

Unsupervised Learning of Molecular Representations for Drug Development

Dissertation

zur Erlangung des Grades eines
Doktors der Naturwissenschaften
(Dr. rer. nat.)
am Fachbereich Mathematik und Informatik
der Freien Universität Berlin

vorgelegt von

Jan Robin Winter

Berlin, 2022

Erstgutachter: Prof. Dr. Frank Noé

Zweitgutachter: Prof. Dr. Andreas Bender

Drittgutachter: Dr. Djork-Arné Clevert

Tag der Disputation: 25.08.2023

Abstract

Representing molecules in a computer-interpretable way plays a crucial role in enabling the application of computational method to the field of chemistry and pharmaceutical drug development in particular. Recently, there has been a surge of interest in using machine learning to predict molecular properties such as the binding affinity to protein targets of interest or to generate molecular structures with desirable properties. However, as chemical entities are challenging to represent in an expressive and computer-interpretable way, much work in the field of cheminformatics has concerned itself with defining clever feature extractors, which encode the chemical graph structure in a uniform, fixed-sized, numerical manner. Recently Deep Neural Networks have shown great success in learning to extract meaningful features directly from raw data representations, outperforming hand-crafted feature extraction protocols and revolutionizing fields such as image analysis or natural language processing. Deep Neural Networks have also been directly applied on raw data representations of molecules such as their structural graph. However, the capabilities of these method in pharmaceutical drug development is usually limited by the scarcity of labeled data as their collection usually involves running expensive wet lab experiments. Unsupervised Learning, on the other hand, is a powerful machine learning strategy that enables the training of Deep Neural Networks without the need of labeled training data. In this thesis we discuss how Unsupervised Learning can be used to train powerful feature extractors on unlabeled chemical structures. We propose for different input representations of molecules (such as line notations, graphs and point clouds) novel methods to extract expressive representations. We show how those representations can efficiently be used as input for downstream molecular property prediction models or to generate novel molecules with desirable properties. Moreover, we discuss how certain symmetries of molecular representations are crucial to respect (e.g. permutation invariance of molecular graphs or rotation and translation invariance of molecular conformations) and develop novel methods particularly designed to extract invariant representations.

Zusammenfassung

Die Darstellung von Molekülen in einer computerlesbaren Form spielt eine unverzichtbare Rolle für die Anwendung von Computermodellen in der Chemie und der pharmazeutischen Wirkstoffentwicklung. Seit Kurzem gibt es zudem ein wachsendes Interesse an der Nutzung von Machine Learning Algorithmen zur Vorhersage von Moleküleigenschaften wie der Bindungsaffinität zu bestimmten Proteinen oder der Generierung von neuen Molekülen mit wünschenswerten Eigenschaften. Da es jedoch schwierig ist, chemische Entitäten in einer expressiven und computerinterpretierbaren Weise darzustellen, haben sich viele Arbeiten auf dem Gebiet der Chemieinformatik mit der Entwicklung von cleveren Darstellungsalgorithmen befasst, bei denen die chemische Struktur in einer einheitlichen, numerischen Form kodiert wird. In jüngster Zeit haben tiefe neuronale Netze großen Nutzen beim Erlernen der Extraktion aussagekräftiger Darstellungen direkt aus Rohdatendarstellungen gezeigt, wodurch sie von Hand erstellten Protokolle zur Darstellung von Daten übertrafen und Bereiche wie die Bildanalyse oder die automatische Spracherkennung revolutionierten. Tiefe neuronale Netze wurden auch direkt auf Rohdatendarstellungen von Molekülen, z. B. deren Strukturformel angewendet. Die Möglichkeiten dieser Methoden sind jedoch in der pharmazeutischen Wirkstoffentwicklung durch den Mangel an Daten mit bekannten Eigenschaften (sogenannte gelabelte Daten) in der Regel begrenzt, da die Erfassung dieser Daten oft die Durchführung teurer Laborexperimente erfordert. Dahingegen ist sogenanntes Unüberwachtes Lernen eine leistungsstarke Strategie des maschinellen Lernens, die das Training von tiefen neuronalen Netzen ermöglicht, ohne dass gelabelte Trainingsdaten benötigt werden. In dieser Arbeit diskutieren wir, wie Unüberwachtes Lernen verwendet werden kann, um nützliche Darstellungsextraktoren auf nicht gelabelten chemischen Strukturen zu trainieren. Wir haben für verschiedene Rohdatendarstellungen von Molekülen (z. B. Textdarstellungen, Graphen und Punktwolken) neue Methoden zur Extraktion aussagekräftiger Darstellungen entwickelt, die als Grundlage für nachgelagerte Modelle zur Vorhersage molekularer Eigenschaften oder zur Generierung neuartiger Moleküle mit erwünschten Eigenschaften verwendet werden können. Darüber hinaus erörtern wir, inwiefern die Berücksichtigung bestimmter Symmetrien in molekularen Darstellungen von entscheidender Bedeutung sind (z. B. Permutationsinvarianz molekularer Graphen oder Rotations- und Translationsinvarianz molekularer Konformationen) und diskutieren neue Methoden, die wir speziell für die Extraktion invarianter Darstellungen entwickelt haben.

Selbstständigkeitserklärung

Name: Winter

Vorname: Jan Robin

Ich erkläre gegenüber der Freien Universität Berlin, dass ich die vorliegende Dissertation selbstständig und ohne Benutzung anderer als der angegebenen Quellen und Hilfsmittel angefertigt habe. Die vorliegende Arbeit ist frei von Plagiaten. Alle Ausführungen, die wörtlich oder inhaltlich aus anderen Schriften entnommen sind, habe ich als solche kenntlich gemacht. Diese Dissertation wurde in gleicher oder ähnlicher Form noch in keinem früheren Promotionsverfahren eingereicht. Mit einer Prüfung meiner Arbeit durch ein Plagiatsprüfungsprogramm erkläre ich mich einverstanden.

Datum: Unterschrift:

Acknowledgements

This thesis is the result of the work done at Bayer AG in collaboration with the Freie Universität Berlin. I would like to express my sincere gratitude to the many people that have contributed to this work in various ways.

First and foremost I would like to thank my industry supervisor Djork-Arné Clevert for introducing me to the field of Deep Learning in Drug Discovery and his continuous support throughout my time at Bayer. I am deeply grateful for his guidance throughout my scientific journey from writing my first simple Neural Network in Python to publishing in NeurIPS.

Furthermore, I would like to extend my profound gratitude to my academic supervisor Prof. Frank Noé for giving me the opportunity to write this thesis in his department where I met a lot of great people and received invaluable scientific insights and inspiration. I am very grateful that he accepted to supervise my work and always supported me with feedback while providing me with the freedom to follow my own ideas.

I would also like to thank all the group members of the Machine Learning Research Team at Bayer who made it such a memorable and enjoyable time.

I want to thank Floriane Montanari for her invaluable help and guidance writing my first scientific papers and sharing so much of her knowledge about computational chemistry with me.

Moreover, I would like to thank all my other collaborators and co-authors: Joren Retel, Paul Kim, Andreas Steffen, Hans Briem, Marco Bertolini and Tuan Le. Without them, much of the work would never have happened.

Last but not least, I thank my family and friends who supported me throughout the time of my research, my parents who fueled my interest in science from the very beginning and Charlotte for her love and encouragement during the whole time.

List of Publications

First-author publications part of this thesis.

- **Winter, Robin**, Montanari, Floriane, Noé, Frank, & Clevert, Djork-Arné (2019). Learning continuous and data-driven molecular descriptors by translating equivalent chemical representations. *Chemical science*, 10(6), 1692-1701.
- **Winter, Robin**, Montanari, Floriane, Steffen, Andreas, Briem, Hans, Noé, Frank, & Clevert, Djork-Arné (2019). Efficient multi-objective molecular optimization in a continuous latent space. *Chemical science*, 10(34), 8016-8024.
- **Winter, Robin**, Retel, Joren, Noé, Frank, Clevert, Djork-Arné & Steffen, Andreas (2020). grünifai: interactive multiparameter optimization of molecules in a continuous vector space. *Bioinformatics*, 36(13), 4093-4094.
- **Winter, Robin**, Noé, Frank & Clevert, Djork-Arné (2021). Auto-encoding molecular conformations. arXiv preprint arXiv:2101.01618
- **Winter, Robin**, Noé, Frank & Clevert, Djork-Arné (2021). Permutation-invariant variational autoencoder for graph-level representation learning. *Advances in Neural Information Processing Systems*, 34.
- **Winter, Robin, Bertolini, Marco**, Le, Tuan, Noé, Frank & Clevert, Djork-Arné (2022). Unsupervised Learning of Group Invariant and Equivariant Representations. arXiv preprint arXiv:2202.07559.

Other publications.

- **Winter, Robin**, & Clevert, Djork-Arné (2017). IVE-GAN: Invariant Encoding Generative Adversarial Networks. arXiv preprint arXiv:1711.08646.
- **Le, Tuan**, Winter, Robin, Noé, Frank, & Clevert, Djork-Arné (2020). Neuraldecipher–reverse-engineering extended-connectivity fingerprints (ECFPs) to their molecular structures. *Chemical science*, 11(38), 10378-10389.
- **Kim, Paul**, Winter, Robin, & Clevert, Djork-Arné (2020). Deep protein-ligand binding prediction using unsupervised learned representations. DOI: 10.26434/chemrxiv.11523117.v1
- **Clevert, Djork-Arné**, Le, Tuan, Winter, Robin, & Montanari, Floriane (2021). Img2Mol–accurate SMILES recognition from molecular graphical depictions. *Chemical science*, 12(42), 14174-14181.
- **Kim, Paul**, Winter, Robin, & Clevert, Djork-Arné (2021). Unsupervised Representation Learning for Proteochemometric Modeling. *International Journal of Molecular Sciences*, 22(23), 12882.

Contents

1	Introduction	9
1.1	Molecular Property Prediction	10
1.2	Hand-Crafted Molecular Representation	10
1.3	Supervised Molecular Representation Learning	12
1.4	Unsupervised Molecular Representation Learning	13
1.5	Generative Models	15
1.6	Representing Molecular Conformations	17
1.7	Representing Molecular Graphs	19
1.8	Group Invariant Representation Learning	21
2	Publications	23
2.1	Publication 1: Learning Continuous and Data-Driven Molecular Descriptors by Translating Equivalent Chemical Representations	25
2.2	Publication 2: Efficient Multi-Objective Molecular Optimization in a Continuous Latent Space	53
2.3	Publication 3: grünifai: Interactive Multiparameter Optimization of Molecules in a Continuous Vector Space	69
2.4	Publication 4: Auto-Encoding Molecular Conformations	73
2.5	Publication 5: Permutation-Invariant Variational Autoencoder for Graph-Level Representation Learning	81
2.6	Publication 6: Unsupervised Learning of Group Invariant and Equivariant Representations	103
3	Conclusion	131

Chapter 1

Introduction

Representing molecules in a computer-interpretable way plays a crucial role in enabling the application of computational method to the field of chemistry and manifests in the field known as *cheminformatics* [Todeschini and Consonni, 2008]. A major application of cheminformatics methods is the prediction of molecular properties such as biological activity or physicochemical properties and are essential for pharmaceutical drug development [Bender and Brown, 2018]. In fact, the average time to bring a drug from the first idea to the market takes approximately 10-15 years and costs ~ 1 billion USD [Wouters et al., 2020, Ertl, 2014]. Moreover, it was estimated that approximately one year of patient life is lost for every 12 seconds of delay in the drug development process [Stewart et al., 2015]. On the other hand, the space of potential drug candidates is vast (10^{23} – 10^{60} molecules) [Polishchuk et al., 2013], emphasizing the necessity of computational method to help in the search for desirable compounds.

In the following we describe why expressive molecular representations are important and how they can be generated and utilized for pharmaceutical drug development. In particular, we discuss the published works that are part of this thesis and put them in the context of related works.

1.1 Molecular Property Prediction

A commonly employed cheminformatics methodology are so-called *Quantitative Structure-Activity Relationship* (QSAR) or *Quantitative Structure-Property Relationship* (QSPR) models [Cherkasov et al., 2014]. As the name suggests, these models aim to link chemical similarity to molecular property similarity, following a fundamental assumption of medicinal chemistry: similar chemical structure results into similar bio-chemical function [Bajorath, 2004, Bender and Glen, 2004]. In practice, such models are designed by fitting a parameterized function $f_\theta : M \rightarrow Y$, on a dataset of molecules $m_i \in M$ with associated properties $y_i \in Y$ (e.g. measured in an experiment). After fitting the function on a set of compounds with known properties, the resulting function can be used to evaluate novel compounds *in silico*, e.g. to prioritize and select compounds to be evaluated next in an *in vitro* experiment, potentially saving significant amounts of time and money.

Function f_θ is usually parameterized by a machine learning method such as a Logistic Regression, Support Vector Machine or Artificial Neural Network. However, such models usually require a fixed-sized numerical input. Hence, the question arises: How can the physical entity of a molecule be mapped to such a fixed-sized numerical representation, while being expressive enough to reflect (bio-) chemical similarities of different compounds and thus be useful to fit downstream models?

1.2 Hand-Crafted Molecular Representation

Probably the most common ways to represent chemical entities is either by their *molecular formula*, i.e. the atomic composition of a molecule (e.g. H₂O) or their *structural formula*, which also includes the molecule’s atoms spacial arrangement and bonding information. The latter is usually represented by a graph, where nodes represent atoms (including element type, charge, etc.) and edges represent bonds (including bond type). Alternatively the molecular topology can be represented by a line notation like the International Chemical Identifier (InChi) [McNaught, 2006] or the Simplified Molecular-Input Line-Entry System (SMILES) [Weininger, 1988]. In Figure 1.1 we depict these aforementioned representations for an example molecule. Although expressive, these representations are not fixed-size (e.g. for different molecules with different number of atoms) and do not properly reflect chemical similarity (e.g., a SMILES string might change significantly upon changing a single atom).

