) (You, T. Chen, Sui, et al. 2020) finally did a complete evaluation of augmentations on graphs and put graph contrastive learning into a comprehensive framework. Furthermore, You, T. Chen, Shen, and Z. Wang 2021 automates the augmentation procedure with a bi-level optimization approach.

For SSL on molecular graphs, there have been multiple attempts of replicating the successes of denoising objectives in natural language processing such as BERT (Devlin, Chang, Lee, and Toutanova 2019). Similar to masking out words in a sentence, the idea is to mask out atoms in a molecule and there has been a moderate success with this approach (Chithrananda, Grand, and Ramsundar 2020; Maziarka, Danel, Mucha, et al. 2020).

A popular work that explores pre-training for molecular property prediction is by W. Hu, B. Liu, Gomes, et al. 2020 who use a contrastive approach which is followed by Y. Wang, J. Wang, Cao, and Farimani 2021. However, GraphCL (You, T. Chen, Sui, et al. 2020) achieves more convincing results on molecular tasks together with the follow-up method by You, T. Chen, Shen, and Z. Wang 2021. The current best performance for self-supervised pre-training and fine-tuning on molecular tasks is achieved by GraphLoG (M. Xu, H. Wang, Ni, et al. 2021) which not only enforces similarity and dissimilarity between positive and negative pairs but also between hierarchical summary vectors representing multiple datapoints.

Impressive improvements on molecular tasks have been reached by some methods, but it has to be noted that evaluations are often not completely rigorous and comparisons are made between different scaffold splits (see Section 2.1.1) or on random splits for some datasets (Maziarka, Danel, Mucha, et al. 2020) instead of employing standard splits such as those of W. Hu, Fey, Zitnik, et al. 2020.

While multiple SSL methods have been explored for molecular graphs, we are the first to do so for BYOL, Barlow Twins, and VICReg. Additionally, none of the methods were evaluated for quantum mechanical properties. Furthermore, there is also no previous work on using 3D information in an SSL setting for molecules. Our work distinguishes itself from previous SSL literature in that it uses 3D structural information for pre-training. Since 3D information is valuable for many downstream molecular tasks, this is a more principled approach than SSL methods which use augmentations such as dropping atoms for which it is not clear why a molecule should still have the same properties.

4. Definitions and Terminology

In this section, we first define the type of data that our method will be operating on and how we represent molecules as a graph and 3D coordinates in our work (Section 4.1). We then lay out the general framework in which our different methods of pre-training models with 3D information can be described (Section 4.2).

4.1. Molecular Graph Data and 3D Data

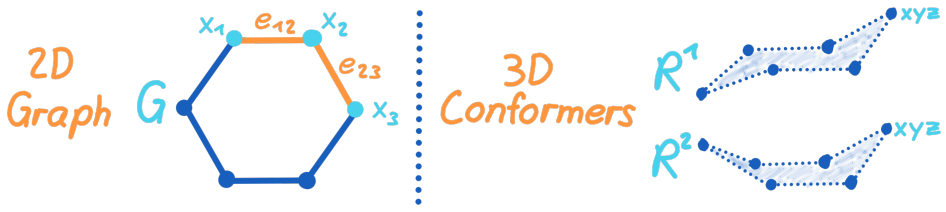


Figure 4.1.: Illustration of the terminology used to define a molecular graph G with its atom features x_v and edge features e_{uv} as well as the corresponding conformers of the molecule R^j.

We represent a molecule as a graph which we define as a 4-tuple $G = (\mathcal{V}, \mathcal{E}, X, E)$ and associated 3D Cartesian coordinates R. $\mathcal{V} = \{1 \dots n\}$ is the set of n vertices which we will also call nodes. Each vertex $v \in \mathcal{V}$ in the graph corresponds to an atom with features $x_v \in \mathbb{R}^{d_v}$ and we denote the set of all node features of a graph as X. Such a node feature vector encodes information about an atom which could, for instance, be its atomic number, charge, hybridization, or whether it is in a ring. Every atomic bond of the molecule is encoded as an edge. $\mathcal{E} \subset \{(u, v) \mid (u, v) \in \mathcal{V}^2 \wedge u \neq v\}$ is the set of edges with associated edge features $e_{uv} \in \mathbb{R}^{d_e}$. The set of edge features for a given graph G is called E and they represent aspects of a bond such as whether it is a single, double, or triple bond. The neighborhood of a node $N(u)$ is the set of all nodes that share an edge with u, that is $\{v \in \mathcal{V} \mid (u, v) \in \mathcal{E} \vee (v, u) \in \mathcal{E}\}$.

Every node v in our graph also has a 3D Cartesian coordinate $r_v \in \mathbb{R}^3$. R is the set of all 3D coordinates of a given graph. Sometimes we discuss multiple possible conformers (explained in Section 2.1.2) of a molecule and there will be c many sets of 3D coordinates $\{R^j\}_{j \in \{1 \dots c\}}$ for each molecule. When referring to the 3D information of a molecule, these coordinates are meant while the 2D information of a molecule should mean the information given by the bonds (the graph connectivity), the bond features, and the atom features.

4.2. Self-Supervised learning

To structure this work, we distinguish between latent space SSL where a loss is applied to latent vectors and predictive SSL where the pre-training task is to generate explicit 3D information. In our presentation of latent space SSL, we deal with a 2D network f^a which is a function with learnable parameters. It takes all 2D information of a molecule as input to produce a latent 2D representation $f^a(G) = z^a \in \mathbb{R}^{d_z}$. Similarly, a 3D network f^b takes a molecule's coordinates as input to produce a latent 3D vector $f^b(R) = z^b \in \mathbb{R}^{d_z}$ (Figure 5.1 is illustrative). These networks consist of an encoder $\mathcal{G}^a$ and a readout projection head $\mathcal{D}^a$ such that $f^a = \mathcal{D}^a \circ \mathcal{G}^a$ and we write the hidden representations produced by the encoders as $h^a = \mathcal{G}^a(G)$ with $h^a \in \mathbb{R}^{d_h}$. For the 3D network it is analogous with the superscript b. Furthermore, the weights of any learnable function/network f are denoted by ω_f.

For latent space SSL, we are considering whole batches of molecules with a batch size N. The i-th molecule in a batch is given by its graph G_i and its conformer-coordinates $\{R_i^j\}_{j\in\{1\dots c_i\}}$ with $i \in \{1 \dots N\}$. As such, the 2D network produces a batch of representations $f^a(G_i) = z_i^a$, one for each molecule and we refer to the whole batch as $Z^a = [z_1^a, \dots, z_N^a] \in \mathbb{R}^{N\times d_z}$. Meanwhile, the 3D network's outputs are the 3D representations $f^b(R_i^j) = z_{i,j}^b$ for the i-th molecule and its j-th conformer. When only considering a single conformer, we drop the index j and the whole batch of 3D representations is $Z^b = [z_1^b, \dots, z_N^b] \in \mathbb{R}^{N\times d_z}$. If two representations in a batch, z_i^a and z_i^b, come from the same molecule (they have the same index i) we call them a positive pair and a negative pair otherwise. Lastly, for discussing the mutual information maximization perspective of contrastive learning, each molecule (G, R) is assumed to come from a data distribution $\mathcal{P}$.

5. 3D Pre-Training

Our goal is to develop a method that generates latent 3D information from a 2D molecular graph and uses it for more accurate molecular property predictions. It is desired that the approach improves the performance of property predictions from different domains and can easily be applied to arbitrary properties.

For this purpose, we propose to pre-train a GNN to generate latent representations of a molecule's 3D geometry. Through this pre-training, the network learns the connection between 2D information of molecular graphs and the molecule's possible 3D conformers. Intuitively, it learns to "understand" molecular dynamics.

After pre-training, we transfer the weights and fine-tune them on property prediction tasks. During fine-tuning, the latent 3D information can be used to inform and improve the downstream predictions. Such 3D pre-training can be helpful for a variety of molecular tasks since the 3D structure is inherently valuable for the workings of any molecule. This is in contrast to pre-training via property predictions where only similar downstream tasks can benefit from it.

The ideal result would be to have a pre-trained "ResNet for molecules"; i.e., a model with very generalizable weights that can be reused for various tasks, similar to how it is commonly done with ResNet models pre-trained on ImageNet (Deng, Dong, Socher, et al. 2009). In the same fashion, having a set of GNN weights that can straightforwardly be downloaded and tuned to improve predictions for molecular tasks would be highly valuable.

A requirement in the pre-training step is to generate representations that capture as much 3D information as possible without encoding a lot of the 2D information. The 2D molecular graphs of the downstream task are available during fine-tuning and we want them to influence our predictions and not the 2D aspects of the pre-training dataset. As such, the weight transfer and tuning method needs to process the 2D information for a given task while retaining the ability to encode 3D information. Lastly, during fine-tuning, the model needs to be able to use the latent 3D information.

The pre-training methods that we devise can be split into two categories at the highest level. First, latent space self-supervision where the aim is to maximize the agreement between latent 2D and 3D representations if they come from the same molecule. The second is predictive self-supervision where the model generates some form of 3D information that can be directly calculated from the coordinates.

One might consider SSL as a misnomer for our method since self-supervised usually implies that the supervision signal can easily be automatically generated from the data which is not the case with 3D information of molecules that is not trivial to generate. Nevertheless, we use the terminology since our framework is highly similar to SSL methods.

5.1. Latent Space 3D Self-Supervision

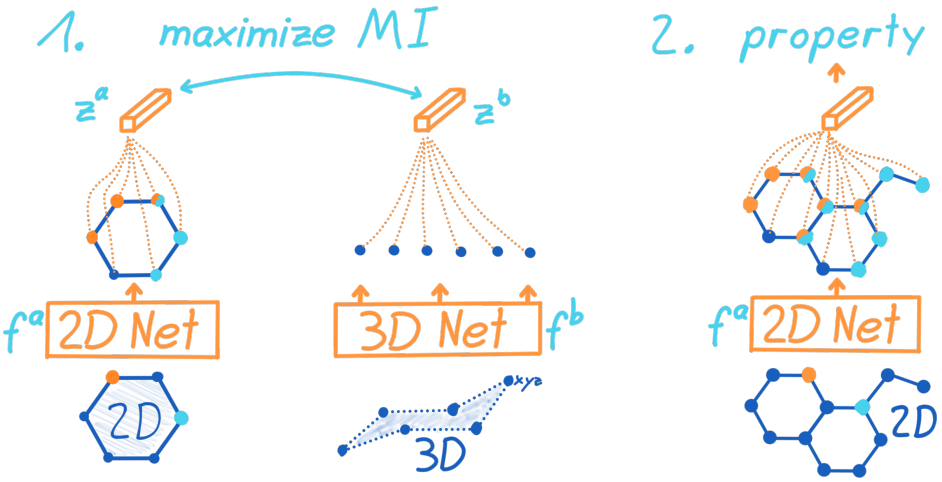


Figure 5.1.: General overview of our latent space SSL methods. In step 1. we pre-train a 2D network by forcing it to generate a 2D representation z^a that has high mutual information with a 3D representation z^b coming from an encoder that has only 3D information as its input. After the 2D model has 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.

We present four self-supervised learning approaches and adapt them to our setting: (1) Contrastive learning, (2) BYOL (Grill, Strub, Altché, et al. 2020), (3) Barlow Twins (Zbontar, Jing, Misra, et al. 2021), and (4) VICReg (Bardes, Ponce, and LeCun 2021). These methods usually operate on an input sample, such as an image, which is augmented in two different ways to generate two views of the sample. Augmentations, such as rotating an image, should change the sample in principle, but leave the semantics the same. In the four methods, the models have to learn to produce representations that are invariant with respect to the augmentations. Two different views of the same image are the inputs to a model that produces two latent representations. Since the inputs are semantically the same, their latent representations should be as similar as possible and this is the objective in latent space self-supervision.

In our setting, a sample is a molecule and the two views are given by the 2D information and the 3D information which are processed by two different models. We will refer to the model that uses the 2D inputs to generate a latent representation of a molecule as the 2D network f^a. Analogously, we define the 3D network f^b which encodes 3D information. Instead of forcing a single model to be invariant with respect to augmentations, here, we want to maximize the agreement between latent 2D representations z^a and 3D representations z^b if they come from the same molecule. This will also be further motivated by the perspective of maximizing the mutual information between the representations in the contrastive learning approach.

During pre-training, the 2D model has to learn to generate 3D information and encode it in its latent representation in order to maximize the similarity with the 3D representation. The pre-trained network is then fine-tuned for the property of interest as illustrated in Figure 5.1.

Adapting these methods to our settings comes with some challenges. Firstly, self-supervised learning literature mostly focuses on computer vision and the approaches are only partially explored for graph data. Contrastive learning has already been applied to GNNs for molecular property prediction (Y. Wang, J. Wang, Cao, and Farimani 2021; You, T. Chen, Sui, et al. 2020; You, T. Chen, Shen, and Z. Wang 2021), but the other methods have not yet been shown to work on molecular data. Moreover, BYOL and Barlow Twins have only been explored for node-level tasks (Thakoor, Tallec, Azar, et al. 2021; Kefato and Girdzijauskas 2021; Bielak, Kajdanowicz, and Chawla 2021) and VICReg has not yet been applied to graph data at all.

Secondly, even if the methods were previously used on graph data, they operate on a single modality. In our setup, however, we have one view in the form of the 2D molecular graph with its features and the other view given by a set of 3D coordinates. This means that we need two models with different weights and a different architecture for the 2D information and the 3D information which is especially problematic for BYOL.

After presenting our four main methods adapted from SSL research, we additionally extend them to a local-global information maximization scheme (see Section 5.1.5). Moreover, we come up with a method to leverage the information of multiple molecular conformers in Section 5.1.6. Lastly, we present and thoroughly discuss our different 3D networks in Section 5.1.7.

5.1.1. Contrastive Setup

We first provide an overview and an intuitive understanding of contrastive learning before explaining the mutual information maximization perspective and the actual objective functions. In our setup, we wish to maximize the similarity between latent 2D representations z_i^a and latent 3D representations z_i^b if they come from the same molecule in our batch of N molecules (i.e., they have the same index i and are therefore a positive pair). We could, for instance, use the cosine similarity as a similarity measure between these two vectors:

$$sim_{cos}(z^a, z^b) = \frac{z^a \cdot z^b}{\|z^a\|_2 \|z^b\|_2}, \tag{5.1}$$

where $\|\cdot\|_2$ is the ℓ_2-norm. The cosine similarity gives a value of 1 when the vectors are co-aligned, of 0 when they are orthogonal, or -1 when they are in opposite directions.

One idea would be to maximize the average similarity of all positive pairs $\frac{1}{N}\sum_{i=1}^{N} sim_{cos}(z_i^a, z_i^b)$ as the objective to optimize the weights of our 2D and 3D networks f^a, f^b. However, this would result in a collapse since there are infinitely many trivial solutions for the weights of f^a and f^b by producing a constant vector. The cosine similarity would always be 1 and the representations will not capture useful information.

To avoid this collapse, contrastive learning also enforces dissimilarity between negative pairs z_i^a and z_k^b where $i \neq k$ i.e., the 2D and the 3D latent vectors in our batch that correspond to different molecules as it is illustrated in Figure 5.2. This way we learn a latent space in

which the positive pair representations are pulled together to have a higher similarity and the negative pairs are pushed apart.

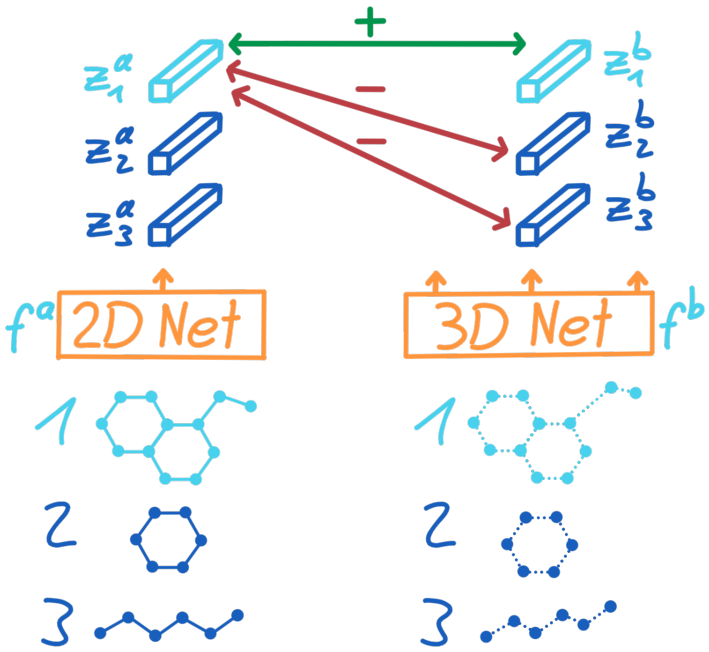


Figure 5.2.: Explanatory illustration for contrastive learning. This example displays a batch of three molecules with their 2D graphs on the left and the corresponding 3D information on the right. To learn a joint embedding space the contrastive loss enforces high similarity between latent representations 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 and third. The final loss is the average of those three.

Mutual information maximization. We can consider the objective of contrastive learning as maximizing the agreement between two hidden representations from the same molecule as measured by the mutual information $\mathcal{I}(h^a, h^b)$. Here h^a and h^b are considered as two random variables belonging either to a joint distribution $p(h^a, h^b)$ of positive pairs or the product of two marginals $p(h^a)p(h^b)$ for negative pairs. However, the exact computation of mutual information is not tractable in our setting and we need to use estimators as a learning objective instead. These estimators are lower bounds on the mutual information and by maximizing them the mutual information is increased.

We consider three different options for the estimator resulting in three loss functions which we compare in our experiments. They are the commonly employed lower bounds on the mutual information as they are used by Hjelm, Fedorov, Lavoie-Marchildon, et al. 2019. These are given by the Donsker-Varadhan representation $\hat{\mathcal{I}}^{(DV)}$ (Donsker and Varadhan 1975; Belghazi, Baratin, Rajeswar, et al. 2018), the Jensen-Shannon estimator $\hat{\mathcal{I}}^{(JSD)}$ (Nowozin,

Cseke, and Tomioka 2016), and noise contrastive estimation (Gutmann and Hyvärinen 2010) termed the InfoNCE estimator $\hat{\mathcal{I}}^{(NCE)}$ by Oord, Y. Li, and Vinyals 2018.

In all approaches, we use a discriminator $\mathcal{D}_\omega : \mathbb{R}^h \times \mathbb{R}^h \mapsto \mathbb{R}$ with weights ω that maps two hidden representations to a similarity score. In our case, this discriminator consists of the readout projection heads $\mathcal{D}^a$ and $\mathcal{D}^b$ whose outputs z^a and z^b are combined to produce a single scalar as agreement score. With this setup, we now present the three specific lower bounds to estimate the mutual information.

The **Donsker-Varadhan Estimator** by Belghazi, Baratin, Rajeswar, et al. 2018 is based on the Donsker-Varadhan dual representation (Donsker and Varadhan 1975) of the KL-Divergence

$$D_{KL}(p(h^a, h^b) \parallel p(h^a)p(h^b)) = \sup_{\mathcal{D}\in\mathcal{F}} \mathbb{E}_{(G,R)\sim\mathcal{P}}[\mathcal{D}(h^a,\ h^b)] - log\mathbb{E}_{[(G,R),(G',R')]\sim\mathcal{P}\times\mathcal{P}}[e^{\mathcal{D}(h^a,(h^b)')}], \tag{5.2}$$

where $\mathcal{F}$ is the class of all functions such that the expectations are finite. Moreover, h^a and h^b in the first term are computed from $(G, R) \sim \mathcal{P}$, while h^a and $(h^b)'$ in the second term are calculated from (G, R) and (G', R') which are independent and identically distributed according to $\mathcal{P}$. This is in saying $p(h^a, h^b) = p(h^a)p(h^b \mid h^a) = p(\mathcal{G}^a(G))p(\mathcal{G}^b(R) \mid (G, R))$ recalling that $\mathcal{G}^a$ is the encoder part of f^a. By characterizing the mutual information as the KL-Divergence between the joint and the product of the marginals $\mathcal{I}(h^a,\ h^b) = D_{KL}(p(h^a, h^b) \parallel p(h^a)p(h^b))$ we obtain

$$\mathcal{I}(h^a,\ h^b) \geq \sup_{\omega} \mathbb{E}_{(G,R)\sim\mathcal{P}}[\mathcal{D}_\omega(h^a,\ h^b)] - log\mathbb{E}_{[(G,R),(G',R')]\sim\mathcal{P}\times\mathcal{P}}[e^{\mathcal{D}_\omega(h^a,(h^b)')}], \tag{5.3}$$

as lower bound since we now only take the supremum over the discriminators $\mathcal{D}_\omega$ which are a restriction of $\mathcal{F}$. We use this lower bound for the final Donsker-Varadhan estimator:

$$\hat{\mathcal{I}}^{(DV)}(h^a,\ h^b) = \mathbb{E}_{(G,R)\sim\mathcal{P}}[\mathcal{D}_\omega(h^a,\ h^b)] - log\mathbb{E}_{[(G,R),(G',R')]\sim\mathcal{P}\times\mathcal{P}}[e^{\mathcal{D}_\omega(h^a,(h^b)')}], \tag{5.4}$$

which we aim to maximize during training via the parameters of the discriminator ω (the parameters of the readout head) to obtain a good estimate of the mutual information and via the parameters of the encoders to maximize the mutual information.

The **Jensen-Shannon Estimator** instead computes the Jensen-Shannon Divergence (JS-Divergence) between the joint and the product of marginals

$$\hat{\mathcal{I}}^{(JSD)}(h^a,\ h^b) = \mathbb{E}_{(G,R)\sim\mathcal{P}}[log(\mathcal{D}_\omega(h^a,\ h^b))] + \mathbb{E}_{[(G,R),(G',R')]\sim\mathcal{P}\times\mathcal{P}}[log(1 - \mathcal{D}_\omega(h^a,\ (h^b)'))], \tag{5.5}$$

where h^a and h^b in the first term are computed from $(G, R) \sim \mathcal{P}$, while h^a and $(h^b)'$ in the second term are calculated from (G, R) and (G', R') which are independent and identically distributed according to $\mathcal{P}$. Commonly, the discriminator computes its similarity score by the inner product of two later representations and the sigmoid. We follow this such that $\mathcal{D}_\omega(h^a,\ h^b) = sigmoid((z^a) \cdot z^b) = sigmoid((\mathcal{D}^a(h^a)) \cdot \mathcal{D}^b(h^b))$. The Jensen-Shannon estimator is intuitively similar to the Donsker-Varadhan estimator in that we can see them both as classifiers that maximize the expected log-ratio of the joint distribution over the product of marginals. However, in our experiments we find the JSD-estimator to be superior.