Molecules can also be represented by scalar descriptors of either experimentally measured (e.g. octanol-water partition coefficient, aqueous solubility) or calculated (number of carbon atoms, number of proton donors/acceptors, molecular weight, total polar surface area, etc.) properties. Although these descriptors are fixed-sized and can reflect chemical similarity (as expressed by the used properties), they might lack in expressiveness, as similar molecules might have

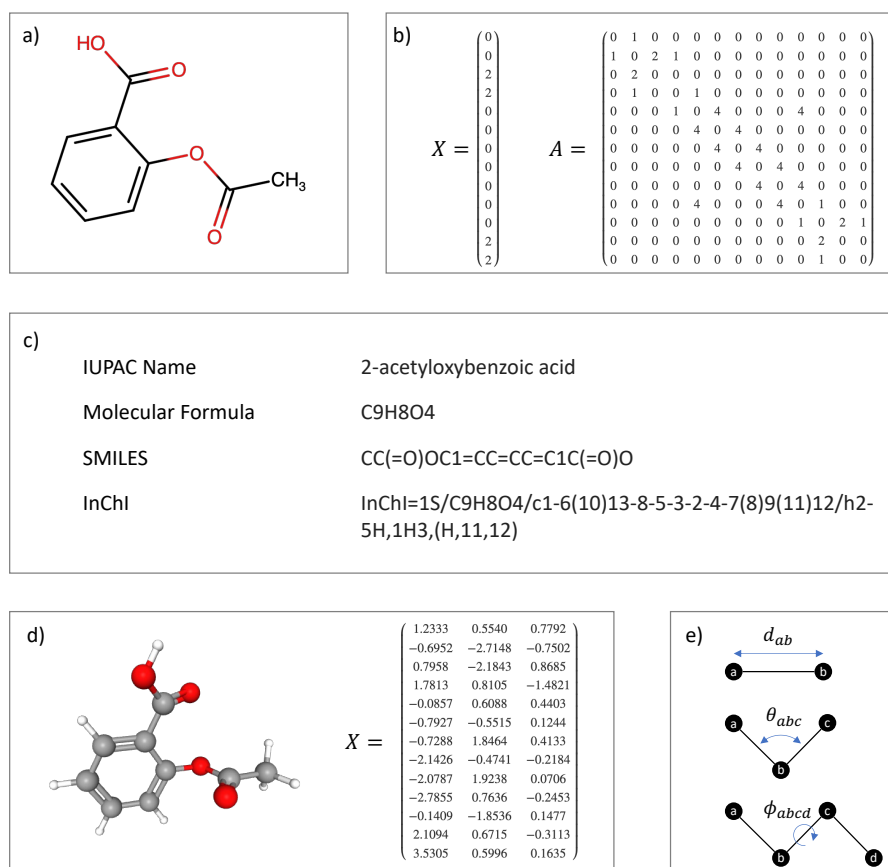


Figure 1.1: Different representations of the drug Aspirin. a) Structural formula. b) Matrix representation of the structural formula (graph), with node features X and edge features A . Here, different atom and edge types are represented by different integers (X : 0 for carbon, 2 for oxygen, A : 0 for no edge, 1 for single bond, 2 for double bond, 4 for aromatic bond). c) Different line notations of Aspirin. d) Example conformation, represented by Cartesian coordinates X (only heavy atoms). e) Definition of internal coordinates: bond length d_{ab} , bond angle θ_{abc} and dihedral angle $\Phi_{a,b,c,d}$.

identical properties or important properties are not reflected by the chosen descriptors.

Over the last decades, a large body of works has been concerned with designing more expressive molecular representations [Todeschini and Consonni, 2008]. As of today, one of the most commonly used molecular descriptor is the extended-connectivity fingerprint (ECFP) [Rogers and Hahn, 2010]. The ECFP descriptor is a variation of the Morgan algorithm [Morgan, 1965] that calculates a fingerprint from a molecule’s graph by hashing individual local atom environments to unique integer values v and transforming (folding) them into a n dimensional vector (e.g. by taking the modulo $v \bmod n$). The resulting vector is a fixed-sized representation which comprises a collection of molecular substructure identifiers. While important global features might be missed and the folding step can lead to so-called *folding collisions* (i.e., different substructures are mapped to same value), ECFPs and its variations have been successively applied in QSAR models for target binding affinity [Wale and Karypis, 2009, Lounkine et al., 2012], ADME (absorption, distribution, metabolism, and excretion) [Glen et al., 2006, Wang et al., 2012, Zang et al., 2017, Göller et al., 2020] and toxicity [Mayr et al., 2016] prediction or virtual screening and similarity search [Hu et al., 2012, Cereto-Massagué et al., 2015].

1.3 Supervised Molecular Representation Learning

With the recent rise of *Deep Learning* [LeCun et al., 2015, Schmidhuber, 2015], many machine learning fields that traditionally relied on extensive feature engineering, such as image analysis, machine translation or speech recognition have recently been revolutionized by deep artificial neural networks trained on raw input signals in an end-to-end fashion [Krizhevsky et al., 2012, Sutskever et al., 2014, Vaswani et al., 2017]. This recent success can be attributed to the significant development of computational hardware such as the *graphics processing unit* (GPU), the increasing amount of (labeled) data available for training and the development of novel algorithms and network architectures such as *Convolutional Neural Networks* (CNN) [LeCun et al., 1998] for image-like data, efficient *Recurrent Neural Networks* (RNN) like *Long Short-Time Memory* (LSTM) [Hochreiter and Schmidhuber, 1997] or the recent *Transformer* network [Vaswani et al., 2017] for sequence-like data.

Motivated by this success, there has been an increasing interest in applying Deep Artificial Neural Networks on graph structured data [Bruna et al., 2013, Henaff et al., 2015, Defferrard et al., 2016, Kipf and Welling, 2016b, Gilmer et al., 2017]. These so-called *Graph Neural Networks* (GNN) can be seen as a generalization of Convolutional Neural Networks from regular grids (e.g. images) to arbitrary graphs with varying node degrees. Each GNN layer updates each node by aggregating messages passed from neighbouring nodes, where messages are

usually outputs of a learnable parameterized function that takes the connected node (hidden) features and potential edge features as input. This procedure gives GNNs also the name *Message Passing Neural Network* (MPNN) [Gilmer et al., 2017].

One major application of GNNs are molecular graphs. In their seminal paper, Duvenaud et al. [Duvenaud et al., 2015] proposed a GNN that was trained end-to-end on molecular graphs for aqueous solubility, drug efficacy and photovoltaic efficiency prediction, outperforming classical ECFP fingerprint based machine learning models. In the following years, there was a surge of work proposing novel GNN architectures and their application on molecular property prediction tasks [Yang et al., 2019, Xiong et al., 2019, Corso et al., 2020, Xu et al., 2018]. Similar to other fields where Deep Learning celebrated success in the recent years, a GNN trained end-to-end on raw input signals (molecular graph) learns to extract its own, task-specific features from the graph. Hence, given enough training data, such features are able to outperform ad-hoc and hand-crafted feature extractors such as the ECFP protocol.

Another line of work uses the SMILES representation of molecules as raw input representation. While holding the same information as molecular graphs, SMILES are a one-dimensional string representation. Hence, Recurrent Neural Networks and Transformers that were mainly developed for Natural Language Processing are a natural choice. SMILES strings effectively represent a molecular graph as a sequence of atom symbols and bond types encountered in a depth-first traversal of the graph. As graphs can be traversed in many different ways, there exist multiple different SMILES strings representing the same molecule. Bjerrum et al. [Bjerrum, 2017] exploited this fact to augment a dataset of SMILES, training an RNN to predict a molecule’s ability to inhibit Dihydrofolate reductase from its SMILES string. Recently, Transformer architectures have been proposed that are pretrained on large unlabeled datasets of SMILES and finetuned on a smaller datasets labeled with molecular property of interest [Wang et al., 2019, Chithrananda et al., 2020].

1.4 Unsupervised Molecular Representation Learning

The aforementioned work on Deep Neural Networks for molecular representation learning is based on supervised learning. In supervised learning a function (here parameterized by a neural network) is trained on a supervisory signal in form of labeled data points, e.g. by optimizing the log-likelihood of predicting the corresponding label given a training sample. While supervised learning is a powerful regime to train a Neural Network to extract expressive task-specific representations to accurately predict molecular properties (e.g. see [Duvenaud et al., 2015]), it can fail to generalize if insufficient labeled training data is available [Bengio et al., 2012]. In fact, a major limiting factor of early drug

discovery is the cost of experimental evaluation, as it involves the synthesis of a compound and measurement of the property of interests in lab experiments. Hence, in an early state of drug development process, only a few hundred of labeled data points might be available, limiting the effectiveness of supervised Deep Neural Networks.

On the other hand, unsupervised learning is tasked to learn representations from data without providing additional labeled information. Probably the most prominent unsupervised learning framework is the so-called *Autoencoder* (AE). An AE consist of an *encoder* and *decoder* function that can be parameterized by a Neural Network. The encoder is tasked to map (encode) input data to a latent representation while the decoder has to reconstruct the input signal as accurately as possible from this latent representation. Often the latent representation is chosen to be of lower dimensionality than the input data, forcing the encoder to compress input signals into a more expressive, higher-level representation. Encoder and decoder are trained jointly on the reconstruction objective, without the need of labels as supervisory signal. For a more in-depth discussion of AE variants and their application we refer to [Tschannen et al., 2018, Bank et al., 2020].

Publication 1: Learning Continuous and Data-Driven Molecular Descriptors by Translating Equivalent Chemical Representations. In the first publication [Winter et al., 2019a], we propose an unsupervised learning framework to extract expressive molecular representations. While labeled data points are usually scarce in the field of drug development, there is effectively an unlimited amount of (unlabeled) drug-like compounds that can be procedurally generated [Ruddigkeit et al., 2012, Polishchuk et al., 2013]. Building up on the seminal work by Gómez-Bombarelli et al. [Gómez-Bombarelli et al., 2018] for molecular generative modeling (see section 1.5 for more details), we train an autoencoder-like model on encoding and decoding molecules to and from a low-dimensional latent representation.

A key aspect of the proposed framework is the use of a so-called sequence-to-sequence model, originally proposed for natural machine translation (NMT) [Sutskever et al., 2014]. Analogous to a NMT model that is trained on encoding a sequence in a query language (e.g. English) and decoding it to a target language (e.g. German), our proposed model is trained on translating between two different line-notations of a molecule, e.g. from InChI to SMILES. In our work we demonstrate how this design choice acts as additional regularization on the encoder, forcing the model to encode only the essential information both input and target line-notation have in common, the molecular topology, and not solely string patterns.

We trained the model on a large dataset of drug-like molecules and demonstrated how the resulting model’s encoder can be utilized to extract expressive molecular representations for downstream tasks. In an extensive benchmark study, we compared descriptors extracted by our model with classical descriptors like

ECFPs and end-to-end trained GNNs and found them to perform competitively in QSAR modeling and significantly outperforming the baselines in ligand-based virtual screening.

As of today, our proposed method is used in a variety of works that either benefit from an expressive molecular representation or from a continuous representation of chemical compounds. For example, many QSAR models of Bayer’s *in silico* ADMET platform are based on our proposed representation [Göller et al., 2020]. In follow-up work, we also demonstrated that our continuous representation of chemical compounds can be utilized as regression target to predict a molecular structure from its supposedly non-invertible ECFP fingerprint [Le et al., 2020] or from a depiction taken from the literature or drawn by hand [Clevert et al., 2021]. Combined with an unsupervised learned protein descriptor (by training a similar approach based on the amino acid sequence of a protein) we also demonstrated how our proposed descriptor can be used for proteochemometric modelling [Kim et al., 2020].

Moreover, Hoffman et al. proposed the use of zeroth order optimization in combination with our pretrained representation to further optimize existing SARS-CoV-2 Main Protease inhibitors toward higher binding affinity [Hoffman et al., 2022]. Kang et al. utilized the pretrained representations to predict the components of kerogen in shale oil/gas from NMR spectra [Kang and Zhao, 2022].

1.5 Generative Models

Another major line of work at the intersection of Deep Neural Networks and cheminformatics is the field of automated *de novo* drug design. The ultimate goal of drug development is to find a chemical compound with desirable properties, such as high binding affinity to a target protein while minimizing off-target binding (i.e., minimizing side effects), keeping a high solubility, and many more. As the space of potential drug candidates is vast (10^{23} – 10^{60} molecules) [Polishchuk et al., 2013] this task can be compared with the proverbial search for the needle in the haystack. Consequently, in the early days of cheminformatics, computer assisted *de novo* design models haven been introduced to help medicinal chemist in this search [Danziger and Dean, 1989, Hartenfeller and Schneider, 2010, Mauser and Guba, 2008, Schneider and Fechner, 2005]. In general an automated *de novo* design model can be split up into a sampling, scoring and optimization part [Schneider and Fechner, 2005], where the model has to efficiently sample novel molecules, score them based on their desirability (e.g. by a QSAR model) and optimize them, i.e navigate the chemical space.