InfoNCE as our last lower-bound estimator with $N-1$ as the number of negative samples is written as

$$\begin{aligned}\hat{\mathcal{I}}^{(NCE)}(h^a, h^b) &= \mathbb{E}_{(G,R)\sim\mathcal{P}}\left[\mathcal{D}_\omega(h^a, h^b) - \mathbb{E}_{K\sim\mathcal{P}^n}\left[log \sum_{(G',R')\in K\cup\{(G,R)\}} e^{\mathcal{D}_\omega(h^a,(h^b)')}/N \,\middle|\, (G,R)\right]\right] \\ &= \mathbb{E}_{[(G,R),K]\sim\mathcal{P}\times\mathcal{P}^{N-1}}\left[log \frac{e^{\mathcal{D}_\omega(h^a,h^b)}}{e^{\mathcal{D}_\omega(h^a,h^b)} + \sum_{(G',R')\in K} e^{\mathcal{D}_\omega(h^a,(h^b)')}}\right] + log\, N,\end{aligned} \tag{5.6}$$

where K consistst of n random variables that are independent and identically distributed from $\mathcal{P}$, while h^a and h^b are the 2D and 3D hidden representations of (G,R), and $(h^b)'$ is the 3D hidden representation of the molecule (G',R'). This estimator also is related to the Donsker-Varadhan estimator in that it is a lower bound for it $\hat{\mathcal{I}}^{(NCE)}(h^a, h^b) \leq \hat{\mathcal{I}}^{(DV)}(h^a, h^b)$ (Oord, Y. Li, and Vinyals 2018).

We do the actual computation of the InfoNCE objective on batches of size N to estimate the expectations. For every sample, (G_i, R_i) in our batch, the other $N-1$ samples are considered as negative samples of K. To obtain the InfoNCE loss that is optimized in practice, we negate the term and drop $log\, N$ since it is a constant:

$$\mathcal{L}_{InfoNCE} = -\frac{1}{N}\sum_{i=1}^{N}\left[log \frac{e^{\mathcal{D}_\omega(h_i^a,h_i^b)}}{\sum_{k=1}^{N} e^{\mathcal{D}_\omega(h_i^a,h_k^b)}}\right]. \tag{5.7}$$

Usually, when InfoNCE is used, the agreement score of the discriminator is given by the cosine similarity of two representations. In our case this means $\mathcal{D}(h^a, h^b) = sim_{cos}(D^a(h^a), D^b(h^b)) = sim_{cos}(z^a, z^b)$. This way, the final InfoNCE loss in our setup is

$$\mathcal{L}_{InfoNCE} = -\frac{1}{N}\sum_{i=1}^{N}\left[log \frac{e^{sim_{cos}(z_i^a,z_i^b)}}{\sum_{k=1}^{N} e^{sim_{cos}(z_i^a,z_k^b)}}\right]. \tag{5.8}$$

Since the InfoNCE loss as an estimation of a lower bound on the mutual information becomes much more accurate with larger N it is often important to use very large batch sizes. The consequence is that this can lead to memory issues which is one of the main disadvantages of InfoNCE compare to the JSD-estimator which often still performs well when using smaller batch sizes (Hjelm, Fedorov, Lavoie-Marchildon, et al. 2019).

Lastly, we also evaluate the normalized teperature-scaled cross entropy loss (NTXent) introduced by T. Chen, Kornblith, Norouzi, and Hinton 2020 which is a slight variation of the InfoNCE loss but has shown good empirical performance:

$$\mathcal{L}_{NTXent} = -\frac{1}{N}\sum_{i=1}^{N}\left[log \frac{e^{sim_{cos}(z_i^a,z_i^b)/\tau}}{\sum_{\substack{k=1\\k\neq i}}^{N} e^{sim_{cos}(z_i^a,z_k^b)/\tau}}\right]. \tag{5.9}$$

NTXent has two changes compared to the InfoNCE loss. Firstly, the similarity scores are scaled by a temperature parameter $sim_{cos}(z^a, z^b)/\tau$ which can be seen as the weight for the hardest negative sample that is present in the batch. Secondly, the similarity of the positive

pair is dropped from the sum in the denominator. This fits well with the intuitive notion of contrastive learning since, to minimize NTXent, we have to maximize the numerator (the similarity of the positive pair) while minimizing the denominator, meaning all the similarities of the negative pairs.

5.1.2. Bootstrap your own latent

Methods like BYOL or SimSiam (X. Chen and K. He 2020) use Siamese Networks (Hadsell, Chopra, and LeCun 2006) between whose outputs only the similarity of representations from semantically same inputs is enforced. They do not need negative samples and therefore don't suffer from the memory usage drawbacks and the easy-negative issue of contrastive learning. Rather, collapse is avoided by a set of implicit biases like a stop gradient operation or network asymmetry.

BYOL is not as easy to adapt to our problem formulation as the other latent self-supervised learning methods since it uses a teacher and a student network (also called online and target networks) between which the weights are copied. This is not possible if we need to have two different networks where one processes 2D inputs and the other 3D information. However, we devise a new multi-modal BYOL that works for different input modalities.

The method uses four networks and is illustrated in Figure 5.3. A student 2D network f_s^a and a teacher 2D network f_t^a as well as student and teacher 3D networks f_s^b and f_t^b. In this subsection, we will indicate whether a representation was generated by the teacher or student network with subscripts $z_s^a = f_s^a(G)$ and the subscripts do not index the batch dimension (the analog holds for the 3D representations).

At the beginning of training, both 2D networks are initialized with the same weights $\omega_{f_s^a} = \omega_{f_t^a}$ and so are the 3D networks $\omega_{f_s^b} = \omega_{f_t^b}$. We then calculate a 2D loss $\mathcal{L}^a$ and a 3D loss $\mathcal{L}^b$ where each loss takes a student and a teacher representation from the 2D and the 3D side as input. The name of the loss is given by the modality (2D or 3D) of the student representation:

$$\mathcal{L}^a(z_s^a, z_t^b) = -sim_{cos}(z_s^a, z_t^b) \qquad \mathcal{L}^b(z_s^b, z_t^a) = -sim_{cos}(z_s^b, z_t^a), \tag{5.10}$$

where *sim* denotes the cosine similarity as given in Equation 5.1.1. In each optimization step, only the weights of the student networks are updated with the gradient of these losses to minimize them. During backpropagation, the teacher networks are *not* updated, which we refer to as stop-gradient operation. After backpropagation for the students occurred, the weights of the teachers are updated via an exponential moving average of the weights of the student networks. Thus the dynamics of multi-modal BYOL are

$$\omega_{f_s^a} \leftarrow optimizer(\omega_{f_s^a}, \nabla_{\omega_{f_s^a}} \mathcal{L}^a, \eta), \tag{5.11}$$

$$\omega_{f_s^b} \leftarrow optimizer(\omega_{f_s^b}, \nabla_{\omega_{f_s^b}} \mathcal{L}^b, \eta), \tag{5.12}$$

$$\omega_{f_t^a} \leftarrow \gamma\omega_{f_t^a} + (1-\gamma)\omega_{f_s^a}, \tag{5.13}$$

$$\omega_{f_t^a} \leftarrow \gamma\omega_{f_t^a} + (1-\gamma)\omega_{f_s^a} \tag{5.14}$$

where γ is the decay rate of the exponential moving average, η is a learning rate, and *optimizer* is an optimizer such as stochastic gradient descent.

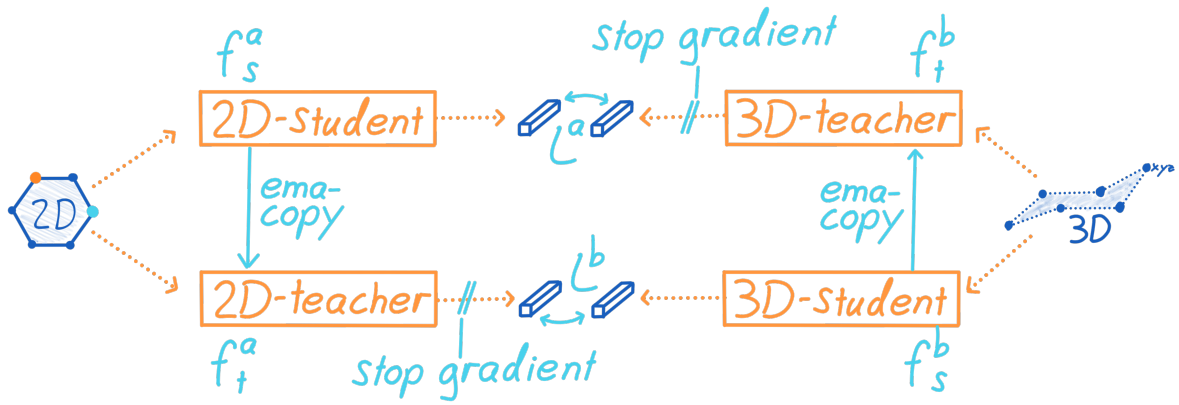


Figure 5.3.: Illustration of our multimodal BYOL architecture.

5.1.3. Barlow Twins

Barlow Twins (Zbontar, Jing, Misra, et al. 2021) is another self-supervised learning method that imposes a loss between latent representations of which we want to enforce that they capture similar information. However, unlike contrastive learning, it does not require negative samples to avoid trivial solutions. Additionally, it does not rely on symmetry-breaking mechanisms like they are employed in SimSiam or BYOL. Rather, collapse is avoided explicitly in the loss function which uses the distance between the identity matrix and the cross-correlation matrix between the outputs of two networks. This way, the diagonal of the cross-correlation matrix causes the representations to be similar. Meanwhile, the off-diagonal terms having to be close to zero decorrelates the feature dimensions of the representations such that the full size of the latent vectors is used to capture information and dimensional collapse (Hua, W. Wang, Xue, et al. 2021) is automatically avoided.

Usually, the two networks in Barlow Twins are identical. However, in our setup, the weights of the 2D Network and the 3D Network cannot be shared and we use an asymmetric architecture. At this point it is important to note, that Zbontar, Jing, Misra, et al. 2021 notice a decrease in performance when introducing asymmetries into their architecture. Besides this our approach is the same meaning that we construct the cross-correlation matrix between the outputs of the 2D and the 3D networks over the batch dimension:

$$\mathcal{C}_{n,m} = \frac{\sum_{i=1}^{N} z^a_{i,n} z^b_{i,m}}{\sqrt{\sum_{i=1}^{N} (z^a_{i,n})^2} \sqrt{\sum_{i=1}^{N} (z^b_{i,m})^2}} \tag{5.15}$$

where $n, m \in 1 \dots d_z$ index the dimensions of the 2D and the 3D representation. This way we end up with a square matrix of values between -1 and 1 with the same dimensionality as the

network outputs. We employ the cross-correlation matrix in the Barlow Twins loss function

$$\mathcal{L}_{BT} = \sum_{n=1}^{d_z}(1 - \mathcal{C}_{nn}) + \lambda \sum_{n=1}^{d_z} \sum_{\substack{m=1 \\ m \neq n}}^{d_z} \mathcal{C}_{nm}^2. \tag{5.16}$$

The first term punishes the diagonal elements of $\mathcal{C}$ if they are different from 1 and therefore similarity between 2D and 3D representations coming from the same molecule is enforced. The second term tries to make the off-diagonal elements closer to 0 such that the components of the embedding vectors are decorrelated and do not contain redundant information.

5.1.4. VICReg: Variance-Invariance-Covariance Regularization

Similar to Barlow Twins, VICReg (Bardes, Ponce, and LeCun 2021) takes a highly principled approach to self-supervised learning and only uses a novel loss function without any constraints on the architectures and without additional tricks to avoid collapse such as the stop gradient operation in BYOL. Moreover, there is also no need for negative samples.

VICReg combines three regularization terms in its objective function. First is a similarity term that rewards a high similarity between 2D and 3D representations if they come from the same molecule, i.e., the similarity is only calculated for the positive pairs. For this, we use the mean-squared error and the embeddings do not have to be normalized onto the unit sphere unlike they often are with the cosine similarity. Recalling that we write $Z^a = [z_1^a, \ldots, z_N^a] \in \mathbb{R}^{N \times d_z}$ for a batch of 2D representations (and analogously for Z^b), our similarity loss is

$$s(Z^a, Z^b) = \frac{1}{N} \sum_{i=1}^{N} \|z_i^a - z_i^b\|_2^2 \tag{5.17}$$

If we were to only optimize this as a loss function, we would run into collapse. The second regularization term is devised to avoid this by enforcing a high variance of the 2D and 3D representations over the batch dimension. This way, a collapse to a constant output would lead to a high loss. To achieve this, a hinge loss is used for Z^a and Z^b that encourages the standard deviation across the batch dimension to be close to 1:

$$v(Z^a) = \frac{1}{d_z} \sum_{n=1}^{d_z} max(0, 1 - std(Z_{:,n}^a)), \tag{5.18}$$

with $Z_{:,n}^a$ being the n-th column of Z^a and $std(x)$ is the unbiased standard deviation given by

$$std(x) = \sqrt{\frac{1}{N-1} \sum_{i=1}^{N} (x_i - \overline{x})^2}, \tag{5.19}$$

where $\overline{x}$ is the mean of x.

The purpose of the last regularization term is to avoid dimensional collapse Hua, W. Wang, Xue, et al. 2021. Like in Barlow Twins, we reduce the redundancy of the information that is captured in the individual components of the representations. However, while in Barlow

Twins the cross-correlation matrix between the 2D and 3D embeddings was used, here we calculate the two correlation matrices:

$$C(Z^a) = \frac{1}{N-1}\sum_{i=1}^{N}(Z_i^a - \overline{Z}^a)(Z_i^a - \overline{Z}^a)^T, \tag{5.20}$$

where $\overline{Z}^a$ is the mean of Z^a over the batch dimension. For both correlation matrices $C(Z^a)$ and $C(Z^b)$ we then force the off-diagonal elements to be close to 0 just like in the second term of the Barlow Twins loss:

$$c(Z^a) = \sum_{n=1}^{d_z}\sum_{\substack{m=1 \\ m\neq n}}^{d_z} C(Z^a)_{n,m}^2. \tag{5.21}$$

To obtain the final loss function, the three regularization terms are added after weighting them with hyperparameters λ, μ, and ν for which we use the same default settings as Bardes, Ponce, and LeCun 2021:

$$\mathcal{L}_{VICReg} = \lambda s(Z^a, Z^b) + \mu[v(Z^a) + v(Z^b)] + \nu[c(Z^a) + c(Z^b)] \tag{5.22}$$

5.1.5. Local-Global Infomax

Instead of maximizing the information that is shared between graph-level representations, we can also do so between 3D node-level representations and a 2D summary vector or the other way around. This is inspired by Deep Graph Infomax (DGI, Velickovic, Fedus, Hamilton, et al. 2019) and Deep InfoMax (DIM, Hjelm, Fedorov, Lavoie-Marchildon, et al. 2019) where the mutual information between global (graph level) and local (node level) representations of the input is maximized.

In this setup, the model producing the global embedding has to learn to generate information that is contained in all the local embeddings. This should be useful since it forces globally relevant information to be captured. Previously, we might have ended up encoding noise that is only present in one local patch of the graph. With the local-global objective, capturing said noise in the global summary vector does not increase the mutual information as much since it does not increase the mutual information between the global representation and any other node representation. The information that is relevant for and contained in all node representations will be favored.

We only describe the case of a global 2D representation and local 3D representations since the setup for the other way around is analogous. For a batch of N molecules where the i-th global 2D representations is z_i^a we write the corresponding local 3D representations as $\{w_{i,j}^b\}_{j\in\{1\ldots n_i\}}$ for the n_i nodes of the i-th graph. These can either be the node embeddings in the last GNN layer or those representations passed through another projection head.

In the DIM framework, the local-global objective is to maximize the average estimated mutual information of each pair of global and local representations. However, this approach can only be used if the number of local representations for each global representation is constant, so if we had a fixed n, such as having the same number of local patches for an image. We need a different strategy since the number of local representations n_i depends on

the molecule. To address this we treat all n_i pairs of global and local representations that come from the same molecule as positive pairs. The negative pairs are all other possible pairs, so the ones where the local representations come from different molecules. For instance, the local-global NTXent objective is Following the DIM framework, any of the previously discussed objectives can then be used to maximize the average mutual information or the similarity between local and global representations. For instance, the local-global objective using the NTXent loss is

$$\mathcal{L}_{NTXent}^{local\text{-}global} = -\frac{1}{N}\sum_{i=1}^{N}\left[log\,\frac{\sum_{j=1}^{n_i} e^{sim_{cos}(z_i^a, w_{i,j}^b)/\tau}}{\sum_{\substack{k=1\\k\neq i}}^{N}\sum_{j=1}^{n_k} e^{sim_{cos}(z_i^a, w_{i,j}^b)/\tau}}\right]. \tag{5.23}$$

5.1.6. Multiple Conformers

For many molecules, there are multiple conformers that have a high probability of occurring. Instead of only using the lowest energy conformer, we want to leverage this additional information during pre-training. The 2D network should not only learn to generate the 3D information of the lowest energy conformer which might not capture the whole picture. The hypothesis is, that if the 2D network encodes the 3D information of multiple conformers in its embeddings, the likelihood increases that the information is useful to inform downstream tasks during fine-tuning.

We propose three approaches to incorporate the 3D information of multiple conformers $\{R_i^j\}_{j\in\{1\dots c_i\}}$. The most straightforward one is **conformer sampling**. We use one of the single conformer setups but when sampling the batch, we additionally sample $j \in \{1\dots c_i\}$ and use the single conformer R_i^j. The probability of sampling a conformer is either distributed uniformly (so $1/c_i$ is the probability for each j) or given by the Boltzmann weight of each conformer.

multi3D is our second approach where we include multiple conformers as additional positive pairs in contrastive learning. 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 now 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. In our contrastive setting, we, therefore, want the similarity between z_i^a and all conformer representations that come from the same molecule $z_{i,j}^b$ to be high. As such we modify the NTXent loss to obtain a multi-3D-NTXent loss

$$\mathcal{L}_{NTXent}^{multi3D} = -\frac{1}{N}\sum_{i=1}^{N}\left[log\,\frac{\sum_{j=1}^{c} e^{sim_{cos}(z_i^a, z_{i,j}^b)/\tau}}{\sum_{\substack{k=1\\k\neq i}}^{N}\sum_{j=1}^{c} e^{sim_{cos}(z_i^a, z_{k,j}^b)/\tau}}\right]. \tag{5.24}$$

One concern with this formulation is the following. Let us consider a single molecule. The objective of high similarity between the many 3D representations and the single 2D

representation might be easier to solve through encoding the same 2D information in the 3D representations instead of capturing the 3D information of all conformers in the single 2D representation. The 2D network would therefore not learn to generate 3D information from its 2D inputs because the mutual information could be maximized through encoded 2D information.

To address this problem, **multi3D+2D** is our third approach. The 2D network is now modified to produce c many latent 2D representations $f^a(G_i) = \{z^a_{i,j}\}_{j\in\{1\dots c\}}$ which are compared to all 3D representations of the same molecule in a similarity function sim. We simply use this similarity in the NTXent loss instead of the cosine similarity. Intuitively, the 2D network now has to generate an embedding for each 3D conformer.

One way to define such a similarity between two same-sized sets of vectors is to use the sum of all pairwise cosine similarities (for brevity we drop the subscript and only write $\{z^a_{i,j}\}$ to mean the set of all representations corresponding to the i-th molecule):

$$sim_{all}(\{z^a_{i,j}\}, \{z^b_{i,j}\}) = \sum_{j=1}^{c}\sum_{k=1}^{c} sim_{cos}(z^a_{i,j}, z^b_{i,k}) \tag{5.25}$$

More principled would be to find the optimal transport matching with the highest cosine similarity, such that one 2D representation always corresponds to one 3D representation. However, this approach was not computationally feasible with the batch sizes we use in contrastive learning. We instead opt for an upper bound on the maximum similarity matching. For every 3D representation, we choose the 2D representation that has the highest similarity. This way one 2D representation could be associated with multiple 3D embeddings and we no longer have a mass preserving matching:

$$sim_{max}(\{z^a_{i,j}\}, \{z^b_{i,j}\}) = \sum_{k=1}^{c} \max_{j\in\{1\dots c\}} sim_{cos}(z^a_{i,j}, z^b_{i,k}). \tag{5.26}$$

Beyond these similarity measures, we explore additional ones based on the inverse of different distance functions and asymmetric metrics such as the maximum mean discrepancy (Gretton, Borgwardt, Rasch, et al. 2012) or the KL- and JS-Divergence when interpreting the conformer representations as samples from a normal distribution. Since these did not yield promising results, they are only described and presented in Appendix A.1.

5.1.7. 3D Network

The 3D network f^b should take a set of 3D coordinates R as input and produce a single vector representing this conformer z^b. The representation that it produces should be invariant with respect to the transformations in $SE(3)$ as described in Section 2.1.3. These are the symmetries of molecules and leveraging 3D information with this prior has been shown to yield large improvements for molecular property prediction since it is more sample efficient and leads to better generalization (Schütt, Kindermans, Sauceda, et al. 2017; Klicpera, Groß, and Günnemann 2020; Klicpera, Giri, Margraf, and Günnemann 2020; Y. Liu, L. Wang, M. Liu, et al. 2021; Satorras, Hoogeboom, and Welling 2021; Klicpera, Becker, and Günnemann 2021).

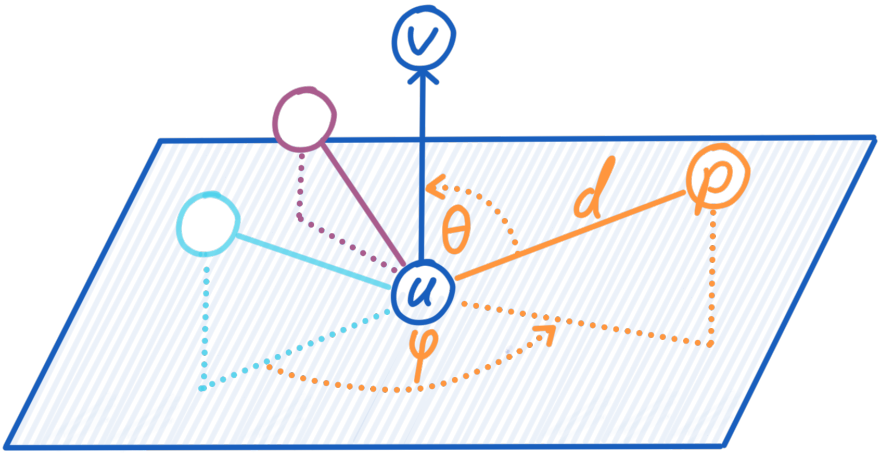


Figure 5.4.: Illustration of the three spherical coordinates used in SMP to specify the relative position of node p with respect to the bond (u, v). (1) The distance d between u and p. (2) The bond angle $\theta = \angle vup$. (3) The angle ϕ between (u, p) and the first bond that is encountered when moving clockwise around (u, v) after the bonds were projected onto the hyperplane that is perpendicular to (u, v).

The architectures that we use are GNNs that incorporate 3D information in different ways and were previously used on molecules. Note that while we discuss the $SE(3)$ or $E(3)$ invariance of our 3D networks, they are all also permutation equivariant as other GNNs (Bronstein, Bruna, Cohen, and Velickovic 2021). We have these models operating on a learnable embedding vector as the node and edge features. The values of this vector are initialized with a standard normal distribution. The 3D network does not take any 2D information like the adjacency matrix, or the actual node and edge features as input. Otherwise, the model might already achieve a high amount of shared information or high mutual information between the latent representations by just encoding the 2D information. The contrastive objectives may be nearly perfectly solved by encoding 2D information and the 2D model would not have to learn any 3D information. We mainly evaluate different versions of the following three architectures.

Spherical Message Passing

SMP by Y. Liu, L. Wang, M. Liu, et al. 2021 is the first 3D network that we employ which is originally used for quantum mechanical property prediction with 3D information as input. In this GNN, the 3D information is used in the form of three $SE(3)$ invariant quantities that are aggregated when creating a message, which are bond lengths and two types of angles. That is to say, the message for edge (u, v) is constructed by aggregating the relative locations of all of u's neighbors $N(u)$ which are specified by spherical coordinates with respect to the edge. The neighborhoods are given by the radius graph where atoms are connected with an edge if their distance is under a threshold. The relative position of an atom $p \in N(u)$ is

identified by the Euclidean distance d between v and p as well as the angle θ between the edges/bonds (u, v) and (u, p), that is $\theta = \angle vup$. Lastly, for the second angular coordinate ϕ, we consider all of the bonds (u, j) with $j \in N(u)$ which are connected to u and project them onto a plane that is perpendicular to (u, v) (except for (u, v)). More precisely, the vectors $\{r_j - r_u\}_{\{j \in N(u) \wedge j \neq v\}}$ are projected onto the hyperplane for which $r_v - r_u$ is the normal vector. Then ϕ, is given by the angle between the projection of (u, p) and the first other projection that is encountered when moving clockwise around $r_v - r_u$ on the plane. The three spherical coordinates are also described in Figure 5.4.

When creating the message for (u, v), we aggregate the spherical coordinates of all of u's other neighbors which is all of the 3D information used in SMP. It is important to realize that this does not uniquely define the relative location of each atom in the molecule unlike what is claimed in Y. Liu, L. Wang, M. Liu, et al. 2021. This can be shown via a counterexample as depicted in Figure 5.5 where we have two different, non-symmetric, conformations of the same molecule but all of the spherical coordinates are the same for both conformations. The issue is that SMP does not capture torsion/dihedral angles which is the relative rotation around a bond of two substructures connected by that bond. No matter what the torsion angle is, the spherical coordinates used in SMP will always stay the same and our model cannot use the whole 3D geometry of the molecule to generate its representation.

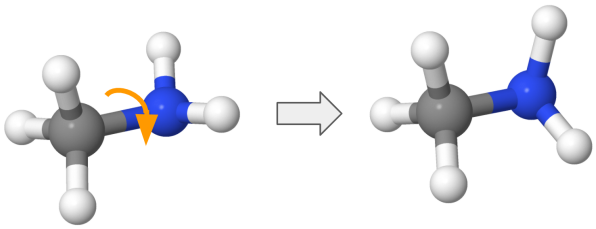


Figure 5.5.: Counterexample to prove that SMP's spherical coordinates do not uniquely define the relative positions of all atoms in a molecule. We have two different non-symmetric conformations with the same spherical coordinates. The two conformations differ in their torsion angle since the blue nitrogen and its two hydrogens were rotated by 60° around the bond as indicated by the orange arrow.

Note that SMP is not invariant with respect to reflections of a conformation and the approach has the desirable property of only being $SE(3)$ invariant. Therefore SMP is able to distinguish most molecules that only differ in their chirality which is an important advantage since some biological and pharmacological molecular properties depend on the chirality. Therefore, this is valuable information that can be used during fine-tuning if chirality annotations are present in the features of the 2D information.

The non-invariance to reflections is due to the ϕ angles that are almost always different between mirror images of a conformation if those mirror images are not symmetric in $SE(3)$ (that is if they are actually chiral/handed). This is the case because the order in which we encounter the bond projections onto the hyperplane when moving around the main bond in clockwise fashion changes depending on the chirality. The small caveat why there are, in

theory, some conformations for which SMP cannot tell mirror images apart is that the order of the bond projections could change without the ϕ angles changing. For instance, a tetrahedral structure with four different components connected to a central carbon where all angles are the same (this cannot happen in practice since four different components connected to the carbon cannot exhibit a perfect tetrahedron).

Net3D

We found that downstream performance could be increase with a second 3D network that we devised ourselves which we describe here and call it **Net3D**. It encodes the 3D information given by the pairwise Euclidean distances of atoms. Therefore, the method is $E(3)$ invariant (proof in Appendix B.1) and is unable to capture a notion of chirality which is a disadvantage compared to SMP. However, the positions of all atoms are uniquely defined up to $E(3)$ symmetry. There is no loss of that relevant 3D information and we can distinguish conformations with different torsion angles which is a large advantage over SMP. 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, Baratin, Arpit, 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, Srinivasan, Mildenhall, et al. 2020 who show empirically and theoretically that 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

$$\gamma(d_{uv}) = (d_{uv}, sin(d_{uv}/2^0), cos(d_{uv}/2^0), \ldots, sin(d_{uv}/2^{F-1}), cos(d_{uv}/2^{F-1})), \tag{5.27}$$

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, Shazeer, Parmar, 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, Srinivasan, Tancik, et al. 2020 in image synthesis and by Zhong, Bepler, Davis, and Berger 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 $d_{uv}^0 = MLP_{init}(\gamma(d_{uv}))$. The second input is a set of n atom representations $\{h_1^l, \dots h_n^l\}$ 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:

$$m_{uv} = MLP_{edge}([h_u^l \parallel h_v^l \parallel d_{uv}^l]) \tag{5.28}$$

$$d_{uv}^{l+1} = d_{uv}^l + m_{uv} \tag{5.29}$$

$$h_u^{l+1} = MLP_h([h_u \parallel \sum_{\substack{v=1 \\ v \neq u}}^{n} m_{uv} * sigmoid(MLP_{softedge}(m_{uv})]). \tag{5.30}$$

The layer is paremeterized 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, Hoogeboom, and Welling 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 $MLP : \mathbb{R}^{3d_h} \mapsto \mathbb{R}^{d_z}$.

The decision for this architecture to encode the pairwise distances is motivated by the goals of (1) flexibility in the number of parameters and expressivity as well as (2) having node/atom representations that we can use for the local-global objectives described in Section 5.1.5. We want flexibility in expressiveness to address our concern of the mutual information in our latent representations being maximized by the 3D network generating and encoding 2D information instead of the 2D network learning to encode 3D information. If this is the easier approach to maximize the mutual information, our 2D network would not capture 3D information in its layers, and using it during fine-tuning fails. We wish to investigate if the impact of the 3D network's expressiveness on this issue. The idea is that a very small 3D network might not be able to infer 2D information from its inputs and can only fulfill the easier task of encoding the directly available 3D information. This way the 2D network is forced to generate 3D information since just encoding its directly available 2D information won't contribute to maximizing the mutual information.

E(n)-Equivariant Graph Neural Networks

EGNNs (Satorras, Hoogeboom, and Welling 2021) were originally used for multiple applications that benefit from $E(n)$ equivariance or invariance, including quantum mechanical property prediction with 3D inputs. EGNN operates on a radius graph where atoms are connected by an edge if their distance is below a threshold. However, (Satorras, Hoogeboom, and Welling 2021) obtained good results in experiments on small molecules when setting the

threshold so high that EGNN was using a complete graph for every molecule in their dataset. Following this, we use the complete graph for all molecules. In this setting, EGNN uses all pairwise distances as 3D input and therefore comes with the same invariance properties as Net3D and also has information that uniquely defines all relative positions of the atoms up to $E(3)$ transformations.

Similar to the other 3D networks we use a learned vector as initial node features. Throughout its layers, EGNN uses the pairwise distances as edge features while doing message passing on the complete graph to end up with node representations that are pooled and projected to the size of our latent 3D representation. The only changes we make to the original EGNN is adding batch normalization (Ioffe and Szegedy 2015) since we found training to be unstable otherwise (also when reproducing their experiments on molecules).

5.2. Predictive 3D Self-Supervision

Instead of our latent space 3D pre-training methods, it would arguably be much simpler to have the network directly predict some explicit form of 3D information such as the pairwise distances between the atoms. This way, the model also has to capture latent 3D information in its hidden representations. After transferring the weights, the 3D information is generated in the hidden layers and can be used to inform downstream property predictions. Since such a predictive pre-training method is more straightforward than the previously presented latent space SSL, it would be preferred if both perform equally well and is, therefore, an important baseline.

As such, we devise two predictive SSL methods where the first one predicts all the pairwise distances between atoms. The second method predicts different local angles and distances from which the whole molecule can be constructed. Additionally, the second method generates multiple possible sets of these 3D predictions that are matched to the molecules conformers with an optimal transport loss. Since we only have a single model during pre-training we now discuss it as a GNN $f(G) = \{h_v\}_{v \in \mathcal{V}}$ that takes a 2D molecular graph G as input to produce a set of atom representations $\{h_v\}_{v \in \mathcal{V}}$ with $h_v \in \mathbb{R}^{d_h}$.

5.2.1. Distance Predictor

In this method to pre-train f, its atom representations $h_{v v \in \mathcal{V}}$ are used to predict all ℓ_2-distances between the atoms. To predict the distance between node v and u, we concatenate their representations and feed them to an MLP $MLP : \mathbb{R}^{2d_h} \mapsto \mathbb{R}$ that produces a single scalar. The distance prediction $dist_{uv}$ is then given by

$$dist_{uv} = softplus(MLP(h_v \parallel h_u) + MLP(h_u \parallel h_v)) \tag{5.31}$$

where $\parallel$ denotes concatenation and $softplus(x) = log(1 + e^x)$. The node representations are concatenated in both orders and fed to the MLP to ensure that the function is symmetric. The final loss to pre-train f is the mean squared error between the predicted distances and the

ℓ_2-distances of all atom pairs (see Appendix A.1 for other approaches that we tried):

$$\mathcal{L}_{DistPred} = \frac{1}{|\mathcal{V}|^2} \sum_{v \in \mathcal{V}} \sum_{u \in V} (\|r_u - r_v\|_2 - dist_{uv})^2 \tag{5.32}$$

5.2.2. Optimal Transport Prediction

While during inference the conformations generated by learned conformation generation methods might be too inaccurate to inform downstream molecular property predictions, we could still use one of the methods for pre-training. A GNN that is used in a conformation generation pipeline will have to encode 3D information in its hidden layers. As such, we reuse parts of the SOTA conformation generation method GeoMol (Ganea, Pattanaik, Coley, et al. 2021) to pre-train a GNN and include the information of multiple conformers with an optimal transport loss.

In this approach, we first use the node representations produced by f to predict all bond distances and the distances between atoms that are two hops apart in the molecular graph (nodes that are connected by a path of length 2 which we will call 2-hop distances). The distances are predicted with the same method as the Distance Predictor approach does it in Equation 5.31. Given the 2-hop distances, we can calculate all bond angles $\angle vup$ for all combinations of edges $(v,u),(u,p) \in \mathcal{E}$ (Figure 5.4 is illustrative).

After predicting the 1-hop and 2-hop distances and calculating the bond angles, the torsion angles are predicted, which uniquely defines the relative positions of all atoms. The torsion angles are predicted by another MLP and are used to calculate an angle $\angle vupq$ for every chain of three edges $(v,u),(u,p),(p,q) \in \mathcal{E}$. The angle $\angle vupq$ is the prediction for the angle between two planes, one plane contains the edges (v,u) and (u,p) and the second contains the edges (u,p) and (p,q).

With all relative atom positions defined, the 3-hop distances are additionally calculated. This results in 5 different 3D quantities summarized as a 3D description R_{pred} which contains the 1-hop, 2-hop, 3-hop distance predictions, and the torsion and bond angle predictions. This is used in the following loss term for a single conformer R^j where the ground truth 3D quantities calculated from R^j are marked with *:

$$\begin{aligned}
\mathcal{L}(R_{pred}, R^j) = & \frac{1}{|\{(v,u) \in \mathcal{E}\}|} \sum_{(v,u) \in \mathcal{E}} (dist_{vu} - dist^*_{vu})^2 \\
& + \frac{1}{|\{(v,u),(u,p) \in \mathcal{E}\}|} \sum_{\{(v,u),(u,p) \in \mathcal{E}\}} (dist_{vp} - dist^*_{vp})^2 \\
& + \frac{1}{|\{(v,u),(u,p),(p,q) \in \mathcal{E}\}|} \sum_{\{(v,u),(u,p),(p,q) \in \mathcal{E}\}} (dist_{uq} - dist^*_{uq})^2 \\
& - \frac{1}{|\{(v,u),(u,p) \in \mathcal{E}\}|} \sum_{\{(v,u),(u,p) \in \mathcal{E}\}} cos(\angle vupq - \angle^* vupq) \\
& - \frac{1}{|\{(v,u),(u,p),(p,q) \in \mathcal{E}\}|} \sum_{\{(v,u),(u,p),(p,q) \in \mathcal{E}\}} cos(\angle vupq - \angle^* vupq)
\end{aligned} \tag{5.33}$$

To leverage the information of multiple conformers in an optimal transport loss, K many randomly different 3D descriptions are predicted. To produce the different predictions, a standard normal distributed random vector is concatenated to the node features of G before calculating the node embeddings with $f(G)$. The K predicted 3D descriptions $\{R^k_{pred}\}_{k\in\{1,\dots,K\}}$ are then matched up with the c conformers $\{R^j\}_{j\in\{1,\dots,c\}}$ such that the smallest loss is possible but the weight is preserved. This is done with an optimal transport loss which is the final pre-training objective:

$$\mathcal{L}_{OptimalTransport} = \min_{T\in\Gamma_{K,L}} \sum_{k=1}^{K}\sum_{j=1}^{c} T_{kj}\mathcal{L}(R^k_{pred}, R^j) \tag{5.34}$$

where T is the transport plan with the optimal matching satisfying the constraints given by $\Gamma_{K,L} = \{T \in \mathbb{R}_+^{K\times c} \mid T\mathbb{1}_c = \frac{1}{K}\mathbb{1}_K \wedge T^T\mathbb{1}_K = \frac{1}{c}\mathbb{1}_c\}$ with $\mathbb{1}_c$ being an all ones vector of size c. The transport plan is calculated with the Earth Movers distance problem solved by the Python Optimal Transport library (Flamary, Courty, Gramfort, et al. 2021).

6. Experimental Evaluation

In this chapter, we present empirical evidence for the value of pre-training GNNs with 3D information of molecules for better downstream molecular property predictions as well as an evaluation of different pre-training approaches and their impact on the performance. Multiple experiments aim at understanding the information captured in the pre-trained model's representations. While this is the case, the main goal of this section is to determine what method helps the most, for which molecular property prediction task, and how to achieve the largest improvements in real-world applications.

In the first Section 6.1, we lay out the datasets used for evaluation, explain the motivation for the choices, and discuss their relevance in real-world applications. Additionally, we give an overview of the used models, present the evaluation metrics, and the used software and hardware. In Section 6.2 we present the results using latent space self-supervision methods and compare them against our two predictive approaches. There we evaluate the approaches described in Section 5.1 and the effect of different architecture choices in each setting. Besides knowing that the 3D pre-training schemes yield an improvement over random initialization, it is important to compare them against conventional pre-training. Additionally, we investigate the benefits to be gained from using different approaches to leverage the information of multiple conformers. Lastly, the Section 6.2 also includes an analysis of failure modes of the methods such as dimensional collapse or collapse to a trivial solution.

We briefly summarize our most important empirical findings as follows:

- 3D pre-training leads to large improvements for all 10 quantum mechanical property prediction tasks that we tested.
- For 10 biophysical molecular property prediction datasets, 3D pre-training either improved performance or was on par with training from scratch. 3D pre-training never decreased downstream performance which sometimes happens with conventional pre-training methods for molecules.
- The information captured during 3D pre-training is universally useful and generalizes across highly different datasets. The pre-training and fine-tuning datasets can be of very different compositions and we still observe large improvements in many settings.
- Leveraging the information of multiple conformers yields more informative representations and using low energy conformers is more important than a high diversity.
- Our best method is latent space SSL with NTXent loss, multiple conformers, and the *multi3D* objective $\mathcal{L}^{multi3D}_{NTXent}$.

6.1. Experimental Setup

This section first details the hardware and software used for the implementation and the datasets used for pre-training or inference. Then, it explains the GNN we pre-train along with the baseline and the metrics used to validate the approach.

6.1.1. Implementation

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

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 and all hyperparameters and training settings are detailed in Appendix A.2.

6.1.2. Datasets

Here, we discuss the datasets used for both the pre-training and the transfer learning. In general, the datasets containing 3D information are used for pre-training, but they also contain some quantum mechanical properties that are interesting to evaluate for transferability. For the other tasks, we focus on properties related to drug-discovery problems, such as virus inhibition or blood-brain-barrier permeability.

Our proposed method consists of a pre-training step with different data than the following fine-tuning to make downstream predictions. We use three datasets containing 3D information for pre-training with diversity in molecule size, and the number of molecules as can be seen in Table 6.1. The pre-training datasets are:

1. **QM9**[1] (Ramakrishnan, Dral, Rupp, and Lilienfeld 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 B.2 in Appendix B.2 where

[1] `https://github.com/klicperajo/dimenet/blob/master/data/qm9_eV.npz`
[2] `https://github.com/learningmatter-mit/geom`

Table 6.1.: 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
esol	1128	13.3	13.7	scaffold
lipo	4200	27.0	29.5	scaffold
freesolv	642	8.7	8.4	scaffold
bace	1512	34.1	36.9	scaffold
bbbp	2039	24.1	26.0	scaffold
hiv	41 127	25.5	27.5	scaffold
tox21	7831	18.6	19.3	scaffold
toxcast	8576	18.8	19.3	scaffold
clintox	1477	26.2	27.9	scaffold
sider	1427	33.6	35.4	scaffold

we also provide a histogram for the number of molecules that have a certain amount of conformers B.3.

3. **QMugs**[3] (Isert, Atz, Jiménez-Luna, and Schneider 2021) has 665k drug-like molecules with three diverse conformers each and multiple conformer specific quantum mechanical properties as regression tasks.

For fine-tuning, we use a variety of datasets that cover a wide range of domains and applications. The molecular properties are relevant for quantum mechanics, physical chemistry, biophysics, and physiology such that we can obtain a good estimate of how valuable our 3D pre-training is for each domain. For quantum mechanical properties, which are often specific to a conformer, it is clear that 3D information is important and there has been a lot of evidence that learned methods highly benefit from its use (Klicpera, Groß, and Günnemann 2020; Klicpera, Giri, Margraf, and Günnemann 2020; Y. Liu, L. Wang, M. Liu, et al. 2021; Schütt, Kindermans, Sauceda, et al. 2017). For these properties, the interest is in how much our method can leverage this information and transfer it to molecules where no 3D geometry is available.

Meanwhile, for biological or physiological properties such as blood-brain barrier penetration, it is not as clear if improvements from 3D information are to be expected. As such, this question needs to be answered next to how much of the benefits 3D pre-training recovers. For this purpose, we use the following molecular graph datasets which are mainly taken from MoleculeNet (Z. Wu, Ramsundar, Feinberg, et al. 2017) and we use the scaffold splits[4] with

[3] https://www.research-collection.ethz.ch/handle/20.500.11850/482129

[4] https://ogb.stanford.edu/docs/graphprop

an 80/10/10 split ratio provided by Open Graph Benchmark (OGB) W. Hu, Fey, Zitnik, et al. 2020. The fine-tuning datasets are:

1. **QM9 and GEOM-Drugs:** On these 3D datasets we also fine-tune and evaluate the quantum mechanical properties of one half of the datasets with a random split. This is done after either pre-training on another 3D dataset (generalization), or after pre-training on the other half of the same dataset (in distribution).
2. **ESOL:** 1128 common organic small molecules with water solubility data (log solubility in mols per liter).
3. **Lipo:** Experimental data for the octanol/water distribution coefficient of 4200 molecules.
4. **FreeSolv:** The hydration free energy of 642 molecules in water.
5. **HIV:** 41k molecules with binary labels for HIV virus replication inhibition.
6. **BACE:** Binary labels of binding results for inhibitors of human β-secretase 1 for 1512 molecules.
7. **BBBP:** 2039 molecules with binary labels for blood-brain barrier penetration.
8. **Tox21:** 7831 molecules with binary labels of their toxic for 12 different targets.
9. **ToxCast:** 8576 molecules with binary labels of toxicity experiment outcomes with 617 targets.
10. **SIDER:** 1427 approved drugs with 27 different side effect groups and the task is to predict whether the drug is in the side effect group.
11. **ClinTox:** 1477 drugs with two binary annotations where the first is to predict toxicity in clinical trials and the second is the FDA approval status.