In their seminal work, Segler et al. [Segler et al., 2018] proposed an RNN trained on generating SMILES representations of novel molecules with desirable properties. Similar to a language model trained on next word prediction, their model is trained on next SMILES character prediction. In their work, they

show how fine-tuning of a pretrained model (on a larger dataset to learn general concepts like SMILES syntax) on a set of molecules with desirable properties leads to a model that is biased in generating SMILES strings of molecules with similar properties.

In an alternative approach, Olivecrona et al. [Olivecrona et al., 2017] proposed the use of Reinforcement Learning [Sutton and Barto, 1998] to alter the generation process of an RNN towards more desirable molecules. Work was also done to utilize Generative Adversarial Neural Networks (GANs) [Goodfellow et al., 2014] to generate molecules with desirable properties [Guimaraes et al., 2017, Putin et al., 2018, Schwalbe-Koda and Gómez-Bombarelli, 2020].

One aspect the aforementioned generative methods have in common, is that they are specifically trained to generate molecules that follow a certain distribution, e.g. reassemble molecules from a dataset or maximize a certain property. Hence, if the notion of desirability changes (as it is usually the case over the course of a drug development project), the model needs to be retrained. Moreover, often one is not interested in generating a set of molecules from a distribution of desirable molecules, but rather in further optimizing a few so-called *lead* compounds (e.g. promising compounds that were found in an initial high throughput screen to be active on a target of interest).

An alternative approach to this *distribution-learning* focused methods are *goal-directed* focused methods [Brown et al., 2019], i.e. methods that try to find the overall best molecule that optimizes a predefined value function.

In their seminal work Gómez-Bombarelli et al. proposed a variational autoencoder trained on SMILES representation for automated chemical design [Gómez-Bombarelli et al., 2018]. Pretrained on large molecular dataset, the resulting model can be used to encode molecules to and from a continuous latent representation. As discussed before, one of the major three challenges in automated denovo design is the navigation of the discrete chemical space [Schneider and Fechner, 2005]. With a model that can map the discrete chemical space to and from a continuous latent space, navigation is significantly simplified, as different molecules can be transformed into each other by simple vector operations in the latent space. In their work, Gómez-Bombarelli et al. utilized this property to apply Bayesian Optimization for a guided search to find points in the space corresponding to compounds with high drug-likeness and synthetic accessibility [Gómez-Bombarelli et al., 2018].

Publication 2: Efficient Multi-Objective Molecular Optimization in a Continuous Latent Space. In the second publication [Winter et al., 2019b], we build upon the continuous representation developed in Publication 1. In this work we tackle two issues not addressed in the related work. First, retraining a generative model or utilizing Bayesian Optimization in high dimensional space can quickly get computationally expensive. We propose the use of a simple yet efficient heuristic optimization method, namely Particle Swarm Optimization

[Kennedy and Eberhart, 1995], to navigate the continuous latent space proposed in Publication 1. Secondly, we demonstrate how our proposed method can be used to optimize multiple challenging objectives at the same time, as it is often required in a drug development process. This is in contrast to related works that mostly limit themselves to the optimization of single and more trivial properties such as the octanol-water partition coefficient (logp) or the quantitative estimate of drug-likeness score (QED). In fact, as our proposed optimization method is based on our proposed continuous molecular descriptor, which showed good performance in QSAR modelling, we were able to include trained QSAR models based on this representation as value function to guide the optimization. For example, we demonstrated how our proposed method is able to find molecules that are predicted to selectively bind to a protein target while having desirable ADME properties, i.e. solubility, metabolic stability and cell membrane permeability. We also demonstrate how our proposed method can be used to optimize a query molecule while keeping a substructure of it fixed and successfully benchmark it against state-of-the-art methods [Brown et al., 2019].

Publication 3: grünifai: Interactive Multiparameter Optimization of Molecules in a Continuous Vector Space. In the third publication [Winter et al., 2020], we developed a web application to interactively steer an *in silico* molecular optimization based on the methods developed in Publications 1 and 2. The web application enables the user to sketch an initial query molecule that is used as starting point for the optimization (e.g. a lead compound). Next, the user can select from a range of pretrained QSAR and rule-based models to define the objective function used for the optimization. Further customization is made possible by selecting desirable ranges for the different models and individual weights for the total score. The optimization can then be started by the user and further customized by rating intermediate results, making compounds that are structurally similar to those up-rated molecules more likely.

Overall, this application makes our developed methods more accessible to a broader range of potential users interested in *in silico* molecular optimization and further enables them with multiple ways to interact with and customize the optimization.

1.6 Representing Molecular Conformations

Up to this point, we only considered the molecular topology, i.e. the two dimensional structure of a molecule. However, naturally a molecule is a three dimensional entity and can exist in many different spatial arrangements. These different spatial arrangements of a molecule (with the same topology) are commonly referred to as *conformations* or geometries of a molecule. Arguably, a molecule’s geometry is one of its most relevant property as it determines its interaction with other molecules such as a protein-target it binds to. Still,

most QSAR models are based on 2D representations, as discussed above, and so-called 3D QSAR has not found much success in comparison [Cherkasov et al., 2014, Doweiko, 2004]. Probably the main reason for that is the fact that there exist an infinite amount of conformations for each molecule (although the amount of local minima is usually limited) and it is often not known which conformation corresponds to a (measured) property used as target in a QSAR model. Still, approaches like Comparative Molecular Field Analysis (CoMFA) showed success to a certain degree suggesting biological differences can be explained as effects of molecular field difference [Cherkasov et al., 2014, Cramer et al., 2008]. However, most of these methods require sampling and alignment of many conformations to make different molecules comparable which can get computational expensive quickly for larger or more flexible molecules. Hence, the question arises whether we can use methods discussed in section 1.4 to learn representations of conformations.

Probably the most common way to represent a molecular conformation is by assigning Cartesian coordinates to each atom in the molecule. However, this representation does not reflect crucial symmetries of molecules that should be accounted for in a representation learning model. Most notably, a molecule’s representation should be invariant to rigid translations and rotations, as those do not change the molecule itself. However, if e.g. an autoencoder is trained based on a Cartesian coordinated representation of conformations, which are not invariant under such transformations, the learned latent representations will not be invariant either. One way to address this problem is by choosing an input representation that respects the symmetries of interest, e.g. in our case the special euclidean group in three dimensions $SE(3)$ (compare e.g. [Kirillov Jr, 2008]). An example for such a representation are internal coordinates (see Figure 1.1e), where a molecular spatial arrangement is described by the distance between bonded atoms (bond length), the angle between two atoms connected by the same atom (bond angle) and the angle between the two intersecting planes defined by four connected atoms (dihedral or torsion angle). That way, the geometry is only defined by relative properties, hence, they do not change under distance-preserving transformations.

Publication 4: Auto-Encoding Molecular Conformations. In the fourth publication [Winter et al., 2021a], we propose an autoencoder model to transform molecular conformations expressed by internal coordinates to and from a fixed-sized continuous latent space. The main idea behind this work is to decouple the conformation and the topology of a molecule. In a separate GNN, we first extract node features for every atom in the molecule that are afterwards used as condition in encoding and decoding internal coordinates. For example, bond lengths are encoded conditioned on the node features of the bonded atoms, and similar for bond angles and dihedral angles. That way, we encode every internal coordinate in latent features that are pooled (averaged) in the final step of the encoder to construct a fixed-sized (i.e. independent of the size of the molecule) representation. The decoder uses this single representation

together with the node features extracted from the topology graph (as condition) to decode all individual internal coordinates, ultimately decoding the whole molecular conformation. To the best of our knowledge, this is the first method to encode conformations of molecules in a shared fixed-sized latent space.

As we decouple molecular topology from geometry, we demonstrate how different energetically reasonable conformations can be efficiently sampled by sampling different conformation embeddings for the same molecule topology (i.e. the node features used as condition in the decoder stay the same). Similar to the representation for molecular topologies proposed in Publication 1, this proposed method can be used to represent, compare, generate and interpolate different molecular conformations. Combining both methods, we demonstrate how a molecule and its conformation can be optimized for certain geometry-dependent properties, which could be used to find molecules in the chemical space that have desirable spatial properties.

1.7 Representing Molecular Graphs

As discussed in Section 1.3, graph neural networks achieved great success in learning to extract expressive representations from the molecular graph by means of supervised learning. Unsupervised learning for graph representations has mainly focused on *node-level* representation learning, which aims at encoding the local structure of nodes in a graph [Cao et al., 2016, Wang et al., 2016, Kipf and Welling, 2016a, Qiu et al., 2018, Pan et al., 2018]. Node-level graph representation are for example useful when predicting a property about single users in a social network or when predicting the existence of an edge between two nodes like it is done in recommender system models [Ying et al., 2018, Fan et al., 2019].

On the other hand, there are many interesting problems, such as molecular property prediction, that require a *graph-level* representation. However, unsupervised learning of graph-level representations has not received much attention. This can probably be explained by the high representation complexity of graphs arising from their inherent invariance to permutations of their nodes. Hence, following the line of argument in section 1.6, graph representations should respect the permutation group symmetry S_n .

In general, a graph G can be represented in a vector format by a node matrix $X \in \mathbb{R}^{N \times D}$ and a adjacency matrix $A \in \mathbb{R}^{N \times N}$, where N is the number of nodes in a graph and D is the number of node features (see Figure 1.1b). Note however, that a permutation of nodes in the graph will lead to a different graph representation $G(X', A')$. In fact, a graph with N nodes can be represented by $N!$ different node and adjacency matrices. Hence, training e.g. a regular autoencoder on such graph representations will lead to latent representations that are not permutation invariant, i.e. each permutation will lead to a different representation of the same graph. For example, the variational graph autoencoder originally proposed by Kipf and Welling [Kipf and Welling, 2016a] solves the reconstruction

task by encoding individual node representations that are used by the decoder to decide if two nodes are connected or not. A graph-level representation could be extracted by concatenation of all node-level representations. However, the resulting representation, would change (equivariantly) under permutation of nodes. Alternatively, nodes could be pooled (e.g. averaged) to extract a permutation invariant graph-level representation. However, without this node order information in the embedding, reconstruction of the input graph’s representation $G(X, A)$ in the right order is not possible and calculation of the order-dependent reconstruction loss used to train the autoencoder not feasible.

Another line of work concerns itself with so-called *contrastive learning* approaches for unsupervised graph-level representation learning. Narayanan et al. proposed a *skipgram* method on rooted subgraphs, similar to the popular *word2vec* method in Natural Language Processing [Narayanan et al., 2017]. Bai et al. proposed a *Siamese* network architecture, enforcing similar graphs (as measured by graph editing distance) to have similar representations (small Euclidean distance) [Bai et al., 2019]. Sun et al. proposed a method similar to *Deep InfoMax* [Hjelm et al., 2018] to extract graph-level representations [Sun et al., 2019].

These contrastive learning approaches belong to the so called *self-supervised* regime [Jaiswal et al., 2020]. Self-supervised learning still belongs to unsupervised learning as no labeled information is needed, however it provides a supervisory signal by defining a specific task that needs to be solved during training. This supervisory signal can be utilized to aggregate graph-level features in a permutation invariant way, similar to supervised learning. However, it has to be noted that contrastive learning methods cannot be used to reconstruct or generate novel graphs as only an encoding model is trained in this framework (in contrast to autoencoders).

Publication 5: Permutation-Invariant Variational Autoencoder for Graph-Level Representation Learning. In the fifth work [Winter et al., 2021b], we propose a graph autoencoder framework for permutation invariant graph-level representation learning. We address the ordering ambiguity in the reconstruction by training alongside the permutation invariant encoder and decoder a third model that predicts the permutation matrix to align encoded and decoded graph to the same node order.

Our proposed model can extract permutation-invariant representation of the whole graph that can also be used to generate novel graphs or to interpolate between two graphs. To the best of our knowledge, our proposed model is the first method that can extract permutation invariant graph-level representations that can also be used for graph generation. To demonstrate the effectiveness of our proposed model, we run a variety of graph reconstruction, generation and interpolation experiments and evaluate the expressive power of extracted representations for downstream graph-level classification and regression tasks. Notably, we demonstrate the model’s ability to extract expressive molecular descriptors from molecular graphs that outperform ECFP fingerprints.

We also show how the proposed method can be used to autoencode molecular conformations as discussed in Section 1.6. As the model is designed to encode and decode densely connected graphs (i.e. a graph where every node is connected to every other node), a conformation can be represented to the model by including the Euclidean distance between two (connected as well as unconnected) atoms as additional edge attribute. In our work, we demonstrate how such graphs can be encoded and decoded and used to sample molecular conformations. Notably, in contrast to Publication 4, the learned representations not only hold the conformational information but also the topology in the same fixed-sized continuous vector space. Hence, those representations can directly be used to optimize both over the topology as well as the conformational space.

1.8 Group Invariant Representation Learning

There has been a recent surge of interest in incorporating knowledge about underlying symmetries of a problem in neural networks as an inductive bias [Cohen and Welling, 2016, Bronstein et al., 2021]. For example Convolutional Neural Networks [LeCun et al., 1995] are by design equivariant with respect to translations of objects in an image and Message Passing Neural Networks [Gilmer et al., 2017] are equivariant to permutations of nodes in the graphs. Still, many properties, such as the class label of an image or the ground-state energy of a molecule, are inherently invariant to a certain symmetry [Miller et al., 2020]. Hence, it is common practice to predict such invariant properties by a model that consists of an equivariant neural network followed by a symmetric function (e.g. a pooling layer).

As discussed in Section 1.7, such models are usually trained in a supervised setting. Unsupervised learning of group invariant representations is challenging, as for instance, an autoencoder with a group-invariant bottleneck can only reconstruct its input up to a group transformation which makes the evaluation of the (transformation-dependent) reconstruction loss unfeasible.