The reason why *muv* and *pcba* are the only datasets from the OGB benchmark suite which we omit is their larger size. The features we use to encode the molecules (e.g., atom-type, chirality ...) are those chosen by the OGB (W. Hu, Fey, Zitnik, et al. 2020) benchmarking pipeline to ensure comparability with other work. This featurization includes chirality and our methods could thus make use of how chiral information impacts the geometry if the 3D network is not invariant to reflections.

6.1.3. Models

We pre-train Principal Neighborhood Aggregation (PNA) Corso, Cavalleri, Beaini, et al. 2020 due to its simplicity and since it achieved and is close to the SOTA for multiple molecular tasks. PNA is highly similar to the first GNN presented in the MPNN framework by Gilmer, Schoenholz, Riley, et al. 2017. PNA differs in its $AGGREGATE$ function where multiple permutation invariant functions are used whose outputs are scaled and concatenated before being fed through an MLP.

To obtain the hyperparameters and aggregator functions that we use in our fine-tuning experiments on QM9 and GEOM-Drugs, we evaluate different settings on QM9's energy of the highest occupied molecular orbital (homo) property and the ensemble Gibbs free energy of GEOM-Drugs. The exact search space, final architecture, and optimization procedure is detailed in Appendix A.2.2. In short, the aggregators are taking the maximum, mean,

minimum, and standard deviation over the incoming messages, the hidden dimension is 200, and we have seven message passing layers. Our PNA additionally differs from the original in that it also uses multiple aggregators in the $READOUT$ function which are taking the maximum, mean, minimum, and sum over all node representations.

The **3D Networks** are SMP (Y. Liu, L. Wang, M. Liu, et al. 2021), EGNN Satorras, Hoogeboom, and Welling 2021 and Net3D as presented in 5.1.7. If not mentioned otherwise, we use Net3D with the parameters specified in Appendix A.2.2.

6.1.4. Conventional pre-training Baseline

The conventional pre-training method that we compare against is GraphCL (You, T. Chen, Sui, et al. 2020) since it is the work with the strongest results in a consistent comparison for molecular tasks. GraphCL uses the usual self-supervised objective where the model has to learn to produce representations that are invariant with respect to augmentations. You, T. Chen, Sui, 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.

As a conventional predictive pre-training approach, we additionally implemented the masked language modeling (Devlin, Chang, Lee, and Toutanova 2019) approach ELECTRA (Clark, Luong, Le, and Manning 2020). However, we omit it from the results since it performed worse than GraphCL on all biophysical tasks. The method is described in Appendix A.1.

6.1.5. Evaluation Metrics

For quantum mechanical property prediction tasks, we report the Mean Absolute Error (MAE); the lower, the better. Meanwhile, we adopt the standard evaluation protocol of W. Hu, Fey, Zitnik, et al. 2020 for the MoleculeNet (Z. Wu, Ramsundar, Feinberg, et al. 2017) datasets. As such, the metric for the regression datasets is the Root Mean Squared Error (RMSE) where lower values are better. For the binary classification tasks, the area under the curve of the area under the curve of the Receiver Operator Characteristic (ROC-AUC); the higher the better. The units of the regression tasks are given in Appendix A.2.1. All downstream performance metrics are obtained from held-out test sets and the pre-training and fine-tuning datasets are always disjoint. Lastly, specified otherwise, when reporting standard deviations, the mean and the standard deviation are always calculated from six different seeds during fine-tuning and for pre-training, we only use a single seed.

6.2. Results

In this section, we first compare our best latent space self-supervision method against different baselines and investigate its generalization capabilities for quantum mechanical and biophysical properties. We then compare predictive SSL with the latent space SSL approach. Next, we show results for the different methods, losses, and 3D networks. Furthermore,

we investigate when our latent SSL methods experience collapse while investigating the information contained in the latent representations. We then assess the impact of using multiple conformers, the pre-training dataset size, the 3D network's expressivity, and the pre-training time. The first of the following subsections describes its experiment in more detail while the other subsections omit details if the setup is the same to avoid repetition.

6.2.1. Main Results for Quantum Mechanical Properties

For quantum mechanical tasks, we know that 3D information is beneficial and therefore expect a working 3D pre-training to improve downstream performance. We additionally wish to answer how the gains compare to (1) explicitly using the 3D information, (2) pre-training by predicting properties, and (3) pre-training with conventional contrastive learning. Lastly, we are interested in our methods generalization abilities and the performance changes when pre-training and fine-tuning on different datasets with highly different molecules.

For this purpose, we pre-train PNA with our strongest method on QM9, GEOM-Drugs, and QMugs. This is then compared against no pre-training, pre-training with GraphCL, pre-training by predicting properties, and SMP. More precisely, we (1) pre-train PNA on 50 000 molecules from QM9 with NTXent using a single conformer (labelled *QM9 3D*). (2) Another instance of PNA is pre-trained on 140 000 molecules of GEOM-Drugs with 5 conformers using the *multi3D* version of NTXent $\mathcal{L}_{NTXent}^{multi3D}$ (labeled *Drugs 3D*). (3) PNA is pre-trained on 620 000 molecules of QMugs using all three conformers with $\mathcal{L}_{NTXent}^{multi3D}$ (labeled *QMugs 3D*). The pre-training baselines are (4) predicting the properties of GEOM-Drugs' pre-training subset (labeled *Prop Pred*) and (5) GraphCL as a conventional contrastive learning approach which we pre-train on all of GEOM-Drugs. All pre-training methods use a batch size of 500.

After pre-training, the models are fine-tuned on the 50 000 molecules from QM9 or 140 000 from GEOM-Drugs that have no overlap with the molecules from the pre-training data. The fine-tuning tasks are QM9's quantum mechanical properties or those of GEOM-Drugs. On the same molecules used for fine-tuning, we also train PNA with random weight initialization (labelled *Rand Init*) to compare how much the downstream performance is improved by the different pre-training methods. Additionally, we train SMP (Y. Liu, L. Wang, M. Liu, et al. 2021) on the same subset as a method that uses 3D information explicitly. The hyperparameters for SMP are taken from the official repository[5] where Y. Liu, L. Wang, M. Liu, et al. 2021 provide their code and we predict *gap* even though it could be calculated as $|homo - lumo|$.

Table 6.2 shows that large improvements over the randomly initialized baseline and over GraphCL are possible with 3D pre-training with all pre-training datasets. After 3D pre-training on one half of QM9 the predictions for homo can be improved by 16 percentage points over the random initialization baseline. The improvement is 13 percentage points when pre-training with the highly different molecules from GEOM-Drugs (e.g., on average 18 atoms vs. 44.4 atoms). 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 always is better with a large margin.

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

Table 6.2.: Comparison of our strongest 3D pre-training method against different baselines for predicting QM9's properties. The numbers show the MAE for predicting QM9's properties (lower better) after pre-training or of a randomly initialized baseline (***Rand Init***). We 3D pre-train either with GEOM-***Drugs***, ***QMugs*** or a disjoint half of ***QM9*** as 3D datasets. The pre-trained baselines are ***GraphCL*** and predicting properties (***PropPred***). ***SMP*** is a 3D GNN that has access to the explicit 3D information which the other methods do not use. Only the standard deviation of homo is estimated with six seeds, the rest uses four. 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

We can also observe that mostly pre-training with the disjoint half of QM9 (column QM9 3D) performs the best which makes sense because the pre-training data and the fine-tuning data come from the same distribution. However, the improvements when pre-training with GEOM-Drugs and QMugs are almost similarly large even though the pre-training data is vastly different from the fine-tuning data in these cases. QM9 contains very small molecules with on average 18 atoms while the molecules of, e.g., GEOM-Drugs contain 44.4 atoms on average and are drug-like molecules.

Our results seem to verify that the 2D network learns to capture 3D information but we cannot entirely rule out that the downstream improvements are instead due to learning some computational primitives or other valuable information during pre-training. This seems unlikely in light of the large improvements compared to, for instance, GraphCL.

We further note that learning the 3D structure implies that the model learns about quantum mechanical interactions and Van der Waals forces. Such knowledge is necessarily transferable across tasks since it is the foundation of how molecules interact in a biological system.

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 and used explicitly. One factor that has to be considered for this comparison is that QM9's properties are of the molecules' specific conformers. There might be a maximum accuracy that can be 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 the most promising future directions in Section 7.1.

Table 6.3.: Comparison of our strongest 3D pre-training method with different pre-training datasets. The downstream tasks are predicting GEOM-Drugs' Gibbs free energy ***Gibbs*** and the average conformational energy $\langle E \rangle$ and we show the MAE (lower better). 3D pre-training substantially improves the property predictions compared to the randomly initialized baseline.

property	Gibbs (kcal/mol)	$\langle E \rangle$ (kcal/mol)
Rand Init	.2035$\pm$.0011	.1026
GraphCL	.1941	.0995
QM9 3D	.1852	.0968
Drugs 3D	**.1811**	**.0952**
QMugs 3D	.1835	.0965

The results in Table 6.3 further confirm that quantum mechanical property predictions can substantially be improved with 3D pre-training. Again, the case where transfer happens in-distribution and we pre-train on one half of GEOM-Drugs and fine-tuning on the other (row Drugs 3D) is the one with the lowest MAE. Also, as with the properties of QM9, 3D pre-training wins out on GraphCL no matter what dataset is used for 3D pre-training. This is the case even though GraphCL was pre-trained on all molecules of GEOM-Drugs including the fine-tuning subset. Meanwhile, the 3D methods have never seen the fine-tuning data during pre-training.

However, the most striking results are the large improvements from .2035 MAE with random initialization 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 and the drug-like molecules are much larger (44.4 atoms on average vs. 18 as can be seen in Table 6.1).

As such, there is a small difference in performance between pre-training with GEOM-Drugs and QM9 in Table 6.3. This could suggest that our method only learns what influence an atom has on a molecule's 3D geometry for a subset of atoms that exist in both datasets (such as C, H, and O). Intuitively speaking, the pos-hoc hypothesis is that the 2D network does not understand how the other atoms impact the structure of the molecule and misses their crucial influence such that the method could be further improved by addressing this issue.

6.2.2. Main Results for Biophysical Properties

In the previous section, we found that 3D pre-training can yield large improvements for predicting quantum properties. In this section, we determine the improvements for biophysical properties. For non-quantum properties, there has not been as much empirical evidence that 3D information is be informative and allows for more accurate predictions. However, for tasks such as binding prediction in the *bace* dataset, it should theoretically be helpful.

The biophysical properties we use for fine-tuning are from the MoleculeNet datasets (Z. Wu, Ramsundar, Feinberg, et al. 2017) described in Section 6.1.2 with the evaluation protocol

of OGB (W. Hu, Fey, Zitnik, et al. 2020). We pre-train our *multi3D* version of NTXent (labeled *3D SSL*) and the GraphCL baseline on all of GEOM-Drugs.

This experiment is the only one where we use different hyperparameters for PNA since the smaller datasets are easily overfitted on with the large architecture we use for the quantum mechanical properties. A smaller PNA yields a better performance with the random initialization baseline. Therefore the PNA in these experiments has a hidden dimension of 50 and 3 message passing layers as propagation depth. Apart from that, it is the same as the PNA described in Appendix A.2.2.

Table 6.4.: Comparison of our strongest 3D pre-training method (trained on GEOM-Drugs and labeled ***3D SSL***) against the random initialization (***Rand Init***) and ***GraphCL*** baselines for various biophysical property OGB datasets. Shown is either the RMSE indicated by ↓ where lower values are better or the ROC-AUC indicated by ↑ where higher values are better. Colors indicate improvement, worse performance, or no change compared to the baseline. 3D pre-training is either on par with random initialization or better. There is no negative transfer as there is with GraphCL.

dataset	Rand Init	GraphCL	3D SSL
esol↓	0.947±0.038	0.959±0.047	0.894±0.028
lipo↓	0.739±0.009	0.714±0.011	0.695±0.012
freesolv↓	2.233±0.261	3.744±0.292	2.337±0.227
bace↑	78.13±1.30	77.18±4.01	79.42±1.94
bbbp↑	68.27±1.98	71.06±2.00	69.10±1.07
tox21↑	73.88±0.64	78.92±0.61	74.46±0.74
toxcast↑	63.62±0.48	64.95±0.31	64.41±0.88
clintox↑	58.98±5.40	51.07±5.52	59.43±3.21
sider↑	55.95±3.27	57.32±5.00	53.37±3.34
hiv↑	77.06±3.16	76.06±1.06	76.08±1.33

In Table 6.4 we see that our 3D pre-training method always either improves over the random initialization baseline or performs similarly. For some datasets there are large improvements such as a reduction in RMSE from 0.739 ± 0.009 to 0.695 ± 0.012 for *lipo*. The datasets for which the biggest performance gains can be observed are not necessarily those for which we would expect them to be the largest based on our chemical understanding. For instance, for *bace* the binding prediction task of *bace* we would expect 3D information to be especially informative but the better ROC-AUC of our method is not outside of the interval given by one standard deviation. Similarly, for *bbbp* blood-brain barrier penetration has to be predicted for which 3D information would presumably be highly relevant and the improvements are only marginal. Meanwhile, the improvements on *esol* and *lipo* are more significant even though we would expect lipophilicity and solubility to be mainly determined by functional groups of the molecule and these do not depend on the geometry.

When comparing our 3D SSL method with GraphCL we find that conventional pre-training with GraphCL offers larger improvements for some datasets. However, it also suffers from

negative transfer (S. J. Pan and Q. Yang 2010) and sometimes performs worse than the baseline. 3D pre-training does not have this problem. In practice this is highly relevant and it might be better to have a method that reliably either increases performance or keeps it the same than a method that can increase performance to a larger extent but also decreases it in some cases.

We note that for the biophysical datsets there are first additional results for combining 3D pre-training with conventional augmentations presented in Appendix A.1.3. They show that it may be a promising avenue for future work which we discuss in Section 7.1.1.

6.2.3. Predictive 3D pre-training

The pairwise distance prediction pre-training method is much simpler than our latent space pre-training and if it works similarly well it would be preferred. Therefore this experiment compares our predictive 3D pre-training with our best latent space SSL method which uses the *multi3D* version of NTXent as described in the previous Section 6.2.1 (labeled *latent SSL* here and *Drugs 3D* in Section 6.2.1). We call the first predictive approach which is presented in Section 5.2.1 as *Distance Predictor*. The second is called *Optimal Transport* and is the method described in Section 5.2.2.

All methods are pre-trained on the same 140 000 molecules of GEOM-Drugs of which *Distance Predictor* uses the highest probability conformer. Meanwhile, *latent SSL* uses 5 conformers where the lowest energy conformer is repeated if there are fewer than 5 conformers as described in Section 5.1.6. The *Optimal Transport* approach uses up to 10 of the most probable conformers. After pre-training, all methods are fine-tuned to predict the properties of QM9.

Table 6.5.: Comparison of our strongest 3D pre-training method against our predictive SSL methods. The numbers show the MAE for predicting QM9's properties (lower better) after pre-training or of a randomly initialized baseline. Colors indicate improvement or worse performance compared to the baseline. Latent space SSL is always better than the predictive approaches which do not consistently improve over the baseline.

method	μ	α	$homo$	$lumo$	gap	$r2$	$ZPVE$	c_v
Rand Init	0.4148	0.3348	82.10	87.74	120.94	22.14	15.08	0.1670
Distance Predictor	0.4626	0.3570	80.58	84.93	116.21	29.23	25.91	0.1587
Optimal Transport	0.3940	0.4219	79.75	79.16	110.72	20.86	21.10	0.1555
latent SSL	0.3512	0.2959	70.78	71.38	102.59	18.96	9.677	0.1409

Table 6.5 shows that our latent space SSL 3D pre-training method is always superior to our predictive methods. The improvements of the latent space approach cannot be replicated by simply predicting the pairwise distances of all atoms. For all properties other than $alpha$, the *Optimal Transport* method yields better performance than the *Distance Predictor*. Moreover, both predictive approaches often yield significant improvements over the random initialization baseline, they also sometimes suffer from negative transfer and produce a worse MAE.

The comparison might be considered unfair because a lot more effort was invested into tuning the latent space approaches as the following sections show. However, this was only done on the *homo* property and we deem the differences between latent and predictive pre-training large enough to be sufficient evidence that the predictive methods cannot match our contrastive pre-training.

6.2.4. Contrastive Learning Losses

Next, we compare the different losses to estimate and maximize the mutual information as they are presented in Section 5.1.1. For this purpose we pre-train PNA on 50 000 molecules from QM9 and another instance on 140 000 molecules of GEOM-Drugs, both with a single conformer. We do so with the Donsker-Varadhan estimator, the Jensen-Shannon estimator, InfoNCE, and NTXent. For NTXent we search over seven temperature parameters $\tau \in [0.05, 0.1, 0.2, 0.3, 0.4, 0.5, 0.7]$ and choose $\tau = 0.1$.

Table 6.6.: Comparison of mutual information estimators for latent space SSL. The middle double-column shows the results for pre-training on one half of QM9 and the right double-column corresponds to pre-training on one half of GEOM-Drugs and the second row indicates what dataset was used for fine-tuning. The ***Rand Init*** row shows the performance when training from scratch without any pre-training. For QM9 the reported number is the MAE of the homo and for GEOM-Drugs it is the MAE when predicting the ensemble Gibbs free energy. NTXent is our best loss function for contrastive learning.

Loss/Estimator	QM9 pre-training		GEOM-Drugs pre-training	
	QM9	GEOM-Drugs	QM9	GEOM-Drugs
Rand Init	82.10±0.33	.2035±.0011	82.10±0.33	.2035±.0011
Donsker-Varadhan	82.49	.2152	85.46	.2013
Jensen-Shannon	80.71	.2078	81.61	.2047
InfoNCE	75.81	**.1938**	79.31	.1894
NTXent	**68.96**±0.32	**.1945**	**71.66**	**.1844**

In Table 6.6 we see that 3D pre-training on GEOM-Drugs or QM9 can yield significant improvements for predicting quantum mechanical properties, especially when using InfoNCE and NTXent as objectives. These two objectives perform better than the Donsker-Varadhan and Jensen-Shannon estimator in every case and the Jensen-Shannon objective is superior to the Donsker-Varadhan estimator which seems to yield no significant improvements over random initialization. The superiority of the Jensen-Shannon loss over the Donsker-Varadhan alternative is in line with the findings of Hjelm, Fedorov, Lavoie-Marchildon, et al. 2019 in their different setting on images. While NTXent seems to perform better than InfoNCE in three settings, this might be due to the additional investment in searching through temperature parameters for NTXent.

6.2.5. Batch Size, Memory Consumption, and Training Time

For the contrastive learning losses like InfoNCE, the estimate of the mutual information becomes more accurate as the batch size increases. Empirically, large batch sizes have been shown to be important for both InfoNCE and NTXent (Hjelm, Fedorov, Lavoie-Marchildon, et al. 2019; T. Chen, Kornblith, Norouzi, and Hinton 2020). In our best-performing setups we are using a batch size of 500 leading to large memory requirements during pre-training. We briefly verify if this is necessary and if the previous findings regarding the batch size also hold in our scenario where we are dealing with different modalities. To check this, we pre-train on QM9 with NTXent as in Section 6.2.4 but with different batch sizes.

Table 6.7.: Downstream performance for different pre-training batch sizes. Shown is the MAE when fine-tuning on the homo property of one half of QM9 after pre-training on the other half of QM9 with NTXent. Large batch sizes are vital for the good performance of NTXent.

	500	400	300	200	100
QM9 MAE	68.96±0.32	69.81	70.18	71.13	73.65
Random Init	82.10±0.33				

In Table 6.7 we observe a higher MAE as the pre-training batch size decreases and that large batch sizes are necessary for best performance in our setup with NTXent. The high memory consumption entailed by this is a large drawback of our latent SSL pre-training methods. While for pre-training with QM9 and a batch size of 500, the required vRAM is a feasible 4.2 GB, the quadratic complexity of Net3D and EGNN in the number of atoms means that this does not hold when operating on larger drug-like molecules. This becomes especially prominent when pre-training with multiple conformers with a strategy such as *multi3D* where the 3D network processes multiple sets of 3D coordinates in parallel. A 12GB vRAM GPU is required when pre-training with GEOM-Drugs and a single conformer or the *conformer sampling* strategy for multiple conformers. When instead using $\mathcal{L}_{NTXent}^{multi3D}$ as objective with 5 conformers, 42GB of vRAM are needed with a batch size of 500.

When these amounts of memory are not available, it makes sense to consider using SMP as 3D Network. In that setting our 3D contrastive pre-training method is not more expensive than conventional contrastive learning setups using augmentations such as GraphCL (You, T. Chen, Sui, et al. 2020). In fact 3D pre-training might be considered cheaper since the pre-training datasets can usually be much smaller as investigated in Section 6.2.14.

Overall, 3D pre-training with $\mathcal{L}_{NTXent}^{multi3D}$ and 3 conformers on all 620 000 molecules of QMugs takes 12 hours. Pre-training with 280 000 molecules from GEOM-Drugs using 5 conformers takes 9 hours. This is in comparison with the GraphCL baseline for which pre-training took 71 hours using the same 280 000 molecules of GEOM-Drugs since it has to train many more epochs to reach convergence.

6.2.6. SSL Methods

Here we compare our the four proposed latent space SSL methods. These are contrastive learning using NTXent, Barlow Twins, multi-modal BYOL, and VICReg as described in Section 5.1. We pre-train these methods on one half of QM9. For a fair comparison, we search through 8 different hyperparameter settings based on the downstream performance on the QM9 homo property (the exact search space and the final parameters can be found in Appendix A.2.3). After these method-specific hyperparameters were selected, we tuned every method with a random search over the same search space.

For contrastive learning, we vary the temperature of NTXent τ. When using BYOL we try different decay rates γ for the exponential moving average weight copying. Here, we include $\gamma = 0$ making our setup similar to a multi-modal version of SimSiam (X. Chen and K. He 2020). For Barlow Twins, the hyperparameter is λ weighting the redundancy loss. Lastly, for VICReg we vary μ and ν, the parameters for the variance and the covariance regularization.

Table 6.8.: Comparison of latent space SSL methods. The numbers show the MAE when predicting QM9's homo property after pre-training on one half of QM9 with the given method and fine-tuning on the other half of QM9. The ***Rand Init*** column shows the MAE without pre-training and with random weight initialization. Contrastive learning is our best latent space SSL method.

	Random Init	Contrastive	BYOL	Barlow Twins	VICReg
QM9 MAE	82.10±0.33	**68.96±0.32**	79.16±0.58	82.38±0.48	85.15

The results in Table 6.8 showcase that contrastive learning clearly is the superior method in our setting. It decreases the MAE from 82.10 $\pm$ 0.33 to 68.96 $\pm$ 0.32 while the other methods either lead to no improvement or to the much smaller drop to 79.16 $\pm$ 0.58 for BYOL. This is not due to collapse to a constant solution since we can observe a high variance between the representations in a batch for all methods. Furthermore, with the final parameter settings, all methods were able to achieve a low value for their loss during pre-training, both on the training and validation data and there are no optimization issues.

Intuitively, the results can be explained by contrastive learning being the only method that optimizes a lower bound on the mutual information, which potentially makes it especially fit for our setting. The other approaches have no direct relation to mutual information and instead rely on maximizing a notion of similarity with tricks to prevent collapse. While this might work for conventional SSL, we see no success in our scenario where the rigorous guarantee on maximizing the mutual information seems valuable.

Another reason for the poor performance of BYOL and especially Barlow Twins and VICReg might be that they rely on having symmetric networks to generate the compared representations. In our scenario, we have very little similarity between the architectures with our 2D and 3D networks operating on different modalities. This hypothesis would fit in line with the findings of Bardes, Ponce, and LeCun 2021 and Zbontar, Jing, Misra, et al. 2021 where introducing asymmetries between the networks hurt performance.

6.2.7. Dimensional Collapse

One phenomenon in latent space SSL methods that often leads to poor downstream performance is dimensional collapse (Hua, W. Wang, Xue, et al. 2021) where most components of our latent representations capture the same information. If this happens, the representations do not contain rich information and if we observe dimensional collapse we can likely assert that the 2D network has not learned to generate useful 3D information from its inputs. In our contrastive learning setup, this could, for instance, happen if the model learns to distinguish positive and negative pairs solely based on their number of atoms instead of using the 3D information to do so.

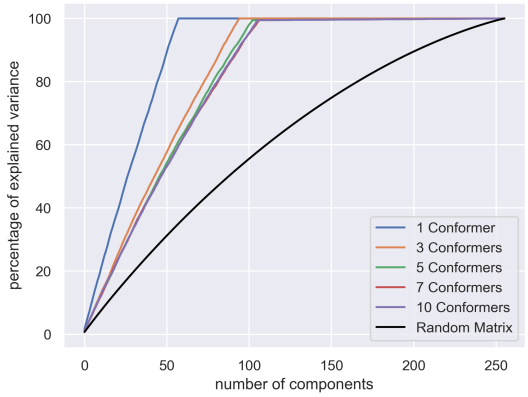
As such, we wish to investigate how many components of our 2D representations actually capture information that does not linearly depend on other dimensions. To do so, we examine the latent representations of a batch of molecules. For a batch of 500 2D representations of dimensionality 256 represented as $Z^a = [z_1^a, \ldots, z_{500}^a] \in \mathbb{R}^{500\times256}$ we calculate the singular value decomposition $Z^a = U\Sigma V^T$. Here, Σ is a 500×256 rectangular diagonal matrix with the singular values σ_k on the diagonal. The singular values squared σ_k^2 are the covariance of $(Z^a)^T Z^a$ and tell us how much of the variance between the dimensions is explained by the component belonging to that singular value.

Intuitively, the singular values show us how much non-linearly dependent information is captured by the component to which they correspond. Therefore, we can use them to find out how many of our 256 components capture information. We sort them in descending order such that σ_1's component captures the most variance and calculate their cumulative sum. This is then normalized between 0 and 100 and plotted in Figure 6.1 such that the values can be interpreted as the percentage of captured variance. This way, a y-value in Figure 6.1 shows us how much of the total variance is captured by the number of components on the x-axis.

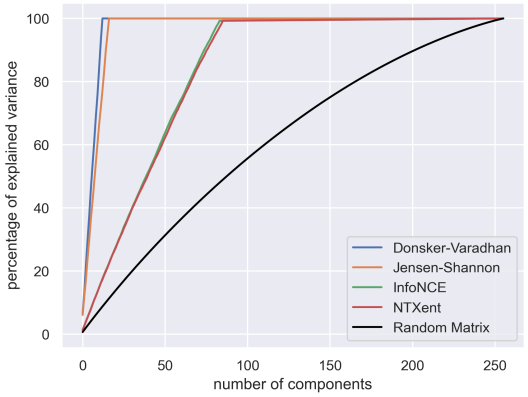
With this method we examine the latent representations of 2D networks that were pre-trained on GEOM-Drugs with different amounts of conformers in the *conformer sampling* strategy described in Section 5.1.6. Furthermore, we generate the plots for the latent representations of 2D networks trained with the different losses on QM9 evaluated in

We find in Figure 6.1a that more information being available during pre-training (a higher number of conformers) leads to more rich representations that require more dimensions to capture all of the information (a higher number of components is needed until 100% of the captured variance is reached). However, with more than 3 conformers, the number of necessary components only increases marginally. This fits the fact that 3 conformers already capture most of the Boltzmann weight for most molecules in the dataset (which can be seen in Appendix B.2 Figure B.2). The observations also fit the results for experiments on how many conformers to use in Section 6.2.12 where three seem to capture most of the information that is valuable for downstream performance.

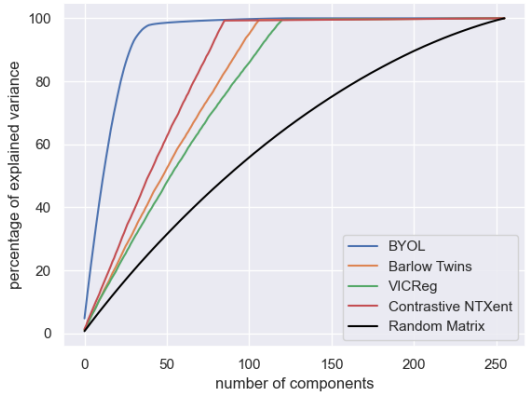
In Figure 6.1b the amount of information captured by each 2D network's latent representations correlates with downstream performance. If there are more dimensions that capture non-independent information the performance is better. However, this does not hold between Figure 6.1a and Figure 6.1b since NTXent with a single conformer pre-trained on QM9 achieved an MAE of 68.96 ± 0.32 while the 10 conformer setup had a worse MAE of 72.29.



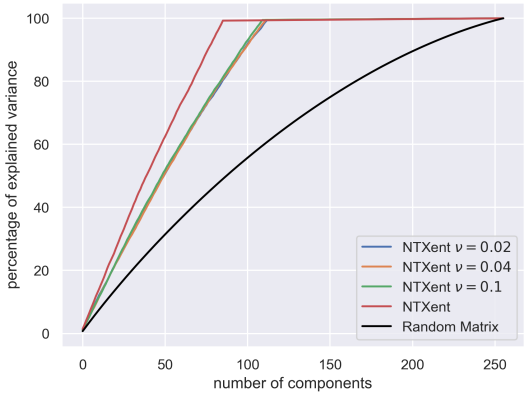
(a) Vary the number of conformers for pre-training on GEOM-Drugs with NTXent and *conformer sampling*.



(b) Vary objective using a single conformer (QM9).



(c) Different SSL methods (QM9).



(d) Single conformer with covariance regularization (QM9).

Figure 6.1.: For a given number of components of latent 2D representations as value on the x-axis, the corresponding y-value shows the maximum percentage of the total variance that can be explained by this amount of components. Intuitively, the plots tell us how many components of our latent representations are needed to capture a certain percentage of the total information that is contained in them. *Random Matrix* corresponds to standard normal distributed latent representations where all components are needed to explain the variance. In plot a) we find a correlation between the number of necessary components to capture 100% of the representation's variance and the information that is available during pre-training in the form of conformers. When comparing very similar settings such as only changing the number of conformers or the loss function, the downstream performance corresponds to the number of needed components.

An explanation for the disparity could be that the number of necessary components correlates with the information available during pre-training and not necessarily with the downstream performance: GEOM-Drugs contains larger molecules, more training samples, and more information which is why the representations capture information in more dimensions than those after pre-training with QM9.

Moreover, in Plot 6.1c the method's downstream performance does not fit the picture either with the two worst performing methods (Barlow Twins and VICReg) requiring the highest number of dimensions to represent all of the information. Presumably, this is due to the feature decorrelation terms that are present in Barlow Twins's and VICReg's loss terms. The claims that the number of necessary components correlates with downstream performance or the information available during pre-training only hold when comparing very similar settings such as when only the number of conformers is varied.

The figures also showcase that only a fraction of the dimensions in the latent representations are used. Yet, reducing the latent dimension to 100 instead of 256 decreased the downstream performance in the case of pre-training with NTXent on a single conformer on QM9. Intuitively, the high number of dimensions might be helpful for optimization since it is "easier" for the comparisons of the contrastive loss to happen in the higher dimensional space.

Considering the low number of used components there may be merit in decorrelating the dimensions with a regularization term as it is used in VICReg by Bardes, Ponce, and LeCun 2021 to prevent dimensional collapse and obtain rich representations. Indeed, when adding the covariance regularization term to the contrastive loss, Figure 6.1d shows that more dimensions of the representations are utilized. However, the downstream MAE of vanilla NTXent was still the best and no improvement was possible. Simply regularizing to increase the number of components does not necessarily translate to better performances.

From these experiments we conclude that (1) using multiple conformers leads to richer representations, (2) more components capturing information does not necessarily imply better downstream performance, and (3) we are not experiencing dimensional collapse.

Nevertheless, we additionally address the concern of our contrastive setup simply using the number of atoms to distinguishing some positive and negative pairs by trying strategies that prevent the possibility of node counting. These include sampling the batches such that they include only molecules with the same number of atoms or large clusters where the number of atoms is identical. Another approach is to drop a random number of nodes from the 2D or 3D inputs. Although these ideas mostly did not hurt performance, there was no consistent improvement to be found either and we, therefore, do not employ them.

6.2.8. 3D Networks

So far, we have only used Net3D as a 3D network. In this section, we justify this choice. In Table 6.9 we compare the different 3D networks presented in Section 5.1.7 when pre-trained and fine-tuned on the same dataset. Net3D uses the pairwise distances as input only after mapping them to a higher dimensional space using the high-frequency functions of γ as it is described in Section 5.1.7. The reasoning for this is the assumption that a lot of the most important 3D information has only small variations. In these settings the value of encoding

schemes such as γ has been theoretically and empirically shown (Rahaman, Baratin, Arpit, et al. 2019; Tancik, Srinivasan, Mildenhall, et al. 2020; Zhong, Bepler, Davis, and Berger 2020; Mildenhall, Srinivasan, Tancik, et al. 2020). However, this was for very different types of applications and the benefit of our γ function has to be shown, otherwise, it is an unnecessary complication. We call it *Net3D w/o* γ if Net3D directly operates on the pairwise distances.

Table 6.9.: Comparison of 3D networks. The MAE of the homo property pre-training and fine-tuning on different halves of QM9. ***Net3D w/o*** γ refers to dropping the high dimensional distance encoding at the beginning of Net3D. Net3D achieves the best MAE.

	QM9 MAE
Rand Init	82.10±0.33
SMP	72.37
EGNN	70.46
Net3D w/o γ	70.34
Net3D	**68.96±0.32**

In Table 6.9 we can observe that Net3D yields the best downstream performance and that using our γ function is a valuable component of it. Possibly EGNN would benefit similarly from this encoding for the pairwise distances considering its high similarity with Net3D. SMP's downstream performance is the worst which could be expected since the 3D input representation which it uses does not uniquely define all the relative positions in a molecule. Therefore 3D information is missing and the 2D network cannot learn to capture it in its representations.

We have to note that SMP is able to distinguish chiral molecules unlike the other 3D networks but this advantage cannot be evaluated with our experiments on quantum mechanical properties. Chirality only becomes relevant when considering the interactions between molecules and in these situations SMP might be able to leverage its advantage such that our evaluation could be criticized as not entirely fair. Additionally, SMP has much lower memory requirements since it does not suffer from the quadratic complexity of EGNN and Net3D in the molecule size.

6.2.9. Constraining the 3D Network

One major concern of our latent space SSL methods is that the mutual information between the 2D and the 3D representations might be maximized by the 3D network inferring 2D information from its inputs instead of the 2D network learning to generate 3D information. In all our methods both options would lead to optimizing the objective but the 2D information is already explicitly available during fine-tuning and much more value is in encoding 3D information during pre-training. Our only approach to prevent this issue is to limit the expressivity of the 3D network with the intuition that this will make it harder for the 3D network to generate 2D information and that the 2D network has to "do the work" instead.

In Table 6.10 we show the performance on QM9's homo property after pre-training with NTXent as in Section 6.2.4 when varying the propagation depth of Net3D (changing L) or the dimensionality of its hidden representations (changing d_d and d_h). When varying the hidden dimension we use a propagation depth of 1 and the same size for the edge and the node representations $d_d = d_h$.

Table 6.10.: Performance when varying the size of the 3D network. Both tables show the MAE when predicting QM9's homo property after pre-training on a disjoint set of QM9. The top table shows it for different depths of Net3D while the bottom table shows it for different hidden dimensions. The 3D network can be very small and still perform well such that its quadratic memory complexity is less relevant.

Depth	1	2	3	4	5
QM9 MAE	**68.96**±0.32	69.34	**68.77**	69.70	70.65

hidden dimension	10	20	40	60	80	100
QM9 MAE	73.15	**68.96**±0.32	**68.66**	69.99	69.70	69.29

The results show our hypothesis, that decreasing the expressivity of Net3D would increase the downstream performance, to be incorrect. Possibly the problem we wish to prevent (2D information is encoded to maximize the mutual information instead of 3D information) does not actually occur and is therefore not solved by reducing the 3D network's size. More likely is that this crude approach just is not the right solution to address the problem. Increasing the difficulty for the 3D network to learn to generate 2D information by introducing high dropout was unsuccessful as well.

Nevertheless, we find that a very small 3D network with only one message passing step and a $d_d = d_h = 20$ is still able to perform well. This is useful for multiple conformers and larger molecules for which the method becomes expensive in memory due to the quadratic complexity of Net3D in a molecule's number of atoms. In light of this we also tried even more simplistic approaches such as variations of Deep Set (Zaheer, Kottur, Ravanbakhsh, et al. 2017) models operating on the set of pairwise distances $\{d_{uv} \mid u, v \in \mathcal{V} \wedge u \neq v\}$ or after encoding them with γ, but none were able to beat Net3D. This and additional approaches for constraining the 3D network are described in Appendix A.1.

Reducing the expressivity of the 3D network or using high dropout is a very crude approach to address the issue of maximizing the mutual information by encoding 2D information. Since this problem might be the main bottleneck of our method we think that finding more principled and effective approaches to solve it is the most promising direction to improve our 3D pre-training.

6.2.10. Local-Global Infomax

Next, we will evaluate the local-global objectives as they were introduced in Section 5.1.5 for the contrastive learning setup. The motivation is guaranteeing that globally relevant

information is captured in the embeddings instead of the method potentially using local noise to distinguish positive and negative pairs. While this reasoning justifies using a global 2D representation and local 3D representations, we test the mirrored option as well. We do so with NTXent adapted to the local-global objective $\mathcal{L}_{NTXent}^{local\text{-}global}$ as described in Equation 5.23 and with the same changes to InfoNCE. The measured quantity is the downstream performance when fine-tuning on the homo property of one half of QM9's molecules after pre-training with GEOM-Drugs or with the other half of QM9.

Table 6.11.: Evaluation of local-global 3D pre-training schemes. Shown is the MAE when predicting QM9's homo property after pre-training either with a disjoint half of QM9 or with GEOM-Drugs. This is for either local 3D representations and a global 2D representation or the other way around. The ***Rand Init*** row shows the MAE without pre-training and with random weight initialization. Local-Global methods perform poorly.

	Global 2D, Local 3D		Local 2D, Global 3D	
	QM9	GEOM-Drugs	QM9	GEOM-Drugs
NTXent	79.78	81.87	85.75	82.47
InfoNCE	82.16	81.48	-	-
Random Init	$82.10_{\pm 0.33}$			

Table 6.11 shows our local-global setup as performing poorly compared to the usual global-global objective. The highest decrease in MAE that we were able to achieve was by 3 percentage points compared to the 16 percentage points demonstrated in Section 6.2.4. For InfoNCE adapted to local 2D and global 3D representations the contrastive loss did not decrease during pre-training which held true across multiple additional hyperparameter settings.

Potentially, the local-global objective is too hard with different input modalities and is more fit for domains such as images where its value was demonstrated by Hjelm, Fedorov, Lavoie-Marchildon, et al. 2019 or for large graphs with the goal of learning node level representations where it was applied by Velickovic, Fedus, Hamilton, et al. 2019.

6.2.11. Multiple Conformers

In this section, we evaluate if there is a benefit from using multiple conformers during pre-training and which of the different approaches presented in Section 5.1.6 best leverage this additional information. There is reasonable doubt that this should be the case considering the results of Axelrod and Gómez-Bombarelli 2020 who explicitly used the 3D information of multiple conformers of GEOM-Drugs and did not find significant benefits over using a single conformer. Another hypothesis we wish to test is that for smaller molecules such as those in QM9, the ability to make predictions informed by multiple conformers is not as important as for larger drug-like molecules. The reasoning is that a single conformer takes most of the Boltzmann weight for QM9's molecules due to the lower degree of freedom.

We test *conformer sampling*, *multi3D*, and *multi3D+2D* when pre-training on either QMugs or one half of GEOM-Drugs and fine-tuning on QM9 or the other half of GEOM-Drugs. In QMugs we have three diverse conformers available for each molecule which are all used, while for GEOM-Drugs different numbers of conformers are available of which we use the five with the highest Boltzmann weight i.e. lowest energy. If there are fewer than five we duplicate the lowest energy conformer (see Section 5.1.6 for details). We recall that for the *multi3D+2D* loss sets with as many elements as conformers are produced by the 2D and 3D networks. Both the discussed sim_{all} and sim_{max} are used as similarity measures between those sets. For the *conformer sampling* strategies of using a uniform weighting or sampling conformers according to their Boltzmann weight, we do not evaluate the latter on QMugs since we do not have it available with exactly three conformers per molecule. When fine-tuning on QM9, pre-training is stopped after 35 epochs with QMugs and after 50 epochs with GEOM-Drugs since that yields the best performance as will be further investigated in the next Section 6.2.12.

Table 6.12.: Comparison of strategies for using multiple conformers. The middle double-column shows the results for pre-training on one half of GEOM-Drugs and the right double-column corresponds to pre-training on QMugs and the second row indicates what dataset was used for fine-tuning. The ***Random Init*** row shows the performance when training from scratch without any pre-training. For QM9 the reported number is the MAE of the homo property and for GEOM-Drugs it is the MAE when predicting the ensemble Gibbs free energy. There are large improvements from using multiple conformers but the differences between the methods are small.

Loss/Estimator	GEOM-Drugs pre-training		QMugs pre-training	
	QM9	GEOM-Drugs	QM9	GEOM-Drugs
Rand Init	82.10±0.33	.2035±.0011	82.10±0.33	.2035±.0011
single conformer	71.66	.1844	82.57	.1966
uniform sampling	**70.66**	**.1823**	72.94	.1874
boltzmann sampling	**70.93**	.1846	x	x
multi3D	**70.78**	**.1811**	**70.77**	**.1831**
sim_{all}	71.11	.1849	72.40	.1936
sim_{max}	**70.81**	.1896	**71.15**	**.1840**

In Table 6.12 we can observe that there are large improvements possible when using multiple conformers. After pre-training on QMugs, the MAE when predicting the homo property decreases from 82.57 to 70.77 and from .1966 to .1831 for predicting the Gibbs free energy for GEOM-Drugs. Notably, these improvements are much larger than when pre-training with GEOM-Drugs. This is likely because the GEOM-Drugs dataset contains the lowest energy conformers and we always use the most probable one with the highest Boltzmann weight when pre-training with a single conformer. Meanwhile, the QMugs dataset contains three diverse conformers per molecule and not the ones with the highest Boltzmann weight. Pre-training with the lowest energy conformer from GEOM-Drugs already captures

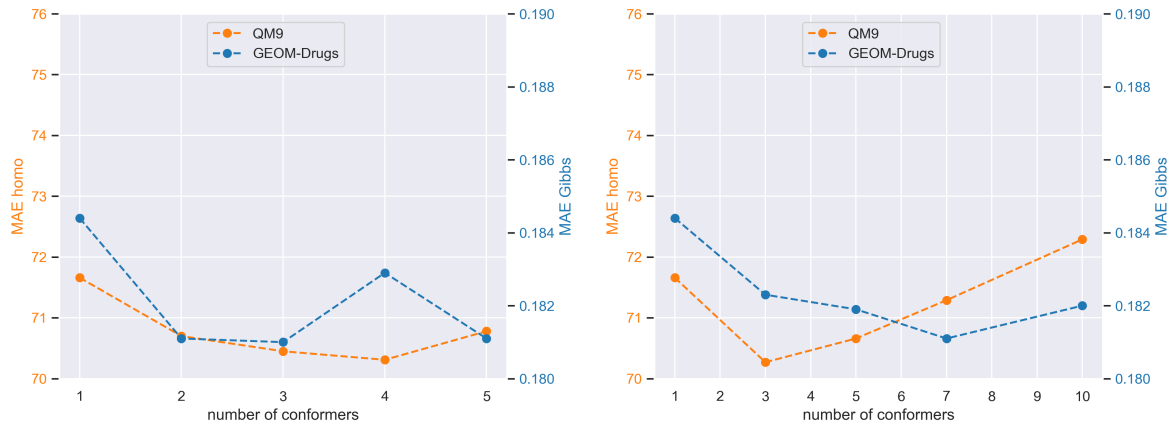
most of the relevant information and using more is not as beneficial. However, for QMugs, using the information of all three diverse conformers is crucial. The effect on the downstream performance of varying the number of conformers is investigated more closely in the next Section 6.2.12

Similar to the small improvements over the random initialization baseline with GEOM-Drugs, the different methods for using multiple conformers mostly perform the same when pre-training with GEOM-Drugs. When pre-training with QMugs instead, the MAEs are overall slightly worse and we find *multi3D* to perform the best. Note that this is with the slight caveat that the epoch at which pre-training is stopped for all methods was chosen based on where *multi3D* had the lowest MAE.