In essence, Publication 5 proposes a method that disentangles the encoded representation of a graph into a permutation invariant part and a permutation equivariant group action (where the predicted permutation matrix is a linear representations of the action of symmetric group S_n). By combining both these representations during reconstruction, we were able to train a model that extracts all permutation-invariant information in one representation, while still using an autoencoder framework with an permutation-dependent reconstruction objective. In a followup work we generalize this concept to any group G .

In general, we can define a group G as a pair (G, \cdot) containing a set G and a binary operation $\cdot : G \times G \rightarrow G$ which is associative and for which there exists an identity element as well as inverse elements. Groups are a general framework that can be used to describe transformations that act on a space, or in particular to describe the symmetries of objects or sets represented in a

certain way. Prominent examples are the symmetric group S_n to e.g. describe permutations of sets or graphs as discussed above or the already mentioned special euclidean group $SE(n)$ that describes rigid rotations and translations in n-dimensional space.

Often, data is represented in ways that do not reflect inherent symmetries of properties of interest. As discussed in Section 1.7, representing graphs by node and adjacency matrices does not reflect the permutation-invariance of graph-level features. Similarly, the class-label of an image or the ground-state energy of a molecular conformation does not change upon translation or rotation of the image or conformation, while the data representation itself (i.e. pixels or Cartesian coordinates of images and atoms in conformations respectively) does.

In group theory, a map $f : X \rightarrow X$ is said to be *invariant* with respect to a *group-action* $g : G \times X \rightarrow X, (g, x) \mapsto g.x$ iff

$$\forall x \in X, \forall g \in G, \quad f(g.x) = f(x). \quad (1.1)$$

Group invariance is a special case of group *equivariance*:

$$\forall x \in X, \forall g \in G, \quad f(g.x) = g.f(x). \quad (1.2)$$

Publication 6: Unsupervised Learning of Group Invariant and Equivariant Representations. In the sixth work [Winter et al., 2022], we propose a general learning strategy based on an encoder-decoder framework to learn group invariant representations of any data type and group. The key idea is to jointly train a group invariant encoder, a decoder and a group equivariant model that predicts the group action to solve the reconstruction task. In this work, we characterize the necessary conditions of these three different parts of our proposed framework independently of specific groups or network architectures. We evaluate our proposed framework on a variety of groups and data types demonstrating its validity and flexibility.

Most notably, we demonstrate the use of our framework for autoencoding point clouds. As discussed in Section 1.6, molecular conformations can be naturally represented by assigning Cartesian coordinates to each atom in the molecule, which corresponds to a point cloud in three dimensions. However, training a classical autoencoder on such an input representation will lead to a latent code that does not respect crucial symmetries like translation- and rotation symmetry. Our proposed framework can be used to encode point clouds in a translation-, rotation- and permutation-invariant latent code by training along side an invariant encoder an additional model that predicts the translation-, rotation-, and permutation group action to align the input point cloud with the point cloud decoded from the invariant embedding. The proposed model is therefore able to extract representations of molecular conformations that respect translation-, rotation- and permutation symmetries directly from the Cartesian coordinates without e.g. first transforming to an internal coordinate system (as was done in Publication 4).

Chapter 2

Publications

2.1 Publication 1: Learning Continuous and Data-Driven Molecular Descriptors by Translating Equivalent Chemical Representations

Full Reference: *Winter, Robin, Montanari, Floriane, Noé, Frank, & Clevert, Djork-Arné (2019). Learning continuous and data-driven molecular descriptors by translating equivalent chemical representations. Chemical science, 10(6), 1692-1701.*

DOI: 10.1039/C8SC04175J

Licence: CC-BY

Journal/Conference: Chemical Science (Impact Factor: 9.8)

Source Code: <https://github.com/jrwnter/cddd>


Paper's main contributions:

- We propose a novel method for learning to extract expressive molecular descriptors from molecular line notations.
- We utilize methods originally developed for natural machine translation to translate between two different molecular line notations.
- We demonstrate the competitive performance of our extracted descriptors in extensive QSAR and virtual screening benchmarks.
- We discuss the application of the extracted continuous molecular descriptor for chemical space exploration and generation of novel molecules.

Author's contribution to the paper:

- Conceptualization of the original idea and its application to molecular representation learning.
- Development of the methodology and implementation.
- Design and evaluation of experiments, data curation and analysis.
- Preparation and creation of initial draft and visualizations.

Acknowledgement: Reproduced from *Chemical Science, 10(6), 1692-1701* with permission from the Royal Society of Chemistry.

Cite this: *Chem. Sci.*, 2019, 10, 1692 All publication charges for this article have been paid for by the Royal Society of Chemistry

Learning continuous and data-driven molecular descriptors by translating equivalent chemical representations†

Robin Winter, ^{*ab} Floriane Montanari, ^a Frank Noé ^b and Djork-Arné Clevert ^a

There has been a recent surge of interest in using machine learning across chemical space in order to predict properties of molecules or design molecules and materials with the desired properties. Most of this work relies on defining clever feature representations, in which the chemical graph structure is encoded in a uniform way such that predictions across chemical space can be made. In this work, we propose to exploit the powerful ability of deep neural networks to learn a feature representation from low-level encodings of a huge corpus of chemical structures. Our model borrows ideas from neural machine translation: it translates between two semantically equivalent but syntactically different representations of molecular structures, compressing the meaningful information both representations have in common in a low-dimensional representation vector. Once the model is trained, this representation can be extracted for any new molecule and utilized as a descriptor. In fair benchmarks with respect to various human-engineered molecular fingerprints and graph-convolution models, our method shows competitive performance in modelling quantitative structure–activity relationships in all analysed datasets. Additionally, we show that our descriptor significantly outperforms all baseline molecular fingerprints in two ligand-based virtual screening tasks. Overall, our descriptors show the most consistent performances in all experiments. The continuity of the descriptor space and the existence of the decoder that permits deducing a chemical structure from an embedding vector allow for exploration of the space and open up new opportunities for compound optimization and idea generation.

Received 19th September 2018

Accepted 17th November 2018

DOI: 10.1039/c8sc04175j

rsc.li/chemical-science

1 Introduction

Molecular descriptors play a crucial role in chemoinformatics, since they allow representing chemical information of actual molecules in a computer-interpretable vector of numbers.¹ While chemical information can be represented by experimental measurements such as physico-chemical property measurements,² a lot of work has been done to derive molecular descriptors from a symbolic representation of a molecule. A widely used concept to generate such theoretical molecular descriptors is *molecular fingerprints*. Molecular fingerprints encode structural or functional features of molecules in a bit string format and are commonly used for tasks like virtual screening, similarity searching and clustering.^{3–5} In particular, circular fingerprints like the *extended-connectivity fingerprints*

(ECFPs) were introduced to model quantitative structure–activity relationships (QSAR) for biological endpoints by way of classical machine learning approaches as well as for ligand-based virtual screening (VS).^{6,7}

Recent advances in the field of Deep Neural Networks (DNNs)^{8,9} also showed an impact in chemoinformatics-related tasks such as molecular property and activity prediction.^{10–12} The proposed DNNs have in common that they use pre-extracted molecular descriptors (mostly ECFPs) as input features. This, however, contradicts the fundamental idea of representation learning: DNNs should learn a suitable representation of the data from a simple but complete featurization, rather than relying on sophisticated human-engineered representations.^{9,13}

Following these considerations, work was also done to apply DNNs directly on supposedly more complete and lower-level representations of a molecule such as the molecular graph¹⁴ or the sequential SMILES (Simplified Molecular Input Line Entry Specification) representation.^{15–17} By training a DNN directly on a comprehensive and low-level representation, it can automatically learn to extract useful descriptors best suited for the specific task it is trained on, resulting in a specific descriptor set for a given dataset. The downside, however,

^aDepartment of Bioinformatics, Bayer AG, Berlin, Germany. E-mail: robin.winter@bayer.com

^bDepartment of Mathematics and Computer Science, Freie Universität Berlin, Berlin, Germany

† Electronic supplementary information (ESI) available: Detailed information regarding the final model architecture, hyperparameter grid, results and computation time. See DOI: 10.1039/c8sc04175j



directly follows from this property. Since features have to be learned from scratch for every new dataset, these methods are prone to overfitting if trained on limited data. This is an issue when it comes to bioactivity data, due to the relatively high cost of generating a data point.^{11,18,19}

Recently, work was also done to learn molecular descriptors in an unsupervised and data-driven way. Gómez-Bombarelli *et al.* proposed a *variational autoencoder*²⁰ to convert the discrete SMILES representation of a molecule to and from a multidimensional continuous representation.²¹ Although their main purpose was to build a framework for *de novo* molecular design, the authors showed that the resulting representations could also be used as descriptors for a downstream classification task. Xu *et al.* proposed a related unsupervised approach based on *sequence to sequence learning*.^{22,23}

Both studies use an *autoencoder*²⁴ methodology applied on the SMILES representation. An autoencoder comprises two neural networks, an encoder and a decoder. The encoder network transforms the input, here a SMILES sequence of variable length with discrete values, to a fixed size continuous representation (*latent representation*). The decoder network takes the latent representation as the input and aims at transforming it back to the input sequence. The whole autoencoder network is trained on minimizing the mean reconstruction error on a single-character level for each input sequence. By introducing an *information bottleneck* between the encoder and the decoder, the network is forced to compress the essential information of the input, so that the decoder still makes as few errors as possible in the reconstruction. If the trained autoencoder is able to encode all the necessary information of a given molecular representation to accurately reconstruct the original molecular representation, Xu *et al.* argue that it may also capture more general chemical information about the molecule and could be used as a molecular descriptor. However, training an autoencoder on reconstructing a sequence which represents a molecule bears the risk that the network solely focuses on syntactic features and repetitive patterns of this sequence, neglecting its semantics and failing to encode higher-level concepts such as molecular properties.

In this work, we want to address this issue by proposing a method that is based on a translation rather than a reconstruction methodology (see Fig. 1). Similar to a human translating a sentence from one language to another by first reading the whole sentence to get a general understanding before starting translation, a so-called Neural Machine Translation (NMT)²³ model first reads the whole input sequence and encodes it into an intermediate continuous vector representation (latent representation) which is then used by the decoder to emit a respective translation. This latent representation can be thought of as the model's "understanding" of the input sequence's "meaning", incorporating all the semantic information shared by the input and output sequences. Here, we want to exploit this translation methodology to extract the "meaning" of a molecular representation like an InChI (International Chemical Identifier)²⁵ by translating it to another syntactically different one, *e.g.* SMILES. Since the decoder uses the encoded latent representation to generate a semantically

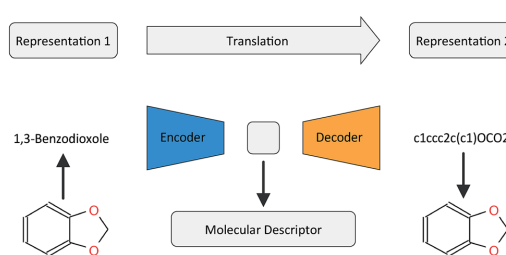


Fig. 1 General architecture of the translation model using the example of translating between the IUPAC and SMILES representations of 1,3-benzodioxole.

equivalent but syntactically different representation, the network does not benefit from encoding unnecessary information about the input sequence. However, the decoder can only succeed in generating the right translation for a given molecular representation if the encoder compresses a comprehensive description of the chemical structure in the latent representation.

By training the translation model in a data-driven way on a large set of chemical structures, we propose a model that can extract the information contained in a comprehensive but discrete and variable-sized molecular representation (*e.g.* SMILES) and transform it into a continuous and fixed-sized representation. Once trained, the resulting model can be used to extract meaningful molecular descriptors for query structures without the need for retraining or including labels. To analyse the quality of the resulting molecular descriptors, we perform a variety of experiments on predictive QSAR and virtual screening tasks. Finally, we show that it is possible to navigate smoothly in this new continuous chemical descriptor space by modifying slightly the molecular representation of an existing compound in a given direction and using the decoder to obtain new chemical structures.

2 Methods

2.1 Molecular representations

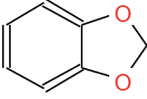
While translation could be performed between arbitrary molecular representations, in this work we focus on the sequence-based SMILES and InChI representations.

The InChI notation represents molecular structures as a sequence of characters divided into layers and sub-layers providing different types of information such as the chemical formula, bonds and charges.

The SMILES notation also represents molecular structures as a sequence of characters. In contrast to the InChI notation, however, a SMILES is not divided into different information layers but encodes the whole molecular structure in one sequence of characters including identifiers for atoms as well as identifiers denoting topological features like bonds, rings and branches. Since a molecule can typically be represented by many of equally valid SMILES, various algorithms have been developed to guarantee the uniqueness of a SMILES notation for



Table 1 Different sequence-based molecular representations for the example 1,3-benzodioxole

Graph	
IUPAC	1,3-Benzodioxole
SMILES	<chem>c1ccc2c(c1)OCO2</chem>
Canonical SMILES	<chem>c2ccc1OCOC1c2</chem>
InChI	InChI = 1S/C7H6O2/c1-2-4-7-6(3-1)8-5-9-7/ h1-4H,5H2

a molecule. In this work we use the library RDKit²⁶ to generate such canonical SMILES.

Table 1 visualizes how the different notations differ in their syntax while representing the same molecule. Although both SMILES and canonical SMILES share the same identifiers and general syntax, the two sequences, coming from different algorithms, are not identical. Hence, not only could translation be performed between InChI and canonical SMILES, but also between any SMILES representation of a molecule and its canonical version. We utilized the SMILES enumeration procedure proposed by E. Bjerrum to generate a random SMILES variant for a given molecule.¹⁶ In order to be invariant to the SMILES representation at inference time, we also used the canonical SMILES as the input half of the time.

In order to use the aforementioned sequence-based molecular representations as the input and output of the translation model, we tokenized the sequences and encoded them in a one-hot vector representation. By defining a lookup table T for the N tokens in sequence representations (e.g. $T_2 = C$, $T_5 = Br$), a one-hot representation of token T_i is defined by an N -dimensional vector with a one in the i -th entry and zeros elsewhere. We defined different lookup tables for both SMILES and InChI representations, mostly tokenizing the sequences on a character level except for "Cl", "Br" and "InChI = 1S/". We tokenized 38 and 28 unique characters for SMILES and InChI sequences, respectively.

2.2 Translation model

Fig. 1 depicts the general concept of the model for an example of translating from the IUPAC representation of a given molecule to its SMILES representation. For the encoder network, we tried both convolutional neural network (CNN) and recurrent neural network (RNN) architectures of different size and depth followed by a fully connected layer that maps the output of the CNN or the concatenated cell states of the RNN to the latent space, respectively (see the ESI[†] for an introduction to the basic concepts of these different neural network architectures). The decoder network consists of an RNN, whose cell states are initialized by an individual fully connected layer for each layer in the RNN, taking the latent space as the input.

To further encourage the model to learn a meaningful representation of a molecule, we extend the translation model by an additional classification model for certain molecular properties. Similar to the method proposed by Gómez-

Bombarelli *et al.*, this classification model takes the latent representation of the translation model as the input and predicts certain molecular properties which can be directly deduced from the molecular structure. We fixed the classification model as a 3-layer fully connected neural network, mapping the latent space to the molecular property vector.

The output of the decoder network's RNN is a sequence of probability distributions over the different possible characters defined in the respective lookup table. The complete model is trained on minimizing the cross-entropy between these probability distributions and the one-hot encoded correct characters in the target sequence as well as minimizing the mean squared error in predicting the molecular properties (classifier network). For the decoder RNN we utilized *teacher forcing*²⁷ during training and a left-to-right beam search²³ during inference.

We monitored the translation accuracy of the model on a single-character level, by comparing the correct character in the target sequence with the most probable character in the decoder RNN's output at each position. To select the best combination of translation task and architecture, we used the predictive performance of machine learning models built on two QSAR datasets using the respective latent representations as descriptors. This ensures that the translation model not only works well at translating (high single-character accuracy) but is also well suited to extract meaningful molecular descriptors from the input sequence (good performance of a simple QSAR model build on the embedding).

2.3 Datasets and preprocessing

The translation model was pretrained on a large dataset composed of molecular structures from the ZINC15 (ref. 28) and PubChem²⁹ databases. Both databases were merged, the duplicates removed and filtered with RDKit using the following criteria: only organic molecules, molecular weight between 12 and 600, more than 3 heavy atoms and a partition coefficient $\log P$ between -7 and 5 . Additionally, we removed the stereochemistry, stripped the salts and only kept the largest fragments. For each molecule, nine molecular properties were extracted: $\log P$, the maximal and minimal partial charge, the number of valence electrons, the number of hydrogen bond donors and acceptors, Balaban's J value,³⁰ the molar refractivity and the topological polar surface area. Molecules which could not be processed by RDKit were removed. After applying this preprocessing procedure the resulting dataset consisted of approximately 72 million compounds.

For the evaluation of the molecular descriptors extracted by the final translation model, we performed eight QSAR and two VS experiments. The QSAR datasets (see Table 2) were taken from various sources and were preprocessed in the same way as the pretraining dataset. Two of the datasets (Ames mutagenicity and lipophilicity) were used to validate the different translation models' architectures. The remaining eight datasets were solely used for evaluating the final model.

The VS experiments were performed on 40 targets of the Directory of Useful Decoys (DUD) and 17 targets of the Maximum Unbiased Validation (MUV) dataset.^{31,32}



Table 2 Ten different QSAR datasets used for benchmarking our molecular descriptor. The final number of compounds in each task after preprocessing is mentioned

Dataset	Acronym	Task	Split	Number of compounds	Reference
Ames mutagenicity	ames	Classification	Validation	6130	33
HERG inhibition	herg	Classification	Test	3440	34
Blood–brain barrier penetration	bbbp	Classification	Test	1879	35
β -Secretase 1 inhibition	bace	Classification	Test	1483	36
Toxicity in honeybees	beet	Classification	Test	188	37
Epidermal growth factor inhibition	egfr	Regression	Test	4451	38
Plasmodium falciparum inhibition	plasmo	Regression	Test	3999	39
Lipophilicity	lipo	Regression	Validation	3817	40
Aqueous solubility	esol	Regression	Test	1056	41
Melting point	melt	Regression	Test	184	42

All compounds of the evaluation datasets were removed from the pretraining dataset.

2.4 Evaluation and baseline

Our new molecular descriptors were benchmarked against state-of-the-art descriptors in QSAR and VS experiments.

For modelling structure–activity relationships, we compare three different approaches: classical machine learning models applied on our descriptors and on circular fingerprints of different radii and folding as implemented in RDKit (see details in the ESI†) as well as an end-to-end molecular graph convolution method as implemented in DeepChem.⁴³ The first two methods require selecting the learning algorithm to plug on top of the molecular representation. For this, we used Random Forest (RF),⁴⁴ support vector machine (SVM) with an RBF kernel⁴⁵ and Gradient Boosting (GB)⁴⁶ as implemented in scikit-learn.⁴⁷ A preliminary check on the two validation tasks showed that SVM was the method that worked best in combination with our descriptors and was therefore the only method applied to all other QSAR datasets for our descriptors. Our descriptors were standardized to zero mean and unit variance for each task individually. We performed an extensive hyperparameter optimization in a nested cross-validation (CV) fashion to select the best set of descriptor, model and hyperparameters for each task (see the ESI† for the detailed hyperparameter grid for each model).

The graph convolution models were trained directly on the different QSAR datasets. Hyperparameters such as learning rate and filter size were optimized in a cross-validation (see the ESI† for the detailed architecture and hyperparameter grid).

Each dataset was split in two different ways for the validation. The random CV corresponds to five random splits while the cluster CV corresponds to five clusters obtained by *K*-means clustering with *K* = 5 on MACCS fingerprints.⁴⁸

To select the best performing combinations, we specifically looked at the coefficient of determination (r^2) and the area under the receiver operating characteristic curve (ROC AUC) for the regression and classification tasks, respectively.

For the ligand-based virtual screening experiments, we followed the benchmarking protocol proposed by Riniker *et al.*⁴⁹ For each target in both VS databases, five active compounds were picked randomly and the remaining compounds were ranked according to their similarity to the active set as measured by

a similarity metric in the respective descriptor space. The process was repeated 50 times for each dataset, each time selecting a new random set of active and decoy compounds. We compared the performance of our descriptors with the 14 molecular fingerprints provided in the benchmark protocol (see the ESI†). The similarity in the discrete baseline fingerprint space was calculated using the Tanimoto similarity. For our continuous descriptors (per-target standardized to zero mean and unit variance) we used cosine similarity. The resulting ranking of the compounds is evaluated by calculating the mean ROC-AUC over the 50 repetitions for each target. Additionally, a Wilcoxon signed-rank test⁵⁰ is performed to analyse the statistical significance of the differences in the mean ranks of our descriptor to the baseline descriptors.

3 Results and discussion

Our translation model is a data-driven method for generating meaningful compound representations by forcing translation of all necessary information between two sequence-based representations of a molecule into a low dimensional continuous embedding (latent space). Since sequence-based representations of molecules such as SMILES or InChI are easily obtained from cheminformatics packages, the pretraining of the model can be performed on a vast chemical space (here, around 72 million compounds were used). Once the pretraining is finalized, the translation model can be used to encode compounds into the embedding or to decode embeddings into compounds. The obtained compound embedding can be utilized as a new continuous and reversible molecular descriptor that can be used to evaluate similarity in chemical space or train machine learning models to predict properties and biological activities.

3.1 Pretraining

We evaluate the different network architectures of the translation model (see Fig. 1) in terms of performance of the extracted descriptors for the two validation tasks. Fig. 2 shows both translation accuracy and predictive performance on the validation sets during the first 20 000 training steps. We show the best performing model for both translation tasks (SMILES to canonical SMILES and InChI to canonical SMILES) as well as the best model for the regular canonical SMILES autoencoding task.



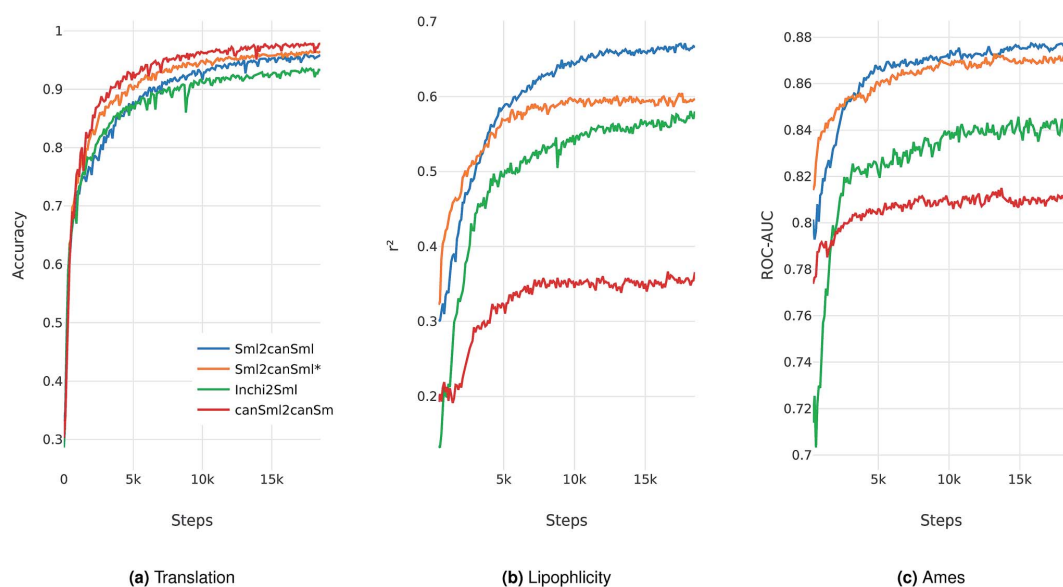


Fig. 2 Performance of the best model on four different translation tasks during the first 20 000 training steps. The Sml2canSml* run was trained without the additional classification task of molecular properties. (a) Translation accuracy. (b) Mean performance on the lipphicity regression task. (c) Mean performance on the Ames classification task. For (b) and (c), the translation model at the respective step was utilized to extract the molecular descriptors fed into an SVM to model both tasks.

Generally, as the models get better at translating the input to the output sequence, the predictive performance of an SVM based on the latent representation also improves. Since the translation model is trained on producing the correct translated sequence for a given input sequence, it is forced to store all important information necessary to do this translation in the bottleneck of the network: the latent representation (see Fig. 1). The more the information of a molecule encoded in the latent representation, the better it is suited as a molecular descriptor to predict certain properties of this molecule. Hence, the prediction performance on QSAR tasks increases.

The overall best performance was achieved with a translation model based on an RNN architecture for the encoder network that was trained on translating from a SMILES representation to its canonical version (see the ESI† for the detailed network architecture). The model based on the InChI to canonical SMILES translation is also able to accurately translate between the two representations. Its intermediate latent representation, however, is not as well suited for training an SVM on the validation task.

We also tried to train models on translating from canonical SMILES to InChI representations. These models, however, failed (in contrast to the opposite task) to learn anything. This is probably due to the higher complexity of the InChI format (including counting and arithmetic as already discussed by Gómez-Bombarelli *et al.*), making the generation of a correct InChI string for a given molecule a difficult task to learn.

In order to assess the impact of the additional classification task of molecular properties, Fig. 2 also shows the performance of the best model without this additional task during training.

Since this model solely focuses on translating, it reaches better translation accuracies faster. However, this difference seems to diminish as training time increases. The additional classification task seems to have a clear positive impact on the predictive performance of the lipphicity task, while resulting in a small improvement on the Ames mutagenicity task. The improvement on the lipphicity task is probably mainly due to its correlation with molecular properties (such as the partition coefficient $\log P$) that were included in the classification task.

All models based on translating between two different molecular representations show a clear improvement over models trained on reconstructing the same input sequence. Interestingly, translating between two molecular representations seems harder to learn than reconstructing the same input representation (see Fig. 2a). This can be explained by the fact that the translation models cannot simply store sequence-based features or patterns in the latent space, but have to learn to extract the information that both the input and output sequences have in common: the molecule they are both representing. These findings imply that, indeed, the translation task encourages the model to encode more relevant information of the molecule in the latent space, resulting in a potentially powerful molecular descriptor.