Due to these results we consider *multi3D* and *conformer sampling* with uniform weighting as our best methods since *multi3D* performs slightly better with pre-training on QMugs but *conformer sampling* is simpler and especially uses much less memory. For *multi3D*, all the conformers need to be processed in parallel and training with more than 5 conformers and a batch size of 500 would not be possible on a 48GB vRAM GPU.

The hypothesis that the downstream performance on the smaller molecules of QM9 would benefit less from using multiple conformers than the molecules of GEOM-Drugs clearly does not hold. Surprisingly, the improvements on the small molecules of QM9 are larger.

6.2.12. Number of Conformers



(a) pre-train with *multi3D*

(b) pre-train with *conformer sampling*

Figure 6.2.: Performance when varying the number of conformers when pre-training with GEOM-Drugs. The left vertical axis shows the MAE of a model when predicting QM9's homo property and the right one shows the Gibbs free energy of GEOM-Drugs when fine-tuning on those datasets. Note the start and endpoints of the vertical axes. Using more than a single conformers is helpful but benefits from even higher amounts are not evident. Three conformers seem sufficient.

Having established that using multiple conformers is helpful in Section 6.2.11, this section assesses the impact of the number of pre-training conformers on the downstream performance. We evaluate this for the *multi3D* and the *conformer sampling* strategy by pre-training with one half of GEOM-Drugs and fine-tuning on the other half of GEOM-Drugs or on QM9. Pre-training is stopped after 50 epochs.

Figure 6.2 shows that there is a benefit to be had from using more than a single conformer. However, using more information by going beyond three conformers does not help and most of the information that is useful for downstream performance is already captured with the three lowest energy conformers.

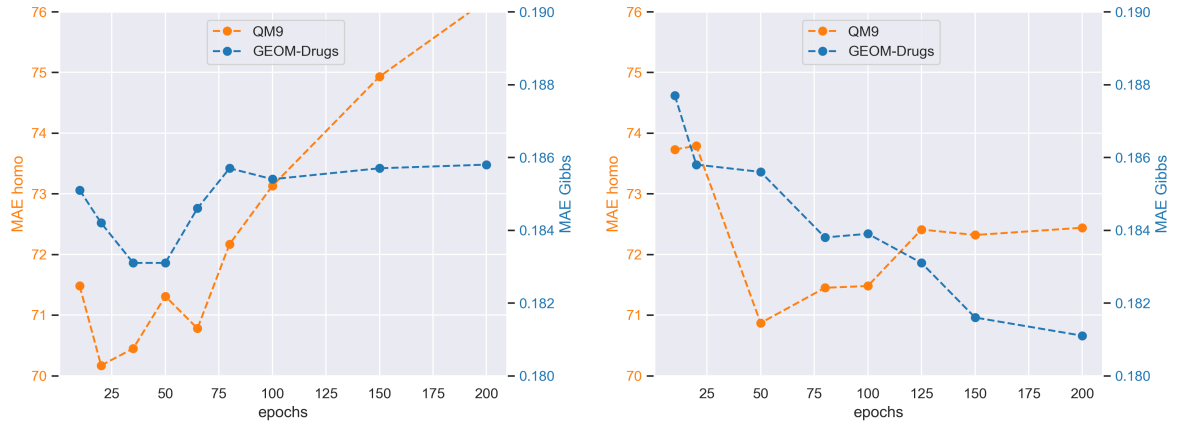
In Figure 6.2b it even becomes evident that including more higher energy conformers can hurt performance. With three conformers the MAE for the homo property is 70.27 and it significantly increases to 72.29 when using 10 conformers (for the random initialization baseline the standard deviation is 0.33 as seen in Table 6.6). While the learned representations using more conformers capture more information (see Figure 6.1a), it is not necessarily useful and comes at the cost of the lower energy conformers' information.

We note that, averaged across all molecules in GEOM-Drugs, 3 conformers are enough to cover 70% of the cumulative Boltzmann weight (see Figure B.2 for the detailed distribution). This corresponds to the observation that most of the relevant information for downstream predictions is already captured with 3 conformers.

6.2.13. Pre-training Time and Generalization

We evaluate how the amount of epochs for which we pre-train affects the downstream performance since a lower contrastive loss after pre-training does not necessarily mean better downstream performance. From the same pre-training run, we select the model with the lowest NTXent validation loss after a certain number of epochs and evaluate its performance when fine-tuned. The number of epochs is then varied. We do so for two different pre-training runs: (1) pre-training on 140 000 molecules from GEOM-Drugs with 5 conformers using the *multi3D* objective $\mathcal{L}_{NTXent}^{multi3D}$ and (2) pre-training on 620 000 molecules from QMugs with 3 conformers using the *multi3D* loss. The downstream tasks are the homo property of QM9 and the Gibbs free energy for GEOM-Drugs.

In Figure 6.3b we pre-train on half of GEOM-Drugs and then fine-tune either on the other half of GEOM-Drugs or on QM9 with its highly different molecules. We can see that pre-training for more than 50 epochs decreases the downstream performance for QM9's homo property (the contrastive loss is still decreasing substantially in this phase of pre-training). Meanwhile, the performance on GEOM-Drugs improves with training time before plateauing. This could be explained by the longer pre-training hurting generalization. After a certain point, the 2D model no longer learns to encode 3D information that is generally informative and therefore also helpful for the molecules of QM9. Instead, it starts maximizing the mutual information by encoding information that is only valuable for tasks on similar molecules such as the other half of GEOM-Drugs. This is to say that we are experiencing typical overfitting by training too long with the only difference that this is during pre-training. In the usual train-validation-test split scenario, early stopping can help to improve performance on the



(a) pre-train with QMugs

(b) pre-train with GEOM-Drugs

Figure 6.3.: Performance when stopping pre-training after a certain amount of epochs when pre-training with QMugs (a) or when pre-training with GEOM-Drugs (b). The left vertical axis shows the MAE of a model when predicting QM9's homo property and the right one shows the Gibbs free energy of GEOM-Drugs when fine-tuning on those datasets. Note the start and endpoints of the vertical axes and that the changes in MAE Gibbs are only very slight. Pre-training too long hurts the downstream performance on QM9.

test data and in our setting, stopping pre-training earlier improves downstream performance.

This behavior is even more prominent in Figure 6.3b where longer pre-training hurts the downstream performance when generalizing from QMugs to QM9 even more. Note that one pre-training epoch here processes around 4 times as many molecules as in the right figure. When pre-training with QMugs and fine-tuning on GEOM-Drugs, we still only see very slight changes in downstream performance over pre-training time, presumably since both of the datasets consist of more similarly sized drug-like molecules.

6.2.14. Pre-training Dataset Size

We wish to determine how relevant the size of the pre-training dataset is for downstream performance and whether the trend suggests that pre-training on even more molecules could lead to bigger improvements. For this purpose, we pre-train PNA on different amounts of molecules from GEOM-Drugs with 5 conformers that are used with the strategy *conformer sampling* described in Section 5.1.6. These models are then fine-tuned to predict the homo property of QM9 as a performance metric.

Figure 6.4 shows that performance improves as the size of the pre-training dataset is increased. The difference between pre-training on 140 000 molecules which yields an MAE of 70.66, and pre-training on 280 000 molecules with 70.09 is relevant but not large. As such, we cannot claim that larger pre-training datasets are likely to drastically improve the performance

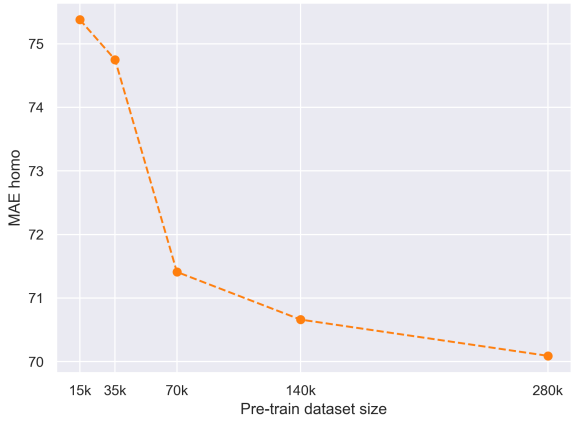


Figure 6.4.: Comparison of performance when varying the size of the pre-training dataset. The vertical axis shows the MAE of a model when predicting QM9's homo property after being pre-trained on the number of molecules specified on the horizontal axis. More pre-training data helps but the benefits saturate.

even more. This is especially considering the results from Section 6.2.13 where pre-training on 620 000 molecules from QMugs leads to an MAE of 70.17.

7. Conclusion

The goal of this thesis was to develop a method to pre-train a GNN with the 3D information of molecules and to evaluate whether this enables better downstream property predictions for molecules where no 3D information is directly available.

On the one side, we proposed predictive pre-training with different pretext tasks. On the other side we devised a framework that learns joint embeddings of a molecular graph and the corresponding 3D information. We adapted four SSL methods to this framework in which we explored local similarity objectives, the best way to encode the 3D information, different mutual information estimators, and developed a method to leverage multiple molecular conformations. We discussed what aspects of a molecule's geometry each method could capture, their theoretical limitations, and what their main practical drawbacks are.

We evaluated our method experimentally on 10 quantum mechanical properties and 10 datasets with biophysical properties. Additionally, we analyzed the latent representations of the models with respect to the information they capture and when the approach fails.

The strongest method was a contrastive learning approach that leverages multiple conformers using a novel 3D GNN. It enables a GNN to generate latent 3D information from 2D molecular graphs which can then be used during fine-tuning to improve molecular property predictions. We found that there consistently are either improvements (always and especially **large gains for quantum mechanical properties**) or the performance is the same compared to the baseline that was not pre-trained. While other pre-training methods sometimes suffer from negative transfer, 3D pre-training did never decreases performance in any test and it is therefore very **reliable** and a valuable method for practice.

Moreover, the learned representations **generalize well**. The pre-training and fine-tuning datasets can be highly different and we still observe large improvements. Lastly, the experiments provided the insight that using low energy conformers is more important for downstream accuracy than a high diversity in conformers.

7.1. Future Work

In this section, we point out future work and experiments that we believe to be interesting and relevant either for improving our method or for moving the field forward.

7.1.1. Combining pre-training Methods

The improvements after 3D pre-training are due to 3D information that is captured in the latent representations and can be used during fine-tuning. These increases in performance should be disjoint with those that are possible after conventional pre-training with, e.g., GraphCL

(You, T. Chen, Sui, et al. 2020). Therefore it would be interesting if 3D pre-training can be combined with other approaches to obtain the benefits of both. Simply using GraphCL's node drop augmentation for the 2D graph and the 3D information (removing all pairwise distances for a removed atom) already showed some promising first results in our additional experiments in Appendix A.1.3 and there may be even more fruitful ways to add other SSL methods to 3D pre-training.

7.1.2. Restricting the 3D Network

We think it likely that the main bottleneck of our method is that the 3D network learns to generate 2D information instead of the 2D network understanding 3D geometry. This can happen because in the current framework the mutual information does not have to be maximized by capturing 3D information. Intuitively, it might be possible to solve the contrastive loss objective completely by distinguishing samples based on 2D information that is directly available in the molecular graph. If it is easier to infer the molecular graph from 3D information than the other way around, this might even be likely.

Our method evidently does not experience this problem to its full extent since it is able to capture some 3D information. However, it might still be the main bottleneck with only a part of the mutual information being maximized by 3D information and the rest with 2D information. We think it would be promising to find a way to remove the 2D information from this equation as much as possible but found no success with crude approaches such as limiting the expressivity of the 3D network or adding dropout to it. More rigorous and well-founded ideas might be able to overcome this limitation and improve our method.

7.1.3. 3D Information for Biophysical properties

It has been extensively studied how 3D information is beneficial for predicting quantum mechanical properties and many methods have been proposed to do so (Schütt, Kindermans, Sauceda, et al. 2017; Klicpera, Groß, and Günnemann 2020; Klicpera, Giri, Margraf, and Günnemann 2020; Y. Liu, L. Wang, M. Liu, et al. 2021; Satorras, Hoogeboom, and Welling 2021; Klicpera, Becker, and Günnemann 2021). Meanwhile, there is little evidence on whether biophysical property predictions also benefit from explicitly using 3D information with learned methods. If there are advantages, it would also be valuable to have empirical evidence about the types of properties for which this holds.

For many biophysical tasks, it is clear that knowledge about a molecule's geometry should theoretically be helpful. For instance, whether a small molecule binds to a protein's binding pocket highly depends on the shape which the molecule takes and whether that fits into the pocket. For this specific task, there have also been methods that use 3D information in highly specialized ways (Gainza, Sverrisson, Monti, et al. 2020). However, it would be very valuable if general 3D GNNs, such as those proposed for quantum mechanical properties, can improve molecular property predictions. The work that we are aware of which comes closest to this is by Axelrod and Gómez-Bombarelli 2020 but they only consider the very few COVID 19 annotations that are available in the GEOM-Drugs dataset. The biggest bottleneck

for this is probably that there are few well-established datasets of molecules with high-quality 3D information that additionally have biophysical annotations. Making progress towards such a dataset would be important. We note that Axelrod and Gomez-Bombarelli 2020 are in the process of including the MoleculeNet (Z. Wu, Ramsundar, Feinberg, et al. 2017) datasets used in the popular OGB benchmark into their GEOM-Drugs dataset with their accurate low energy conformers. As such this bottleneck might be alleviated soon and promising interesting experiments could follow.

7.1.4. Learned Conformers for Property Prediction

In theory, all information about the possible conformers is contained in the molecular graph. The issue is that using classical methods to infer it is too slow for many applications or completely infeasible for larger molecules. Chapter 1 already alluded to the possibility of generating explicit 3D information with learned conformation generation methods. While some of these methods are still very slow (M. Xu, S. Luo, Bengio, et al. 2021; Shi, S. Luo, M. Xu, and J. Tang 2021), there has recently been tremendous progress in this field. As such, it would be interesting to find out if the generated conformers are already accurate enough to lead to better molecular property predictions when used as input to one of the strong 3D GNNs for molecules (Schütt, Kindermans, Sauceda, et al. 2017; Klicpera, Groß, and Günnemann 2020; Klicpera, Giri, Margraf, and Günnemann 2020; Y. Liu, L. Wang, M. Liu, et al. 2021; Satorras, Hoogeboom, and Welling 2021; Klicpera, Becker, and Günnemann 2021).

A. Experiments

This appendix describes additional experiments which were conducted and provides information about the exact architectures that were used, hyperparameter search spaces, and other experimental details to understand the results in more depth. All code to reproduce the experiments or to pre-train an arbitrary GNN for molecular tasks is available at `https://github.com/HannesStark/self-supervised-graphs`.

A.1. Additional Experiments

In this section, we list additional experiments that we conducted but do not describe in the main body of work. These experiments inform some of our architecture and hyperparameter choices but do not provide major new insights into the main questions of this work of whether or not 3D information is valuable in pre-training and how this pre-training can best be done.

A.1.1. Global Architecture as 2D Network

An MPNN as 2D model might not be able to capture much 3D information and be poorly fit for learning the distance relations between atoms that are not close in the molecular graph since the message passing is restricted by the number of propagation steps we perform. For capturing the 3D relations it might be important to have information from distant nodes in the graph. For this reason we try replacing the GNN that we use as 2D network with three global approaches: (1) augmenting PNA with global attention where a Transformer operates on the node embeddings in parallel to the MPNN and after each layer the representations are shared by concatenating them and feeding them through a linear layer. (2) The recently proposed Spectral Attention Networks (Kreuzer, Beaini, Hamilton, et al. 2021) that make heavy use of the graph Laplacian spectrum as positional encoding for graph transformers (Dwivedi and Bresson 2020). (3) A global architecture similar to EGNN (Satorras, Hoogeboom, and Welling 2021) that operates on the complete graph and learns soft edges. The results were that, unlike expected, the global methods benefit less from our pre-training scheme. The *PNATransformer* performed admirably on QM9's properties and it would be interesting to further explore this type of architecture for different graph benchmarks.

A.1.2. ELECTRA Pre-training Baseline:

As additional conventional predictive pre-training method we implemented a version of ELECTRA (Clark, Luong, Le, and Manning 2020) for molecules[1]. This is a masked language

[1] `https://github.com/HannesStark/molecule-ELECTRA`

modeling (Devlin, Chang, Lee, and Toutanova 2019) approach where atoms are randomly masked out and during pre-training a network which is called generator has to learn to reproduce them. The special thing about ELECTRA is that it uses an additional discriminator which takes the output of the generator and tries to predict whether an atom was masked out or not. While usually the generator in masked language models is used for downstream tasks, ELECTRA uses this discriminator for fine-tuning. With this approach, we saw marginal improvements for some biophysical tasks, but never higher ones than with GraphCL (You, T. Chen, Sui, et al. 2020).

A.1.3. Combining Pre-Training Methods

Here we simply use GraphCL's node drop augmentation for the 2D graph and the 3D information (removing all pairwise distances for a removed atom) with drop ratio 0.2 during our 3D pre-training process.

Table A.1.: Comparison of performance when combining 3D pre-training with conventional pre-training by randomly dropping nodes on the 2D or 3D side (labeled ***3D SSL +***) for various biophysical property OGB datasets. Our strongest 3D pre-training method is trained on GEOM-Drugs and labeled ***3D SSL***. Random initialization ***Rand Init*** and ***GraphCL*** are the baselines. Shown is either the RMSE indicated by ↓ where lower values are better or the ROC-AUC indicated by ↑ where higher values are better. Colors indicate improvement, worse performance, or no change compared to the baseline. 3D pre-training is either on par with random initialization or better. There is no negative transfer as there is with GraphCL.

dataset	Rand Init	GraphCL	3D SSL	3D SSL +
esol↓	0.947±0.038	0.959±0.047	0.894±0.028	0.918±0.037
lipo↓	0.739±0.009	0.714±0.011	0.695±0.012	0.710±0.007
freesolv↓	2.233±0.261	3.744±0.292	2.337±0.227	2.791±0.323
bace↑	78.13±1.30	77.18±4.01	79.42±1.94	79.28±3.61
bbbp↑	68.27±1.98	71.06±2.00	69.10±1.07	68.64±2.19
tox21↑	73.88±0.64	78.92±0.61	74.46±0.74	73.73±0.69
toxcast↑	63.62±0.48	64.95±0.31	64.41±0.88	63.95±0.38
clintox↑	58.98±5.40	51.07±5.52	59.43±3.21	83.59±3.64
sider↑	55.95±3.27	57.32±5.00	53.37±3.34	58.43±1.28
hiv↑	77.06±3.16	76.06±1.06	76.08±1.33	75.38±0.95

A.1.4. Similarity Measures in Contrastive Learning

For the NTXent Loss we tried using different similarity measures instead of the cosine similarity. These include distances that we transform into a similarity measure. If d is a distance between a 2D and a 3D representation z^a and z^b we convert it to a similarity with either $sim = 1/(d+1)$ or with $sim = -d$. Here, d is either the mean squared error, the L1

distance, or the KL-Divergence. Additionally, we tried the JS-Divergence subtracted from 1 to turn it into a similarity measure. This was evaluated on one half of QM9 after pre-training on the other half of QM9 and after pre-training on one half of the GEOM-Drugs dataset. For both pre-training schemes, the cosine similarity worked the best.

A.1.5. Optimal Weight Transfer

For fine-tuning we evaluate different warmup methods for the learning rates of different sets of parameters. 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 or cosine interpolation. In this setting, we try different durations for each set of parameters. For instance, we first warm up the batch norm parameters for 700 steps, then do no warmup for the newly initialized parameters, and then warm up all parameters for 350 optimization steps. Like this, we determine the warmup schedule described inA.2.2.

A.1.6. Smaller 3D Networks

To prevent our 3D network from generating 2D information instead of the 2D network learning 3D information, we try different approaches to constrain it. This includes using very small models such as Deep Sets (Zaheer, Kottur, Ravanbakhsh, et al. 2017) operating on the set of pairwise distances $\{d_{uv} \mid u, v \in \mathcal{V} \wedge u \neq v\}$ or after encoding them with γ. Another attempt is to introduce noise in the 3D representations before using them in the contrastive loss or adding redundancy regularization to the 3D representations to, intuitively speaking, increase the difficulty of what the 3D network has to learn. None of the approaches were more successful than Net3D.

A.1.7. Additional Set Similarity Measures

In the *multi3D+2D* setting we additionally evaluate the following alternative similarity measures between the set of 2D representations and the set of 3D representations: (1) The inverse of one plus the maximum mean discrepancy. (2) Consider both sets as normally distributed and calculate their means and standard deviations. The similarity measure is then the inverse of one plus the KL-Divergence between the two normal distributions defined by the means and standard deviations. (3) We use the standard deviations and means to calculate the Jensen-Shannon divergence between their normal distributions and the inverse of one minus the JS-Divergence is the similarity measure.