3.2 QSAR modelling

Next, we extracted molecular descriptors of the remaining (test) QSAR datasets (see Table 2) with the best performing translation model and benchmarked them as described in the Methods section. Fig. 3 shows the results of this evaluation for

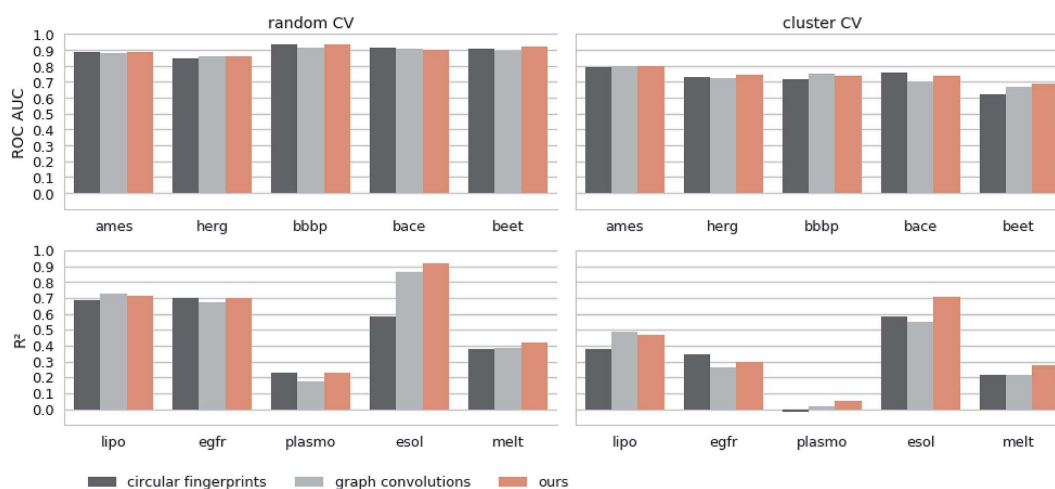


Fig. 3 Results of the 5 regression and 5 classification QSAR-tasks. Separate results are shown for both cross-validation on random splits (random CV) and cross-validation on cluster splits (cluster CV). We compare the results of an SVM trained on our descriptors with the best model (SVM, RF and GB) trained on the best performing circular fingerprint as well as an end-to-end trained graph-convolution model after extensive hyperparameter optimization.

random-split and cluster-split cross-validation respectively, comparing our molecular descriptor to the best model based on the different circular fingerprints and the graph-convolution networks trained end-to-end for each QSAR dataset individually (see the ESI† for detailed results).

Generally, the hyperparameter-optimized methods perform on a comparable level for most of the QSAR tasks, each method showing at least on one task a slightly better mean performance over the different splits.

The lipophilicity and aqueous solubility datasets show the largest variance in performance between the models. The graph-convolution method outperforms the models based on the baseline fingerprint in predicting these physico-chemical endpoints. For the solubility endpoint, however, this is only true in the case of random splits. In the case of cluster splits, the graph-convolution model apparently fails to generalize on the hold-out clusters. This is probably due to the relatively small size of the solubility dataset. Since the graph-convolution method is trained end-to-end, it has to learn to extract meaningful features for each dataset from scratch which could lead to overfitting, if training data are limited. In contrast, the baseline fingerprints and our descriptors are built upon predefined or pretrained feature extraction methods respectively, independently from the task at hand. Our proposed molecular descriptors show good performance in predicting physico-chemical endpoints (lipophilicity, solubility and melting point) even in the cluster cross-validations on the small datasets (solubility and melting point).

Summing up, our proposed molecular descriptors exhibit competitive or better performance than the best baseline models in all investigated QSAR tasks.

Additionally, we would like to emphasize that we fixed our feature extraction method based on two datasets (Ames and

lipophilicity on random splits) to avoid a model selection bias on the remaining test sets. This, however, was not done for the baseline methods. The fingerprint-based models could choose between nine different flavours of circular fingerprints and three different learning algorithms for each task respectively and due to the considerable training time the graph-convolution models were not trained in a nested cross-validation. Hence, it is remarkable that, although we applied a much harsher evaluation scheme on our method, it still achieved comparable – if not better – results to the baseline methods.

3.3 Virtual screening

The goal of ligand-based virtual screening (VS) is to rank a large set of compounds with respect to their activity on a certain target based on their similarity to some known active query compounds. It is based on the assumption that similar compounds will have a similar biological activity.

To investigate how well our descriptors are suited for ligand-based virtual screening, we followed the benchmark protocol of Riniker *et al.* to compare our extracted descriptors against other state-of-the-art molecular descriptors. In Table 3 the ranking performance of the descriptors is compared on the DUD and MUV databases respectively. On both databases our descriptor significantly outperformed the second best descriptor ($p < 0.05$). Thus, similarities measured between compounds in our proposed descriptor space are better correlated with their pharmacological similarity than similarities measured in the baseline fingerprint spaces.

Interestingly, the best baseline descriptor in the DUD screen (laval) is only fifth in the MUV screen. The best baseline descriptor in the MUV screen (ap) is not even represented in the



Table 3 Results of the VS-experiment on the DUD and MUV databases for the best 10 descriptors respectively. p -Values of the Wilcoxon signed-ranked test between our descriptor and the second best are given respectively

(a) DUD: $p = 5 \times 10^{-38}$										
Descriptor	ours	laval	tt	lecfp4	lecfp6	ecfp4	rdk5	avalon	ecfp6	fcfp4
ROC-AUC	0.949	0.899	0.890	0.887	0.886	0.884	0.884	0.881	0.881	0.874
(b) MUV: $p = 0.04$										
Descriptor	ours	ap	tt	avalon	laval	ecfc4	rdk5	ecfc0	fcfc4	fcfp4
ROC-AUC	0.679	0.677	0.670	0.644	0.643	0.637	0.627	0.626	0.615	0.605

top ten performing descriptors in the DUD screen. Our descriptor, however, shows robust performance over all analysed targets (see Fig. 4), even though the translation model was selected based on its performance on two QSAR validation tasks.

3.4 Exploring the continuous descriptor space

As opposed to the previously discussed baseline fingerprints, our proposed descriptor is continuous and the encoding into the descriptor space is reversible, due to the decoder part of our translation model. This opens new possibilities in terms of compound optimization and exploration of the chemical space. As already shown by Gómez-Bombarelli *et al.*, a continuous encoding of a molecular structure enables us to explore the neighbourhood of this molecule by decoding from points close to the query molecule's embedding.

In Fig. 5, we incrementally shift the embedding of a query molecule in two different directions and decode it back to a molecule. The directions we are shifting the molecule's embedding along are defined by the first and second principal component of the pretraining dataset (molecules from

PubChem and ZINC) in our descriptor space. We observe that the incremental shifts in the continuous descriptor space correspond to smooth transitions in the discrete chemical space. Apparently, the first principal component of our pretraining dataset correlates with the size of molecules: adding or subtracting a value along this axis corresponds to adding or removing atoms from the structure. Shifts along the second principal component of the pretraining dataset seem to be correlated with altering the molecule's polarity. To objectively analyse potential correlations between certain axes in the continuous descriptor space and molecular properties, we repeated the experiment with 1000 randomly picked compounds from the validation dataset and shifted each of them 10 steps in the negative and 10 steps in the positive direction along the two principle components, respectively. The mean Spearman correlation coefficient r between the compound's molar weight and the respective step along the first principle component was $r = 0.9470$ ($p = 0.00048$). The mean correlation between the compound's partition coefficient $\log P$ and the respective step along the second principle component was $r = -0.916$ ($p = 0.00015$). These results suggest a general

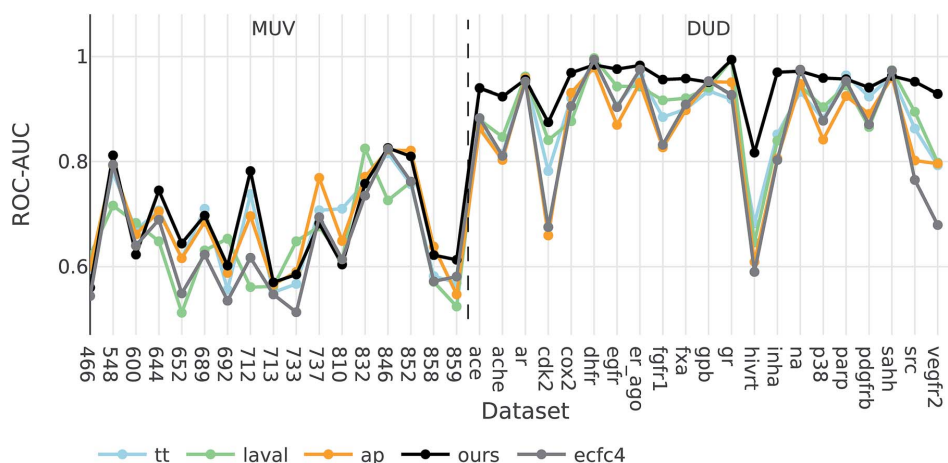


Fig. 4 ROC-AUC of the VS experiments for each target for the overall best descriptors as well as ecfc4 fingerprints.



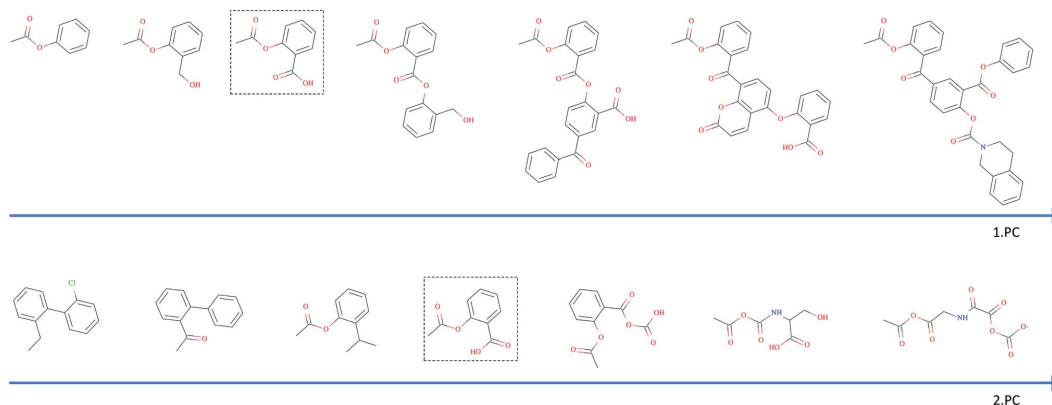


Fig. 5 Shifting of an example query molecule (here acetylsalicylic acid) along the first (top) and second (bottom) principal components of the pretraining dataset. The query molecule (dashed box) was encoded in our descriptor space, which was iteratively shifted – in both negative and positive directions – and decoded back to a SMILES.

Table 4 Aggregated results of the exploration of our descriptor space in 100 different random directions for 1000 different compounds in successive steps. For each step, $d_{\text{Euclidean}}$ is the mean Euclidean distance between the representations at this step and the representation of the respective starting compounds. d_{Tanimoto} is the mean Tanimoto distance between the ecfc4 fingerprints of the successfully decoded compounds and their starting compounds at each step. Rate_1 , rate_2 , and rate_3 describe the mean valid SMILES reconstruction rate, taking the first one, two and three most probable beam search outputs into account respectively

$d_{\text{Euclidean}}$	2.0	3.9	5.7	7.5	9.2	10.7	12.2	13.5	14.8	15.9	16.9	17.7
d_{Tanimoto}	0.02	0.02	0.05	0.10	0.19	0.31	0.46	0.60	0.71	0.78	0.83	0.86
Rate_1	1.00	1.00	1.00	1.00	0.99	0.99	0.98	0.98	0.98	0.97	0.97	0.97
Rate_2	1.00	1.00	1.00	1.00	1.00	1.00	0.99	0.99	0.99	0.99	0.99	0.99
Rate_3	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.99

correlation between shifts in certain directions in the descriptor space and certain molecular properties.

All analysed points along these two axes, when decoded, resulted in a valid SMILES (interpretable by RDKit). To further investigate how well our model is suited to explore around arbitrary molecule representations in arbitrary directions, we iteratively moved along 100 random directions for 1000 randomly picked compounds, respectively (see the ESI† for examples of generated compounds). Table 4 shows the aggregated results for this exploration. As expected, we observed a clear correlation between the (Euclidean) distance in our descriptor space and the (Tanimoto) distance in the circular fingerprint space. Thus, shifting the representation of a molecule in our descriptor space corresponds to gradual transitions in the chemical space. On average, even if shifted over long distances, our model succeeds in generating a high proportion of valid SMILES (>97%). If the most probable output of the model's beam search decoder results in an invalid SMILES, we observe that it is likely that one of the next most probable sequences results in a valid SMILES (>99%).

In a similar study Blaschke *et al.*, for example, analyzed 4 different autoencoder frameworks on the SMILES to SMILES reconstruction task and reported a valid SMILES proportion of only approximately 20% using their best model, if moved away

by a similar (Tanimoto) distance (note, however, that a direct comparison is problematic since Blaschke *et al.* sampled directly from the probability distribution of the last decoder layer and did not perform a beam search as we did).⁵¹ Another study by Segler *et al.* demonstrates that a simple RNN solely trained on generating SMILES sequences (no encoder/decoder framework) can obtain similar high valid SMILES ratios of 96% with random sampling.⁵²

4 Conclusion

We proposed a novel methodology that is able to learn to extract meaningful molecular descriptors, solely by an unsupervised training on a large dataset of molecular structures. We showed that the molecular descriptors extracted by our method significantly outperform state-of-the-art molecular fingerprints in ligand-based virtual screening (VS) experiments. Moreover, we show that machine learning models based on our descriptor perform similarly – if not better – on various quantitative structure–activity relationships (QSAR tasks), when compared to multiple state-of-the-art molecular fingerprints and computationally expensive graph-convolution models. Generally, our proposed descriptors show, compared to the baseline methods, consistent performance in all experiments, even across



different experimental concepts such as QSAR and VS. We believe that our method combines the advantages of both baseline models. Our method does not depend on fixed feature extraction rules but learns its own extraction method in a data-driven way. However, since it is pretrained on a large set of molecules, the resulting features generalize well and are less prone to overfitting.

As we focused in this work on translating between different string-based molecular representations, an evident follow-up would be the translation of conceptually different molecular representations such as the molecular graph or 3D-structure-based representations like the van der Waals and/or electro-negative potential surface.

Since our proposed molecular descriptors are continuous and can be translated back into a valid molecular structure, they open new possibilities in terms of compound optimization and navigation of the chemical space. We observe smooth and meaningful transitions in the chemical structure when a molecule's embedding is shifted in certain directions, where shifts along different axes in our descriptor space correspond to different structural and functional properties in the chemical space.

Moreover, Gómez-Bombarelli *et al.* already showed that their autoencoder framework could be utilized to automatically design molecules with respect to multiple properties such as synthetic accessibility and drug-likeness. Since our model's latent space was shown to be significantly better correlated with the molecule's biochemical properties, we think that our proposed translation method could significantly improve such a method's ability to generate and optimize molecules, also enabling optimization with respect to biological activity. These aspects will be explored and discussed in an upcoming study.

Availability

The source code and a pretrained model (to extract our proposed molecular descriptors out of the box) are available at <https://github.com/jrwnter/cddd>.

Conflicts of interest

There are no conflicts to declare.

Acknowledgements

This project was supported by several research grant. FN acknowledges funding from the European Commission (ERC CoG 772230 "ScaleCell") and MATH+ (AA1-6). DC and FM acknowledge funding from the Bayer AG Life Science Collaboration ("DeepMinDS"). RW acknowledges Bayer AG's PhD scholarship.

Notes and references

- 1 R. Todeschini and V. Consonni, *Handbook of molecular descriptors*, John Wiley & Sons, 2008, vol. 11.

- 2 C. Hansch, P. P. Maloney, T. Fujita and R. M. Muir, *Nature*, 1962, **194**, 178.
- 3 P. Willett, J. M. Barnard and G. M. Downs, *J. Chem. Inf. Comput. Sci.*, 1998, **38**, 983–996.
- 4 A. Cereto-Massagué, M. J. Ojeda, C. Valls, M. Mulero, S. Garcia-Vallvé and G. Pujadas, *Methods*, 2015, **71**, 58–63.
- 5 M. J. McGregor and P. V. Pallai, *J. Chem. Inf. Comput. Sci.*, 1997, **37**, 443–448.
- 6 H. Li, C. Yap, C. Ung, Y. Xue, Z. Li, L. Han, H. Lin and Y. Z. Chen, *J. Pharm. Sci.*, 2007, **96**, 2838–2860.
- 7 G. Hu, G. Kuang, W. Xiao, W. Li, G. Liu and Y. Tang, *J. Chem. Inf. Model.*, 2012, **52**, 1103–1113.
- 8 J. Schmidhuber, *Neural Networks*, 2015, **61**, 85–117.
- 9 Y. LeCun, Y. Bengio and G. Hinton, *Nature*, 2015, **521**, 436.
- 10 H. Chen, O. Engkvist, Y. Wang, M. Olivecrona and T. Blaschke, *Drug Discovery Today*, 2018, **23**(6), 1241–1250.
- 11 E. B. Lenselink, N. Dijke, B. Bongers, G. Papadatos, H. W. Vlijmen, W. Kowalczyk, A. P. IJzerman and G. J. Westen, *J. Cheminf.*, 2017, **9**, 45.
- 12 A. Mayr, G. Klambauer, T. Unterthiner and S. Hochreiter, *Front. Environ. Sci.*, 2016, **3**, 80.
- 13 M. D. Zeiler and R. Fergus, *European conference on computer vision*, 2014, pp. 818–833.
- 14 D. K. Duvenaud, D. Maclaurin, J. Iparraguirre, R. Bombarelli, T. Hirzel, A. Aspuru-Guzik and R. P. Adams, *Advances in neural information processing systems*, 2015, pp. 2224–2232.
- 15 D. Weininger, *J. Chem. Inf. Comput. Sci.*, 1988, **28**, 31–36.
- 16 E. J. Bjerrum, arXiv preprint arXiv:1703.07076, 2017.
- 17 S. Jastrzebski, D. Lesniak and W. M. Czarnecki, arXiv preprint arXiv:1602.06289, 2016.
- 18 G. Papadatos, A. Gaulton, A. Hersey and J. P. Overington, *J. Comput.-Aided Mol. Des.*, 2015, **29**, 885–896.
- 19 C. Kramer, T. Kalliokoski, P. Gedeck and A. Vulpetti, *J. Med. Chem.*, 2012, **55**, 5165–5173.
- 20 D. P. Kingma and M. Welling, arXiv preprint arXiv:1312.6114, 2013.
- 21 R. Gómez-Bombarelli, J. N. Wei, D. Duvenaud, J. M. Hernández-Lobato, B. Sánchez-Lengeling, D. Sheberla, J. Aguilera-Iparraguirre, T. D. Hirzel, R. P. Adams and A. Aspuru-Guzik, *ACS Cent. Sci.*, 2016, **4**(2), 268–276.
- 22 Z. Xu, S. Wang, F. Zhu and J. Huang, *Proceedings of the 8th ACM International Conference on Bioinformatics, Computational Biology, and Health Informatics*, 2017, pp. 285–294.
- 23 I. Sutskever, O. Vinyals and Q. V. Le, *Advances in neural information processing systems*, 2014, pp. 3104–3112.
- 24 D. E. Rumelhart, G. E. Hinton and R. J. Williams, *Learning internal representations by error propagation*, California univ san diego la jolla inst for cognitive science technical report, 1985.
- 25 S. Heller, A. McNaught, S. Stein, D. Tchekhovskoi and I. Pletnev, *J. Cheminf.*, 2013, **5**, 7.
- 26 *RDKit, Open-source cheminformatics*, <http://www.rdkit.org>, online, accessed 11-April-2013.
- 27 R. J. Williams and D. Zipser, *Neural Comput.*, 1989, **1**, 270–280.



- 28 J. J. Irwin and B. K. Shoichet, *J. Chem. Inf. Model.*, 2005, **45**, 177–182.
- 29 S. Kim, P. A. Thiessen, E. E. Bolton, J. Chen, G. Fu, A. Gindulyte, L. Han, J. He, S. He and B. A. Shoemaker, *Nucleic Acids Res.*, 2015, **44**, D1202–D1213.
- 30 A. T. Balaban, *Chem. Phys. Lett.*, 1982, **89**, 399–404.
- 31 N. Huang, B. K. Shoichet and J. J. Irwin, *J. Med. Chem.*, 2006, **49**, 6789–6801.
- 32 S. G. Rohrer and K. Baumann, *J. Chem. Inf. Model.*, 2009, **49**, 169–184.
- 33 K. Hansen, S. Mika, T. Schroeter, A. Sutter, A. Ter Laak, T. Steger-Hartmann, N. Heinrich and K.-R. Müller, *J. Chem. Inf. Model.*, 2009, **49**, 2077–2081.
- 34 P. Czodrowski, *J. Chem. Inf. Model.*, 2013, **53**, 2240–2251.
- 35 I. F. Martins, A. L. Teixeira, L. Pinheiro and A. O. Falcao, *J. Chem. Inf. Model.*, 2012, **52**, 1686–1697.
- 36 G. Subramanian, B. Ramsundar, V. Pande and R. A. Denny, *J. Chem. Inf. Model.*, 2016, **56**, 1936–1949.
- 37 K. Venko, V. Drgan and M. Novič, *SAR QSAR Environ. Res.*, 2018, **29**, 743–754.
- 38 A. P. Bento, A. Gaulton, A. Hersey, L. J. Bellis, J. Chambers, M. Davies, F. A. Krüger, Y. Light, L. Mak and S. McGlinchey, *Nucleic Acids Res.*, 2014, **42**, D1083–D1090.
- 39 D. Plouffe, A. Brinker, C. McNamara, K. Henson, N. Kato, K. Kuhen, A. Nagle, F. Adrián, J. T. Matzen and P. Anderson, *Proc. Natl. Acad. Sci. U. S. A.*, 2008, **105**, 9059–9064.
- 40 Z. Wu, B. Ramsundar, E. N. Feinberg, J. Gomes, C. Geniesse, A. S. Pappu, K. Leswing and V. Pande, *Chem. Sci.*, 2018, **9**, 513–530.
- 41 J. S. Delaney, *J. Chem. Inf. Comput. Sci.*, 2004, **44**, 1000–1005.
- 42 C. A. Bergström, U. Norinder, K. Luthman and P. Artursson, *J. Chem. Inf. Comput. Sci.*, 2003, **43**, 1177–1185.
- 43 *DeepChem*, <https://github.com/deepchem/deepchem>, accessed, 2018 02 03.
- 44 L. Breiman, *Mach. Learn.*, 2001, **45**, 5–32.
- 45 C. Cortes and V. Vapnik, *Mach. Learn.*, 1995, **20**, 273–297.
- 46 J. H. Friedman, *Ann. Stat.*, 2001, 1189–1232.
- 47 F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot and E. Duchesnay, *J. Mach. Learn. Res.*, 2011, **12**, 2825–2830.
- 48 P. Gedeck, B. Rohde and C. Bartels, *J. Chem. Inf. Model.*, 2006, **46**, 1924–1936.
- 49 S. Riniker and G. A. Landrum, *J. Cheminf.*, 2013, **5**, 26.
- 50 F. Wilcoxon, *Biom. Bull.*, 1945, **1**, 80–83.
- 51 T. Blaschke, M. Olivecrona, O. Engkvist, J. Bajorath and H. Chen, *Mol. Inf.*, 2018, **37**, 1700123.
- 52 M. H. Segler, T. Kogej, C. Tyrchan and M. P. Waller, *ACS Cent. Sci.*, 2017, **4**(1), 120–131.



Supporting Information for Learning Continuous and Data-Driven Molecular Descriptors by Translating Equivalent Chemical Representations

Robin Winter,^{*,†,‡} Floriane Montanari,[†] Frank Noé,[‡] and Djork-Arné Clevert^{*,†}

[†]*Department of Bioinformatics, Bayer AG, 13353 Berlin, Germany*

[‡]*Department of Mathematics and Computer Science, Freie Universität Berlin, 14195
Berlin, Germany*

E-mail: robin.winter@bayer.com; djork-arne.clevert@bayer.com

Common neural network architectures

In the following we introduce the basic concepts of the different neural network architectures used in our translation model

Fully-connected Neural Network

The most basic form of a Deep Neural Network is the Fully-connected Neural Network (FNN). In an FNN, each neuron in a layer of the network is connected to each neuron in the previous layer (see Figure 1 a). Thus, the output of a neuron in a fully-connected layer is a linear combination of all outputs of the previous layer, usually followed by a non-linear function.

Convolutional Neural Network

A convolutional neural network (CNN) builds up on multiple small kernels which are convolved with the outputs of the previous layer. In contrast to a neuron in a fully-connected layer, each output of a convolutional-layer is only a linear combination of neighboring outputs in the previous layer (see Figure 1 b). The number of considered neighbors is defined by the size of the kernel. Since the same kernels are applied along all outputs of the previous layer, a convolutional architecture is especially well suited for recognition of patterns in the input, independently of their exact position. Hence, CNNs are popular in image analysis tasks but were also successfully applied on sequence-based data.^{1,2}

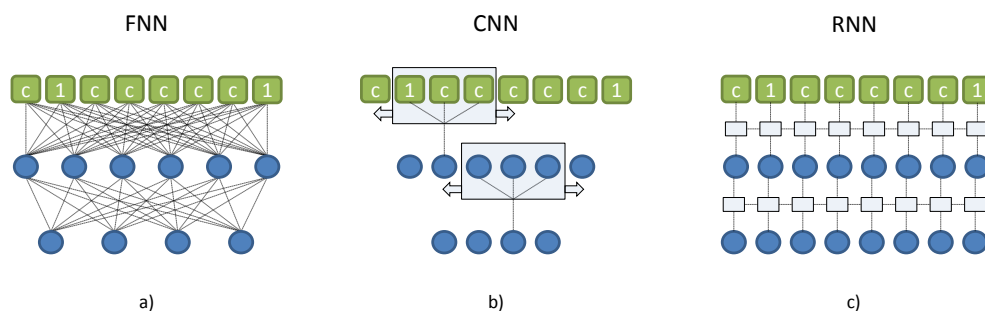


Figure 1: Three different 2-layer neural network architectures with SMILES representation of benzene as input. a) Fully-connected Neural Network (FNN): Each neuron is connected with each neuron/input of the previous layer. b) Convolutional Neural Network (CNN): The Input to a certain layer is convolved with a kernel of size 3. c) Recurrent Neural Network (RNN): Illustration of a RNN with an unrolled graph. Note that a 2-layer of RNN has two separate cell states (blue boxes)

Recurrent Neural Network

A more tailored architecture for sequence-based data is the Recurrent Neural Network (RNN). In contrast to FNN and CNN where all information flows in one direction, from the input layer through the hidden layers to the output layer (feed-forward neural networks), an RNN

has an additional feedback loop (recurrence) through an internal memory state. An RNN processes a sequence step by step while updating its memory state concurrently. Hence, the activation of a neuron at step t is not only dependent on the input at t but also on the state at position $t - 1$. By including a memory cell, an RNN is in theory able to model long-term dependencies in sequential data, such as keeping track of opening and closing brackets in the SMILES syntax. The concept of a neural network with a feedback loop can be simplified by unrolling the RNN network over the whole input sequence (see Figure 1 c). Unrolling the graph emphasizes that an RNN which takes long sequences as input becomes a very deep neural network suffering from problems such as vanishing or exploding gradients³. To avoid this problem, Hochreiter et al. proposed the *Long short-term memory* (LSTM) network, which extends the RNN architecture by an *input gate*, an *output gate* and a *forget gate*.⁴ In this work we use a modified version of the LSTM: the *gated recurrent unit* (GRU).⁵

Baseline Molecular Descriptors

In this section we will introduce the basic concepts of the different molecular descriptors we used as baseline.