A.1.8. Small Experiments

Hard Negatives. An issue in contrastive learning is often that the negative samples are too easy to distinguish and we need to mine for harder negatives. We try to generate harder negatives by adding different amount of standard normal noise to 3D Cartesian coordinates

before calculating Euclidean distances or by directly adding the noise to the distances. This approach did not help and downstream performance decreased.

Distance Predictor. Instead of the mean squared error in Section 5.2.1 we also tried using an ℓ_1-loss. Furthermore, we also tried predicting the distances as the ℓ_2-distances between the node representations. This was either done with the node representations directly or after projecting them to a lower-dimensional space such as $\mathbb{R}^3$ where they could be interpreted as 3D coordinates. For all approaches we tried transferring different amounts of GNN layers.

PNA. Before any evaluations of pre-training with 3D information we first searched through various hyperparameters for PNA (Corso, Cavalleri, Beaini, et al. 2020) to arrive at the setting that worked the best on the homo property of QM9. The same was later done for the OGB datasets.

A.2. Experimental Details

This section provides experimental details to understand the used architectures in detail and to reproduce the experimental results. All the specified standard deviations and estimated means calculated from different weight initializations use the seeds $1, \ldots, 6$.

A.2.1. Units

For the GEOM-Drugs dataset, all reported numbers have the unit kcal/mol.

Table A.2.: 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	$\frac{cal}{molK}$

A.2.2. Parameter Details

The parameter search space and final parameters for the PNA architecture is specified in Table A.3 and those of Net3D in Table A.4.

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[2] 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:

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

(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.

Table A.3.: 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]

A.2.3. SSL methods comparison

Here we detail the hyperparameter settings that were searched through when comparing the different SSL methods in the experiment in Section 6.2.6.

1. Contrastive learning with NTXent: $\tau \in [0.02, 0.05, 0.1, 0.2, 0.3, 0.4, 0.5, 0.7]$ where $\tau = 0.01$ performed the best.
2. Multi-modal BYOL: $\gamma \in [0, 0.0005, 0.001, 0.005, 0.01, 0.03, 0.05, 0.07]$ where $\gamma = 0.005$ performed the best.
3. Barlow-Twins: $\lambda \in [0.002, 0.0039, 0.005, 0.007, 0.01, 0.012, 0.015, 0.02]$ where $\gamma = 0.0039$ performed the best.

Table A.4.: 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**]

4. VICReg: $\lambda = 1$; $\mu \in [1, 0.5]$; $\nu \in [0.02, 0.04, 0.1, 0.3]$ where $\lambda = 1, \mu = 1, \nu = 0.04$ performed the best

B. Clarifications and additional Explanation

B.1. E(3) Invariance and Uniqueness of L2-distances

Molecular conformations are $SE(3)$ symmetric and therefore it is interesting to use representations of their geometry that capture these symmetries. One possibility is to use pairwise ℓ_2-distances between the atoms that are $E(3)$ invariant properties, meaning that they are also invariant to reflections. This means that if we apply any translation, rotation, and reflection to a conformer, the ℓ_2-distances do not change. The following proof that this holds for ℓ_2-distances is similar to the discussions by Satorras, Hoogeboom, and Welling 2021. The shown $E(n)$ invariance implies $E(3)$ invariance of ℓ_2-distances

E(n)-invariance of ℓ_2-distances: Let $\{r_i\}$ be a set of N points where $r_i \in \mathbb{R}^n$ and the pairwise ℓ_2-distances are $\ell_2(r_i, r_j)$. All $\ell_2(r_i, r_j)$ are invariant to transformations in $E(n)$.

Proof: Consider all $E(n)$ transformations $\mathbb{R}^n \mapsto \mathbb{R}^n : r \mapsto Or + t$ where $O \in \mathbb{R}^{n \times n}$ is orthogonal and $t \in \mathbb{R}^n$ is a translation. For all i, j we have:

$$\begin{aligned}
\ell_2(Or_i + t, Or_j + t) &= \sqrt{(Or_i + t - [Or_j + t])^T (Or_i + t - [Or_j + t])} \\
&= \sqrt{(Or_i - Or_j)^T (Or_i - Or_j)} \\
&= \sqrt{(r_i - r_j)^T O^T O (r_i - r_j)} \\
&= \sqrt{(r_i - r_j)^T (r_i - r_j)} \\
&= \ell_2(r_i, r_j)
\end{aligned} \tag{B.1}$$

Additionally, the pairwise ℓ_2-distances uniquely define all the relative positions of the atoms up to $E(3)$ symmetries. This means that for all sets of atom locations that only differ by transformations in $E(3)$, there is a unique set of ℓ_2-distances belonging to all of them.

B.2. Explanatory Figures

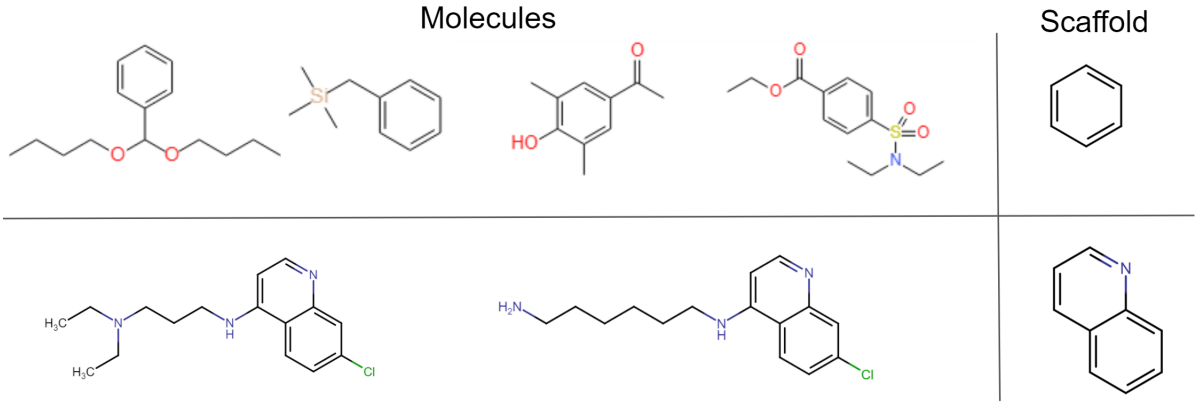


Figure B.1.: Depiction of two groups of molecules where all the molecules in the top row have the same Bemis-Murcko scaffold (Bemis and Murcko 1996) which is different from the scaffold to which the two molecules in the bottom row belong. We can easily obtain the scaffold of a molecule using RDKit (Landrum 2016).

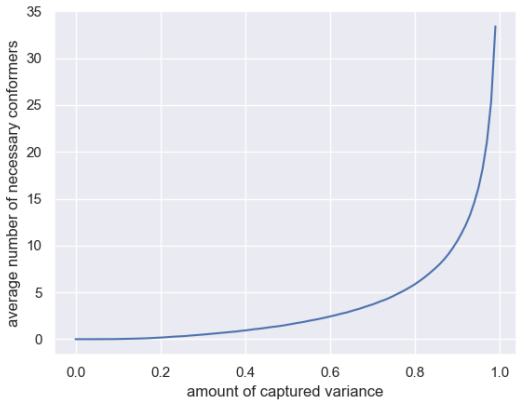


Figure B.2.: Average number of conformers necessary to cover a certain amount of Boltzmann weight in GEOM-Drugs. For a given amount of cumulative Boltzmannweight on the horizontal axis, the vertical axis shows the average number of conformers necessary to pass that threshold.

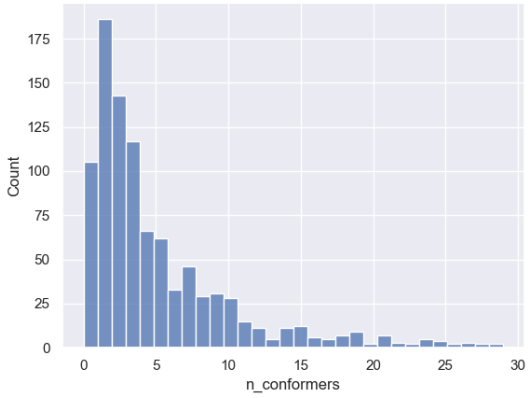


Figure B.3.: Histogram of how many molecules there are in GEOM-Drugs with a certain amount of conformers. The histogram is created for 1000 molecules of GEOM-Drugs.

List of Figures

List of Tables

Acronyms

BYOL Bootstrap your own latent. 3, 15, 17, 21, 22, 26–28, 51, 70

EGNN Equivariant Graph Neural Networks. 12, 35, 36, 43, 50, 55, 66

GNN Graph Neural Network. iv, 1–4, 6, 9–12, 20, 22, 29, 32, 34, 36, 37, 39, 40, 42, 63–66, 69

GraphCL Graph Contrastive Learning. 16, 43–47, 50, 63, 64, 67

homo energy of the highest occupied molecular orbital. 42, 44, 45, 49–51, 55, 57–61, 69–71

InfoNCE noise contrastive estimation of mutual information. 14, 25, 49, 50, 57

MAE Mean Absolute Error. 43, 45, 46, 48–52, 54–62

ML Machine Learning. 3, 4, 9–12

MLP multi-layer perceptron. 7, 34–37, 42, 70, 71

MPNN Message Passing Neural Network. 4, 6, 10–12, 42, 66

NTXent normalized teperature-scaled cross entropy loss. 25, 26, 30, 31, 39, 44, 47–54, 56, 57, 60, 67, 70

OGB Open Graph Benchmark. 42, 47, 65, 67, 69, 76

PNA Principal Neighborhood Aggregation. 42–44, 47, 49, 61, 66, 69, 70, 76

QSAR Quantitative Structure-Activity Relationship. 9, 10

RMSE Root Mean Squared Error. 43, 47, 67

ROC-AUC area under the curve of the Receiver Operator Characteristic. 43, 47, 67

SMP Spherical Message Passing. 12, 32–34, 43–45, 50, 55, 75

SOTA state-of-the-art. 10–13, 15, 37, 42

SSL self-supervised learning. iv, v, 1–3, 9, 14–17, 19–22, 36, 39, 43–45, 47–52, 55, 63, 64, 67, 70, 75, 76

VICReg Variance-Invariance-Covariance Regularization. 15, 17, 21, 22, 28, 51, 54

Bibliography

[1] 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. URL: https://doi.org/10.1021/acs.jpclett.9b03664.

[2] 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. URL: https://doi.org/10.1038/s41586-021-03819-2.

[3] 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. URL: https://doi.org/10.1038/s41524-019-0221-0.

[4] 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. URL: https://www.sciencedirect.com/science/article/pii/S0092867420301021.

[5] W. P. Walters and R. Barzilay. "Applications of Deep Learning in Molecule Generation and Molecular Property Prediction". In: *Accounts of Chemical Research* 54.2 (2021). PMID: 33370107, pp. 263–270. DOI: 10.1021/acs.accounts.0c00699. eprint: https://doi.org/10.1021/acs.accounts.0c00699. URL: https://doi.org/10.1021/acs.accounts.0c00699.

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

[7] J. Klicpera, S. Giri, J. T. Margraf, and S. Günnemann. "Fast and Uncertainty-Aware Directional Message Passing for Non-Equilibrium Molecules". In: *NeurIPS-W*. 2020.

[8] 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. URL: https://arxiv.org/abs/2102.05013.

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

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

[11] 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. URL: https://openreview.net/forum?id=pAbm1qfheGk.

[12] 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. URL: http://proceedings.mlr.press/v139/shi21b.html.

[13] 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].

[14] W. Hu, B. Liu, J. Gomes, M. Zitnik, P. Liang, V. S. Pande, and J. Leskovec. "Strategies for Pre-training Graph Neural Networks". In: *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL: https://openreview.net/forum?id=HJlWWJSFDH.

[15] 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. URL: https://proceedings.neurips.cc/paper/2020/hash/3fe230348e9a12c13120749e3f9fa4cd-Abstract.html.

[16] M. Xu, H. Wang, B. Ni, H. Guo, and J. Tang. "Self-supervised Graph-level Representation Learning with Local and Global Structure". 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. 11548–11558. URL: http://proceedings.mlr.press/v139/xu21g.html.

[17] S. J. Pan and Q. Yang. "A Survey on Transfer Learning". In: *IEEE Trans. Knowl. Data Eng.* 22.10 (2010), pp. 1345–1359. DOI: 10.1109/TKDE.2009.191. URL: https://doi.org/10.1109/TKDE.2009.191.

[18] J.-B. Grill, F. Strub, F. Altché, C. Tallec, P. H. Richemond, E. Buchatskaya, C. Doersch, B. Á. Pires, Z. Guo, M. G. Azar, B. Piot, K. Kavukcuoglu, R. Munos, and M. Valko. "Bootstrap Your Own Latent - A New Approach to Self-Supervised Learning". 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. URL: https://proceedings.neurips.cc/paper/2020/hash/f3ada80d5c4ee70142b17b8192b2958e-Abstract.html.

[19] D. Weininger. "SMILES, a chemical language and information system. 1. Introduction to methodology and encoding rules". In: *Journal of Chemical Information and Computer Sciences* 28.1 (1988), pp. 31–36. DOI: 10.1021/ci00057a005. eprint: https://pubs.acs.org/doi/pdf/10.1021/ci00057a005. URL: https://pubs.acs.org/doi/abs/10.1021/ci00057a005.

[20] 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. URL: https://doi.org/10.1021/acs.jcim.9b00237.

[21] G. W. Bemis and M. A. Murcko. "The Properties of Known Drugs. 1. Molecular Frameworks". In: *Journal of Medicinal Chemistry* 39.15 (1996). PMID: 8709122, pp. 2887–2893. DOI: 10.1021/jm9602928. eprint: https://doi.org/10.1021/jm9602928. URL: https://doi.org/10.1021/jm9602928.

[22] G. Landrum. "RDKit: Open-Source Cheminformatics Software". In: (2016). URL: https://github.com/rdkit/rdkit/releases/tag/Release_2016_09_4.

[23] L. Chan, G. R. Hutchison, and G. M. Morris. "Bayesian optimization for conformer generation". In: *Journal of Cheminformatics* 11.1 (May 2019), p. 32. ISSN: 1758-2946. DOI: 10.1186/s13321-019-0354-7. URL: https://doi.org/10.1186/s13321-019-0354-7.

[24] P. C. D. Hawkins, A. G. Skillman, G. L. Warren, B. A. Ellingson, and M. T. Stahl. "Conformer Generation with OMEGA: Algorithm and Validation Using High Quality Structures from the Protein Databank and Cambridge Structural Database". In: *Journal of Chemical Information and Modeling* 50.4 (2010). PMID: 20235588, pp. 572–584. DOI: 10.1021/ci100031x. eprint: https://doi.org/10.1021/ci100031x. URL: https://doi.org/10.1021/ci100031x.

[25] S. Grimme. "Exploration of Chemical Compound, Conformer, and Reaction Space with Meta-Dynamics Simulations Based on Tight-Binding Quantum Chemical Calculations". In: *Journal of Chemical Theory and Computation* 15.5 (2019). PMID: 30943025, pp. 2847–2862. DOI: 10.1021/acs.jctc.9b00143. eprint: https://doi.org/10.1021/acs.jctc.9b00143. URL: https://doi.org/10.1021/acs.jctc.9b00143.

[26] L. A. Nguyen, H. He, and C. Pham-Huy. "Chiral drugs: an overview". eng. In: *International journal of biomedical science : IJBS* 2.2 (June 2006). PMC3614593[pmcid], pp. 85–100. ISSN: 1550-9702. URL: https://pubmed.ncbi.nlm.nih.gov/23674971.

[27] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl. "Neural Message Passing for Quantum Chemistry". In: *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*. Ed. by D. Precup and Y. W. Teh. Vol. 70. Proceedings of Machine Learning Research. PMLR, 2017, pp. 1263–1272. URL: http://proceedings.mlr.press/v70/gilmer17a.html.

[28] S. Kullback. *Information Theory and Statistics*. New York: Wiley, 1959.

[29] T. B. Kimber, Y. Chen, and A. Volkamer. "Deep Learning in Virtual Screening: Recent Applications and Developments". In: *International Journal of Molecular Sciences* 22.9 (2021). ISSN: 1422-0067. DOI: 10.3390/ijms22094435. URL: https://www.mdpi.com/1422-0067/22/9/4435.

[30] A. Mauri, V. Consonni, and R. Todeschini. "Molecular Descriptors". In: *Handbook of Computational Chemistry*. Ed. by J. Leszczynski, A. Kaczmarek-Kedziera, T. Puzyn, M. G. Papadopoulos, H. Reis, and M. K. Shukla. Cham: Springer International Publishing, 2017, pp. 2065–2093. ISBN: 978-3-319-27282-5. DOI: 10.1007/978-3-319-27282-5_51. URL: https://doi.org/10.1007/978-3-319-27282-5_51.

[31] R. Todeschini and V. Consonni. "Molecular Descriptors for Chemoinformatics". In: *Methods and Principles in Medicinal Chemistry* (2009). DOI: 10.1002/9783527628766.

[32] A. Capecchi, D. Probst, and J.-L. Reymond. "One molecular fingerprint to rule them all: drugs, biomolecules, and the metabolome". In: *Journal of Cheminformatics* 12.1 (June 2020), p. 43. ISSN: 1758-2946. DOI: 10.1186/s13321-020-00445-4. URL: https://doi.org/10.1186/s13321-020-00445-4.

[33] W. P. Walters and R. Barzilay. "Applications of Deep Learning in Molecule Generation and Molecular Property Prediction". In: *Accounts of Chemical Research* 54.2 (2021). PMID: 33370107, pp. 263–270. DOI: 10.1021/acs.accounts.0c00699. eprint: https://doi.org/10.1021/acs.accounts.0c00699. URL: https://doi.org/10.1021/acs.accounts.0c00699.

[34] 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. URL: https://doi.org/10.1038/sdata.2014.22.

[35] M. Krenn, F. Häse, A. Nigam, P. Friederich, and A. Aspuru-Guzik. "Self-referencing embedded strings (SELFIES): A 100% robust molecular string representation". In: *Machine Learning: Science and Technology* 1.4 (Nov. 2020), p. 045024. DOI: 10.1088/2632-2153/aba947. URL: https://doi.org/10.1088/2632-2153/aba947.

[36] 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. URL: https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

[37] A. Rives, J. Meier, T. Sercu, S. Goyal, Z. Lin, J. Liu, D. Guo, M. Ott, C. L. Zitnick, J. Ma, and R. Fergus. "Biological structure and function emerge from scaling unsupervised learning to 250 million protein sequences". In: *Proceedings of the National Academy of Sciences* 118.15 (2021). ISSN: 0027-8424. DOI: 10.1073/pnas.2016239118. eprint: https://www.pnas.org/content/118/15/e2016239118.full.pdf. URL: https://www.pnas.org/content/118/15/e2016239118.

[38] J. Zhou, G. Cui, S. Hu, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, and M. Sun. "Graph neural networks: A review of methods and applications". In: *AI Open* 1 (2020), pp. 57–81. DOI: 10.1016/j.aiopen.2021.01.001. URL: https://doi.org/10.1016/j.aiopen.2021.01.001.

[39] D. Duvenaud, D. Maclaurin, J. Aguilera-Iparraguirre, R. Gómez-Bombarelli, T. Hirzel, A. Aspuru-Guzik, and R. P. Adams. "Convolutional Networks on Graphs for Learning Molecular Fingerprints". In: *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*. Ed. by C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett. 2015, pp. 2224–2232. URL: https://proceedings.neurips.cc/paper/2015/hash/f9be311e65d81a9ad8150a60844bb94c-Abstract.html.

[40] W. Hu, M. Fey, M. Zitnik, Y. Dong, H. Ren, B. Liu, M. Catasta, and J. Leskovec. "Open Graph Benchmark: Datasets for Machine Learning on Graphs". 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. URL: https://proceedings.neurips.cc/paper/2020/hash/fb60d411a5c5b72b2e7d3527cfc84fd0-Abstract.html.

[41] P. Battaglia, J. B. C. Hamrick, V. Bapst, A. Sanchez, V. Zambaldi, M. Malinowski, A. Tacchetti, D. Raposo, A. Santoro, R. Faulkner, C. Gulcehre, F. Song, A. Ballard, J. Gilmer, G. E. Dahl, A. Vaswani, K. Allen, C. Nash, V. J. Langston, C. Dyer, N. Heess, D. Wierstra, P. Kohli, M. Botvinick, O. Vinyals, Y. Li, and R. Pascanu. "Relational inductive biases, deep learning, and graph networks". In: *arXiv* (2018). URL: https://arxiv.org/pdf/1806.01261.pdf.

[42] F. Geerts, F. Mazowiecki, and G. Perez. "Let's Agree to Degree: Comparing Graph Convolutional Networks in the Message-Passing Framework". In: *Proceedings of the 38th International Conference on Machine Learning*. Ed. by M. Meila and T. Zhang. Vol. 139. Proceedings of Machine Learning Research. PMLR, July 2021, pp. 3640–3649. URL: https://proceedings.mlr.press/v139/geerts21a.html.

[43] G. Bouritsas, F. Frasca, S. Zafeiriou, and M. M. Bronstein. "Improving Graph Neural Network Expressivity via Subgraph Isomorphism Counting". In: *CoRR* abs/2006.09252 (2020). arXiv: 2006.09252. URL: https://arxiv.org/abs/2006.09252.

[44] 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. URL: https://proceedings.neurips.cc/paper/2020/file/99cad265a1768cc2dd013f0e740300ae-Paper.pdf.

[45] D. Beaini, S. Passaro, V. Létourneau, W. L. Hamilton, G. Corso, and P. Lió. "Directional Graph Networks". 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. 748–758. URL: http://proceedings.mlr.press/v139/beani21a.html.

[46] C. Bodnar, F. Frasca, N. Otter, Y. G. Wang, P. Liò, G. Montúfar, and M. M. Bronstein. "Weisfeiler and Lehman Go Cellular: CW Networks". In: *CoRR* abs/2106.12575 (2021). arXiv: 2106.12575. URL: https://arxiv.org/abs/2106.12575.

[47] M. Brockschmidt. "GNN-FiLM: Graph Neural Networks with Feature-wise Linear Modulation". 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. 1144–1152. URL: http://proceedings.mlr.press/v119/brockschmidt20a.html.

[48] B. Tang, S. T. Kramer, M. Fang, Y. Qiu, Z. Wu, and D. Xu. "A self-attention based message passing neural network for predicting molecular lipophilicity and aqueous solubility". In: *J. Cheminformatics* 12.1 (2020), p. 15. DOI: 10.1186/s13321-020-0414-z. URL: https://doi.org/10.1186/s13321-020-0414-z.

[49] M. Withnall, E. Lindelöf, O. Engkvist, and H. Chen. "Building attention and edge message passing neural networks for bioactivity and physical-chemical property prediction". In: *J. Cheminformatics* 12.1 (2020), p. 1. DOI: 10.1186/s13321-019-0407-y. URL: https://doi.org/10.1186/s13321-019-0407-y.

[50] J. Li, D. Cai, and X. He. "Learning Graph-Level Representation for Drug Discovery". In: *CoRR* abs/1709.03741 (2017). arXiv: 1709.03741. URL: http://arxiv.org/abs/1709.03741.

[51] W. Torng and R. B. Altman. "Graph Convolutional Neural Networks for Predicting Drug-Target Interactions". In: *J. Chem. Inf. Model.* 59.10 (2019), pp. 4131–4149. DOI: 10.1021/acs.jcim.9b00628. URL: https://doi.org/10.1021/acs.jcim.9b00628.

[52] C. W. Coley, W. Jin, L. Rogers, T. F. Jamison, T. S. Jaakkola, W. H. Green, R. Barzilay, and K. F. Jensen. "A graph-convolutional neural network model for the prediction of chemical reactivity". In: *Chem. Sci.* 10 (2 2019), pp. 370–377. DOI: 10.1039/C8SC04228D. URL: http://dx.doi.org/10.1039/C8SC04228D.

[53] T. S. Hy, S. Trivedi, H. Pan, B. M. Anderson, and R. Kondor. "Predicting molecular properties with covariant compositional networks". In: *The Journal of Chemical Physics* 148.24 (2018), p. 241745. DOI: 10.1063/1.5024797. eprint: https://doi.org/10.1063/1.5024797. URL: https://doi.org/10.1063/1.5024797.

[54] O. T. Unke and M. Meuwly. "PhysNet: A Neural Network for Predicting Energies, Forces, Dipole Moments, and Partial Charges". In: *Journal of Chemical Theory and Computation* 15.6 (2019). PMID: 31042390, pp. 3678–3693. DOI: 10.1021/acs.jctc.9b00181. eprint: https://doi.org/10.1021/acs.jctc.9b00181. URL: https://doi.org/10.1021/acs.jctc.9b00181.

[55] Z. Wu, B. Ramsundar, E. N. Feinberg, J. Gomes, C. Geniesse, A. S. Pappu, K. Leswing, and V. S. Pande. "MoleculeNet: A Benchmark for Molecular Machine Learning". In: *CoRR* abs/1703.00564 (2017). arXiv: 1703.00564. URL: http://arxiv.org/abs/1703.00564.

[56] V. P. Dwivedi and X. Bresson. "A Generalization of Transformer Networks to Graphs". In: *CoRR* abs/2012.09699 (2020). arXiv: 2012.09699. URL: https://arxiv.org/abs/2012.09699.

[57] D. Kreuzer, D. Beaini, W. L. Hamilton, V. Létourneau, and P. Tossou. "Rethinking Graph Transformers with Spectral Attention". In: *CoRR* abs/2106.03893 (2021). arXiv: 2106.03893. URL: https://arxiv.org/abs/2106.03893.

[58] G. Mialon, D. Chen, M. Selosse, and J. Mairal. "GraphiT: Encoding Graph Structure in Transformers". In: *CoRR* abs/2106.05667 (2021). arXiv: 2106.05667. URL: https://arxiv.org/abs/2106.05667.

[59] C. Ying, T. Cai, S. Luo, S. Zheng, G. Ke, D. He, Y. Shen, and T.-Y. Liu. "Do Transformers Really Perform Bad for Graph Representation?" In: *CoRR* abs/2106.05234 (2021). arXiv: 2106.05234. URL: https://arxiv.org/abs/2106.05234.

[60] P. Gainza, F. Sverrisson, F. Monti, E. Rodolà, D. Boscaini, M. M. Bronstein, and B. E. Correia. "Deciphering interaction fingerprints from protein molecular surfaces using geometric deep learning". In: *Nature Methods* 17.2 (Feb. 2020), pp. 184–192. ISSN: 1548-7105. DOI: 10.1038/s41592-019-0666-6. URL: https://doi.org/10.1038/s41592-019-0666-6.

[61] S. K. Mylonas, A. Axenopoulos, and P. Daras. "DeepSurf: A surface-based deep learning approach for the prediction of ligand binding sites on proteins". In: *CoRR* abs/2002.05643 (2020). arXiv: 2002.05643. URL: https://arxiv.org/abs/2002.05643.

[62] Q. Liu, P.-S. Wang, C. Zhu, B. B. Gaines, T. Zhu, J. Bi, and M. Song. "OctSurf: Efficient hierarchical voxel-based molecular surface representation for protein-ligand affinity prediction". In: *Journal of Molecular Graphics and Modelling* 105 (2021), p. 107865. ISSN: 1093-3263. DOI: https://doi.org/10.1016/j.jmgm.2021.107865. URL: https://www.sciencedirect.com/science/article/pii/S1093326321000346.

[63] P. Chen, W. Liu, C.-Y. Hsieh, G. Chen, and P. A. Heng. *Utilizing Edge Features in Graph Neural Networks via Variational Information Maximization*. 2020. URL: https://openreview.net/forum?id=BygZK2VYvB.

[64] 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. URL: https://proceedings.neurips.cc/paper/2017/hash/303ed4c69846ab36c2904d3ba8573050-Abstract.html.

[65] S. L. Batzner, T. E. Smidt, L. Sun, J. P. Mailoa, M. Kornbluth, N. Molinari, and B. Kozinsky. "SE(3)-Equivariant Graph Neural Networks for Data-Efficient and Accurate Interatomic Potentials". In: *CoRR* abs/2101.03164 (2021). arXiv: 2101.03164. URL: https://arxiv.org/abs/2101.03164.

[66] F. Fuchs, D. E. Worrall, V. Fischer, and M. Welling. "SE(3)-Transformers: 3D Roto-Translation Equivariant Attention Networks". 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. URL: https://proceedings.neurips.cc/paper/2020/hash/15231a7ce4ba789d13b722cc5c955834-Abstract.html.

[67] V. G. Satorras, E. Hoogeboom, F. B. Fuchs, I. Posner, and M. Welling. "E(n) Equivariant Normalizing Flows for Molecule Generation in 3D". In: *CoRR* abs/2105.09016 (2021). arXiv: 2105.09016. URL: https://arxiv.org/abs/2105.09016.

[68] S. Axelrod and R. Gómez-Bombarelli. "Molecular machine learning with conformer ensembles". In: *CoRR* abs/2012.08452 (2020). arXiv: 2012.08452. URL: https://arxiv.org/abs/2012.08452.

[69] E. Mansimov, O. Mahmood, S. Kang, and K. Cho. "Molecular Geometry Prediction using a Deep Generative Graph Neural Network". In: *Scientific Reports* 9.1 (Dec. 2019), p. 20381. ISSN: 2045-2322. DOI: 10.1038/s41598-019-56773-5. URL: https://doi.org/10.1038/s41598-019-56773-5.

[70] G. N. C. Simm and J. M. Hernández-Lobato. "A Generative Model for Molecular Distance Geometry". In: *Proceedings of the 37th International Conference on Machine Learning*. Ed. by H. D. III and A. Singh. Vol. 119. Proceedings of Machine Learning Research. PMLR, July 2020, pp. 8949–8958. URL: http://proceedings.mlr.press/v119/simm20a.html.

[71] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol. "Extracting and composing robust features with denoising autoencoders". In: *Machine Learning, Proceedings of the Twenty-Fifth International Conference (ICML 2008), Helsinki, Finland, June 5-9, 2008*. Ed. by W. W. Cohen, A. McCallum, and S. T. Roweis. Vol. 307. ACM International Conference

Proceeding Series. ACM, 2008, pp. 1096–1103. DOI: 10.1145/1390156.1390294. URL: https://doi.org/10.1145/1390156.1390294.

[72] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*. Ed. by J. Burstein, C. Doran, and T. Solorio. Association for Computational Linguistics, 2019, pp. 4171–4186. DOI: 10.18653/v1/n19-1423. URL: https://doi.org/10.18653/v1/n19-1423.

[73] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby. "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale". In: *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. URL: https://openreview.net/forum?id=YicbFdNTTy.

[74] S. Chopra, R. Hadsell, and Y. LeCun. "Learning a Similarity Metric Discriminatively, with Application to Face Verification". In: *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2005), 20-26 June 2005, San Diego, CA, USA*. IEEE Computer Society, 2005, pp. 539–546. DOI: 10.1109/CVPR.2005.202. URL: https://doi.org/10.1109/CVPR.2005.202.

[75] F. Schroff, D. Kalenichenko, and J. Philbin. "FaceNet: A unified embedding for face recognition and clustering". In: *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*. IEEE Computer Society, 2015, pp. 815–823. DOI: 10.1109/CVPR.2015.7298682. URL: https://doi.org/10.1109/CVPR.2015.7298682.

[76] A. van den Oord, Y. Li, and O. Vinyals. "Representation Learning with Contrastive Predictive Coding". In: *CoRR* abs/1807.03748 (2018). arXiv: 1807.03748. URL: http://arxiv.org/abs/1807.03748.

[77] M. I. Belghazi, A. Baratin, S. Rajeswar, S. Ozair, Y. Bengio, R. D. Hjelm, and A. C. Courville. "Mutual Information Neural Estimation". In: *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*. Ed. by J. G. Dy and A. Krause. Vol. 80. Proceedings of Machine Learning Research. PMLR, 2018, pp. 530–539. URL: http://proceedings.mlr.press/v80/belghazi18a.html.

[78] M. Gutmann and A. Hyvärinen. "Noise-contrastive estimation: A new estimation principle for unnormalized statistical models". In: *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2010, Chia Laguna Resort, Sardinia, Italy, May 13-15, 2010*. Ed. by Y. W. Teh and D. M. Titterington. Vol. 9. JMLR Proceedings. JMLR.org, 2010, pp. 297–304. URL: http://proceedings.mlr.press/v9/gutmann10a.html.

[79] R. D. Hjelm, A. Fedorov, S. Lavoie-Marchildon, K. Grewal, P. Bachman, A. Trischler, and Y. Bengio. "Learning deep representations by mutual information estimation and maximization". In: *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL: https://openreview.net/forum?id=Bklr3j0cKX.

[80] K. He, H. Fan, Y. Wu, S. Xie, and R. B. Girshick. "Momentum Contrast for Unsupervised Visual Representation Learning". In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*. Computer Vision Foundation / IEEE, 2020, pp. 9726–9735. DOI: 10.1109/CVPR42600.2020.00975. URL: https://doi.org/10.1109/CVPR42600.2020.00975.

[81] 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. URL: http://proceedings.mlr.press/v119/chen20j.html.

[82] M. Caron, I. Misra, J. Mairal, P. Goyal, P. Bojanowski, and A. Joulin. "Unsupervised Learning of Visual Features by Contrasting Cluster Assignments". 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. URL: https://proceedings.neurips.cc/paper/2020/hash/70feb62b69f16e0238f741fab228fec2-Abstract.html.

[83] Y. Tian, X. Chen, and S. Ganguli. "Understanding self-supervised learning dynamics without contrastive pairs". 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. 10268–10278. URL: http://proceedings.mlr.press/v139/tian21a.html.

[84] P. H. Richemond, J.-B. Grill, F. Altché, C. Tallec, F. Strub, A. Brock, S. L. Smith, S. De, R. Pascanu, B. Piot, and M. Valko. "BYOL works even without batch statistics". In: *CoRR* abs/2010.10241 (2020). arXiv: 2010.10241. URL: https://arxiv.org/abs/2010.10241.

[85] X. Chen and K. He. "Exploring Simple Siamese Representation Learning". In: *CoRR* abs/2011.10566 (2020). arXiv: 2011.10566. URL: https://arxiv.org/abs/2011.10566.

[86] M. Caron, H. Touvron, I. Misra, H. Jégou, J. Mairal, P. Bojanowski, and A. Joulin. "Emerging Properties in Self-Supervised Vision Transformers". In: *CoRR* abs/2104.14294 (2021). arXiv: 2104.14294. URL: https://arxiv.org/abs/2104.14294.

[87] M. Assran, M. Caron, I. Misra, P. Bojanowski, A. Joulin, N. Ballas, and M. Rabbat. *Semi-Supervised Learning of Visual Features by Non-Parametrically Predicting View Assignments with Support Samples*. 2021. arXiv: 2104.13963 [cs.CV].

[88] A. Recasens, P. Luc, J.-B. Alayrac, L. Wang, F. Strub, C. Tallec, M. Malinowski, V. Patraucean, F. Altché, M. Valko, J.-B. Grill, A. van den Oord, and A. Zisserman. "Broaden Your Views for Self-Supervised Video Learning". In: *CoRR* abs/2103.16559 (2021). arXiv: 2103.16559. URL: https://arxiv.org/abs/2103.16559.

[89] J. Zbontar, L. Jing, I. Misra, Y. LeCun, and S. Deny. "Barlow Twins: Self-Supervised Learning via Redundancy Reduction". 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. 12310–12320. URL: http://proceedings.mlr.press/v139/zbontar21a.html.

[90] A. Bardes, J. Ponce, and Y. LeCun. "VICReg: Variance-Invariance-Covariance Regularization for Self-Supervised Learning". In: *CoRR* abs/2105.04906 (2021). arXiv: 2105.04906. URL: https://arxiv.org/abs/2105.04906.

[91] Y. Xie, Z. Xu, Z. Wang, and S. Ji. "Self-Supervised Learning of Graph Neural Networks: A Unified Review". In: *CoRR* abs/2102.10757 (2021). arXiv: 2102.10757. URL: https://arxiv.org/abs/2102.10757.

[92] T. N. Kipf and M. Welling. "Variational Graph Auto-Encoders". In: *NIPS Workshop on Bayesian Deep Learning* (2016).

[93] P. Velickovic, W. Fedus, W. L. Hamilton, P. Liò, Y. Bengio, and R. D. Hjelm. "Deep Graph Infomax". In: *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL: https://openreview.net/forum?id=rklz9iAcKQ.

[94] F.-Y. Sun, J. Hoffmann, V. Verma, and J. Tang. "InfoGraph: Unsupervised and Semi-supervised Graph-Level Representation Learning via Mutual Information Maximization". In: *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL: https://openreview.net/forum?id=r1lfF2NYvH.

[95] K. Hassani and A. H. K. Ahmadi. "Contrastive Multi-View Representation Learning on Graphs". 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. 4116–4126. URL: http://proceedings.mlr.press/v119/hassani20a.html.

[96] Z. Peng, W. Huang, M. Luo, Q. Zheng, Y. Rong, T. Xu, and J. Huang. "Graph Representation Learning via Graphical Mutual Information Maximization". In: *WWW '20: The Web Conference 2020, Taipei, Taiwan, April 20-24, 2020*. Ed. by Y. Huang, I. King, T.-Y. Liu, and M. van Steen. ACM, 2020, pp. 259–270. DOI: 10.1145/3366423.3380112. URL: https://doi.org/10.1145/3366423.3380112.

[97] Y. Zhu, Y. Xu, F. Yu, Q. Liu, S. Wu, and L. Wang. "Graph Contrastive Learning with Adaptive Augmentation". In: *WWW '21: The Web Conference 2021, Virtual Event / Ljubljana, Slovenia, April 19-23, 2021*. Ed. by J. Leskovec, M. Grobelnik, M. Najork,

J. Tang, and L. Zia. ACM, 2021, pp. 2069–2080. DOI: 10.1145/3442381.3449802. URL: https://doi.org/10.1145/3442381.3449802.

[98] Y. You, T. Chen, Y. Shen, and Z. Wang. "Graph Contrastive Learning Automated". 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. 12121–12132. URL: http://proceedings.mlr.press/v139/you21a.html.

[99] S. Chithrananda, G. Grand, and B. Ramsundar. "ChemBERTa: Large-Scale Self-Supervised Pretraining for Molecular Property Prediction". In: *CoRR* abs/2010.09885 (2020). arXiv: 2010.09885. URL: https://arxiv.org/abs/2010.09885.

[100] L. Maziarka, T. Danel, S. Mucha, K. Rataj, J. Tabor, and S. Jastrzebski. "Molecule Attention Transformer". In: *CoRR* abs/2002.08264 (2020). arXiv: 2002.08264. URL: https://arxiv.org/abs/2002.08264.

[101] Y. Wang, J. Wang, Z. Cao, and A. B. Farimani. "MolCLR: Molecular Contrastive Learning of Representations via Graph Neural Networks". In: *CoRR* abs/2102.10056 (2021). arXiv: 2102.10056. URL: https://arxiv.org/abs/2102.10056.

[102] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. "ImageNet: A Large-Scale Hierarchical Image Database". In: *CVPR09*. 2009.

[103] S. Thakoor, C. Tallec, M. G. Azar, R. Munos, P. Velickovic, and M. Valko. "Bootstrapped Representation Learning on Graphs". In: *CoRR* abs/2102.06514 (2021). arXiv: 2102.06514. URL: https://arxiv.org/abs/2102.06514.

[104] Z. T. Kefato and S. Girdzijauskas. "Self-supervised Graph Neural Networks without explicit negative sampling". In: *CoRR* abs/2103.14958 (2021). arXiv: 2103.14958. URL: https://arxiv.org/abs/2103.14958.

[105] P. Bielak, T. Kajdanowicz, and N. V. Chawla. "Graph Barlow Twins: A self-supervised representation learning framework for graphs". In: *CoRR* abs/2106.02466 (2021). arXiv: 2106.02466. URL: https://arxiv.org/abs/2106.02466.

[106] M. D. Donsker and S. R. S. Varadhan. "Asymptotic evaluation of certain markov process expectations for large time, I". In: *Communications on Pure and Applied Mathematics* 28.1 (1975), pp. 1–47. DOI: https://doi.org/10.1002/cpa.3160280102. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/cpa.3160280102. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/cpa.3160280102.

[107] S. Nowozin, B. Cseke, and R. Tomioka. "f-GAN: Training Generative Neural Samplers using Variational Divergence Minimization". In: *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*. Ed. by D. D. Lee, M. Sugiyama, U. von Luxburg, I. Guyon, and R. Garnett. 2016, pp. 271–279. URL: https://proceedings.neurips.cc/paper/2016/hash/cedebb6e872f539bef8c3f919874e9d7-Abstract.html.

[108] R. Hadsell, S. Chopra, and Y. LeCun. "Dimensionality Reduction by Learning an Invariant Mapping". In: *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2006), 17-22 June 2006, New York, NY, USA*. IEEE Computer Society, 2006, pp. 1735–1742. DOI: 10.1109/CVPR.2006.100. URL: https://doi.org/10.1109/CVPR.2006.100.

[109] T. Hua, W. Wang, Z. Xue, Y. Wang, S. Ren, and H. Zhao. "On Feature Decorrelation in Self-Supervised Learning". In: *CoRR* abs/2105.00470 (2021). arXiv: 2105.00470. URL: https://arxiv.org/abs/2105.00470.

[110] A. Gretton, K. M. Borgwardt, M. J. Rasch, B. Schölkopf, and A. Smola. "A Kernel Two-Sample Test". In: *Journal of Machine Learning Research* 13.25 (2012), pp. 723–773. URL: http://jmlr.org/papers/v13/gretton12a.html.

[111] K. T. Schütt, P.-J. Kindermans, H. E. Sauceda, S. Chmiela, A. Tkatchenko, and K.-R. Müller. *SchNet: A continuous-filter convolutional neural network for modeling quantum interactions*. 2017. arXiv: 1706.08566 [stat.ML].

[112] M. M. Bronstein, J. Bruna, T. Cohen, and P. Velickovic. "Geometric Deep Learning: Grids, Groups, Graphs, Geodesics, and Gauges". In: *CoRR* abs/2104.13478 (2021). arXiv: 2104.13478. URL: https://arxiv.org/abs/2104.13478.

[113] 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. URL: https://proceedings.mlr.press/v97/rahaman19a.html.

[114] 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).

[115] 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.

[116] 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. URL: https://openreview.net/forum?id=SJxUjlBtwB.

[117] S. Ioffe and C. Szegedy. "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift". In: *Proceedings of the 32nd International Conference on Machine Learning*. Ed. by F. Bach and D. Blei. Vol. 37. Proceedings of Machine Learning Research. Lille, France: PMLR, July 2015, pp. 448–456. URL: https://proceedings.mlr.press/v37/ioffe15.html.

[118] R. Flamary, N. Courty, A. Gramfort, M. Z. Alaya, A. Boisbunon, S. Chambon, L. Chapel, A. Corenflos, K. Fatras, N. Fournier, L. Gautheron, N. T. Gayraud, H. Janati, A. Rakotomamonjy, I. Redko, A. Rolet, A. Schutz, V. Seguy, D. J. Sutherland, R. Tavenard, A. Tong, and T. Vayer. "POT: Python Optimal Transport". In: *Journal of Machine Learning Research* 22.78 (2021), pp. 1–8. URL: http://jmlr.org/papers/v22/20-451.html.

[119] 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).

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

[121] 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).

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

[123] 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].

[124] K. Clark, M.-T. Luong, Q. V. Le, and C. D. Manning. "ELECTRA: Pre-training Text Encoders as Discriminators Rather Than Generators". In: *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL: https://openreview.net/forum?id=r1xMH1BtvB.

[125] M. Zaheer, S. Kottur, S. Ravanbakhsh, B. Póczos, R. Salakhutdinov, and A. J. Smola. "Deep Sets". 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. 3391–3401. URL: https://proceedings.neurips.cc/paper/2017/hash/f22e4747da1aa27e363d86d40ff442fe-Abstract.html.