Circular fingerprints like the extended-connectivity fingerprints (ECFPs) were introduced for the purpose of building machine learning models for quantitative structure-activity relationships (QSAR) models. This class of fingerprints iterates over the non hydrogen atoms of a molecular graph and encodes the neighbourhood up to a given radius using a linear hash function. The resulting set of neighbourhood hash codes for a dataset can then be handled as a sparse matrix or folded to a much smaller size (1024 or 2048 are typical folding size choices). Folding such a potentially large bit space (2^{32}) to a much smaller space produces collisions in the final fingerprint vector, where two different neighbourhood codes could end up at the same position in the folded vector, resulting in a loss of information. Additionally, once folded, it is impossible to trace back important bits to actual compound substructures,

and therefore the model interpretability is lost.

In the ligand-based virtual screening experiments we followed the benchmarking protocol proposed by Riniker et al.⁶ In this protocol we benchmarked our proposed descriptor against the 14 fingerprints available in the pipeline. Accordingly Riniker et al., these fingerprints can be divided into four different classes: dictionary-based (e.g. Molecular ACCess System MACCS), topological or path-based (e.g. atom pair (AP) fingerprints or topological torsions (TT)), circular fingerprints (e.g. ECFP) and pharmacophores (eg. FCFP). For a more comprehensive description of the different baseline fingerprints we refer to the work of Riniker et al..

QSAR modelling

We used the Python library RDKit (v.2017.09.2.0) to calculate Morgan fingerprints with radius 1, 2 and 3 (equivalent to ecfc2, ecfc4, ecfc6) each folded to 512, 1024 and 2048 bits, resulting in nine different baseline molecular descriptors.

Three different machine learning algorithms were used to model the QSAR tasks: Random Forest (RF), Support Vector Machine (SVM) and Gradient Boosting (GB). We utilized the Python library sklearn (v.0.19.1) to build, train and cross-validate the models. The hyperparameter optimization was performed for each QSAR task, model and fingerprint triplet individually. The hyperparameter grid for the different models was defined as follows:

- Random Forest:
 - Number of estimators (trees): 20, 100 and 200
- Support Vector Machine:
 - Penalty parameter C of the error term: 0.1, 0.5, 1, 3, 5, 10, 30 and 50
 - Coefficient gamma for the RBF-kernel: $1/(N_f + b)$, where N_f is the number of features and b is picked from $-300, -150, -50, -15, 0, 15, 50, 150$ and 300

- Gradient Boosting:
 - Number of estimators: 20, 100 and 200
 - Learning rate: 0.5, 0.1 and 0.01

In addition to the classical machine learning methods trained on circular fingerprints we also trained graph-convolution models on the different QSAR tasks. Briefly, the architecture consists of two successive graph convolution layers, an input atom vector of size 75 and rectified linear units (ReLU) were used as non linearity. For each task a individual graph-convolution model were trained and optimized with respect to following hyperparameters:

- batch size: 64, 128 and 256
- learning rate: 0.0001, 0.0005, 0.001 and 0.005
- convolutional filters: 64, 128 and 256
- dimension of the dense layer: 128, 256, 512, 1024, and 2048
- number of epochs: 40, 60 and 100

Translation Model Architecture

As final translation model we selected the model with the best performance on the validation QSAR tasks (see Figure 2). For the encoding part we stacked 3 GRU cells with 512, 1024 and 2048 units respectively. The state of each GRU cell is concatenated and fed into a fully-connected layer with 512 neurons and hyperbolic tangent activation function. The output of this layer (values between -1 and 1) is the latent space which can be used as molecular descriptor in the inference time. The decoding part takes this latent space as input and feeds it into a fully-connected layer with $512 + 1024 + 2048 = 3584$ neurons. This output is split into 3 parts and used to initialize 3 stacked GRU cells with 512, 1024 and 2048 units respectively. The output of the GRU cells is mapped to predicted probabilities for the

different tokens via a fully-connected layer.

The classifier network consists of a stack of three fully-connected layers with 512, 128 and 9 neurons respectively, mapping the latent space to the molecular property vector. The model was trained on translating between SMILES and canonical SMILES representations. Both sequences were tokenized as described in the method section and fed into the network.

In order to make the model more robust to unseen data, input dropout was applied on a character level (15%) and noise sampled from a zero-centered normal distribution with a standard derivation of 0.05. We used an Adam optimizer⁷ with a learning rate of $5 * 10^{-4}$ which was decreased by a factor of 0.9 every 50000 steps. The batch size was set to 64. To handle input sequences of different length we used a so-called bucketing approach. This means that we sort the sequences by their length in different buckets (in our case 10) and only feed sequences from the same bucket in each step. All sequences were padded to longest sequence in each bucket. We used the framework TensorFlow 1.4.1⁸ to build and execute our proposed model.

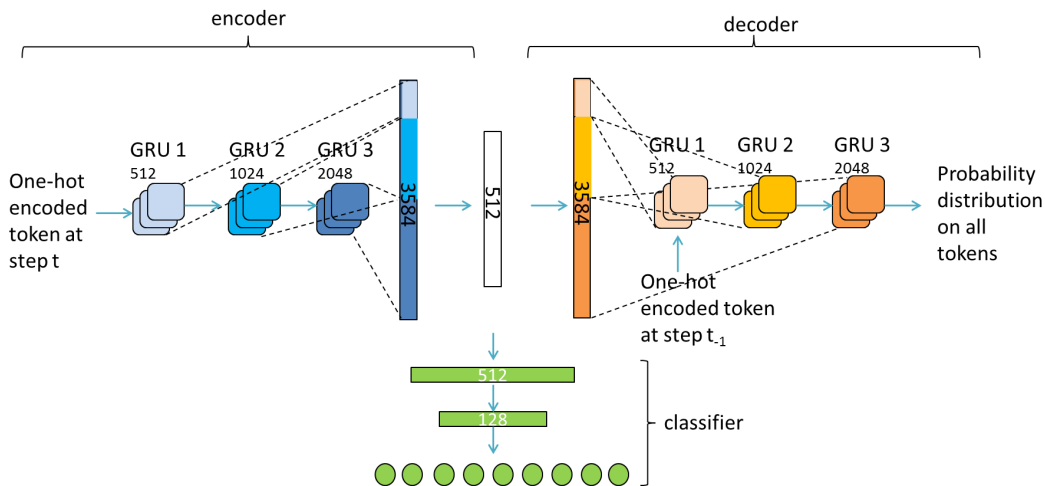


Figure 2: Final model architecture.

QSAR results

Tables 1, 2, 3 and 4 show the detailed results for the hyperparameter optimized models. For the baseline fingerprint models, only the result for the best performing fingerprint is shown. For each task we show the performance of an SVM on our descriptor (ours), the performance of a graph-convolution model (GC) and the best performing model-fingerprint combination (RF, SVM or GB). For the classification tasks we measure the area under the Receiver Operating Characteristic curve (roc-auc), the area under the precision-recall curve (pr-auc) accuracy (acc), F1-measure (f1). For the regression tasks we measure the coefficient of determination (r^2), Spearman’s rank correlation coefficient (r), mean squared error (mse) and mean absolute error (mae).

Table 1: Averaged results for the four classification QSAR task in random-split cross-validation after hyperparameter optimization.

Task	roc-auc	pr-auc	acc	f1	Descriptor	Model
ames	0.89	0.91	0.81	0.83	ecfc2 1024	RF
	0.88	0.90	0.81	0.83	-	GC
	0.89	0.91	0.82	0.83	ours	SVM
herg	0.85	0.94	0.82	0.89	ecfc4 1024	RF
	0.86	0.94	0.83	0.89	-	GC
	0.86	0.94	0.82	0.88	ours	SVM
bbbp	0.93	0.97	0.90	0.94	ecfc2 512	RF
	0.92	0.97	0.88	0.92	-	GC
	0.93	0.97	0.90	0.94	ours	SVM
bace	0.91	0.89	0.84	0.82	ecfc2 512	RF
	0.91	0.88	0.82	0.81	-	GC
	0.90	0.86	0.84	0.83	ours	SVM
beetox	0.91	0.88	0.89	0.69	ecfc6 2048	RF
	0.89	0.79	0.88	0.69	-	GC
	0.92	0.83	0.92	0.80	ours	SVM

VS Results

Tables 5 and 6 show the detailed results for the virtual screening experiments performed on the DUD and MUV databases. Our descriptor (ours) is compared to different baseline

Table 2: Averaged results for the four regression QSAR task in random-split cross-validation after hyperparameter optimization.

Task	r^2	r	mse	mae	Descriptor	Model
lipo	0.69	0.83	0.42	0.46	ecfc2 1024	SVM
	0.73	0.84	0.36	0.44	-	GC
	0.72	0.83	0.38	0.46	ours	SVM
egfr	0.70	0.84	0.62	0.57	ecfc4 2048	RF
	0.67	0.83	0.68	0.60	-	GC
	0.70	0.85	0.62	0.57	ours	SVM
plasmo	0.23	0.45	0.25	0.38	ecfc2 2048	SVM
	0.18	0.41	0.27	0.40	-	GC
	0.23	0.45	0.25	0.38	ours	SVM
esol	0.58	0.82	1.38	0.91	ecfc6 1024	SVM
	0.86	0.92	0.58	0.56	-	GC
	0.92	0.96	0.34	0.42	ours	SVM
melt	0.38	0.62	1700	33	ecfc2 2048	SVM
	0.39	0.67	1700	34	-	GC
	0.42	0.64	1600	32	ours	SVM

Table 3: Averaged results for the four classification QSAR task in cluster-split cross-validation after hyperparameter optimization.

Task	roc-auc	pr-auc	acc	f1	Descriptor	Model
ames	0.79	0.81	0.74	0.70	ecfc4 2048	SVM
	0.80	0.80	0.74	0.74	-	GC
	0.80	0.82	0.74	0.71	ours	SVM
herg	0.73	0.89	0.76	0.85	ecfc4 2048	RF
	0.72	0.89	0.77	0.86	-	GC
	0.75	0.90	0.76	0.84	ours	SVM
bbbp	0.72	0.88	0.79	0.84	ecfc4 2048	RF
	0.75	0.90	0.77	0.83	-	GC
	0.74	0.88	0.81	0.86	ours	SVM
bace	0.76	0.70	0.67	0.59	ecfc2 2048	GB
	0.70	0.67	0.55	0.52	-	GC
	0.74	0.67	0.65	0.55	ours	SVM
beetox	0.62	0.39	0.75	0.00	ecfc6 512	GB
	0.67	0.48	0.72	0.20	-	GC
	0.69	0.45	0.78	0.21	ours	SVM

Table 4: Averaged results for the four regression QSAR task in cluster-split cross-validation after hyperparameter optimization.

Task	r^2	r	mse	mae	Descriptor	Model
lipo	0.38	0.70	0.84	0.68	ecfc4 2048	RF
	0.49	0.69	0.68	0.64	-	GC
	0.47	0.69	0.71	0.65	ours	SVM
egfr	0.34	0.58	1.01	0.77	ecfc6 1024	RF
	0.26	0.51	1.16	0.86	-	GC
	0.30	0.55	1.09	0.81	ours	SVM
plasmo	-0.02	0.21	0.33	0.44	ecfc6 1024	RF
	0.02	0.20	0.32	0.44	-	GC
	0.05	0.24	0.31	0.43	ours	SVM
esol	0.58	0.82	1.38	0.91	ecfc6 2048	SVM
	0.55	0.76	1.62	0.96	-	GC
	0.70	0.89	1.01	0.75	ours	SVM
melt	0.22	0.51	1800	34	ecfc2 1024	SVM
	0.21	0.48	1930	34	-	GC
	0.28	0.55	1700	32	ours	SVM

descriptors by the are under the Receiver Operating Characteristic curve (ROC-AUC), the Enrichment Factor (EF) for the top 5% ranked compounds 5%, the Robust Initial Enhancement (RIE) with parameter $\alpha = 20$ and the Boltzmann-Enhanced Discrimination of ROC (BEDROC) with parameter $\alpha = 20$. For detailed description of the baseline descriptors and evaluation metrics, we refer the reader to the work of Riniker et al.⁶

Table 5: Results of the ligand-based virtual screen of the DUD database.

Descriptor	ROC-AUC	BEDROC	RIE	EF5
ours	0.949	0.792	12.485	15.294
laval	0.899	0.746	11.750	14.216
tt	0.890	0.744	11.718	14.257
lecfp4	0.887	0.771	12.138	14.813
lecfp6	0.886	0.767	12.072	14.691
ecfp4	0.884	0.764	12.024	14.623
rdk5	0.884	0.747	11.761	14.291
avalon	0.881	0.733	11.537	13.922
ecfp6	0.881	0.755	11.879	14.429
fcfp4	0.874	0.754	11.868	14.472
ap	0.868	0.717	11.298	13.676
ecfc4	0.867	0.749	11.782	14.354
maccs	0.863	0.667	10.511	12.730
fcfc4	0.852	0.728	11.459	13.891
ecfc0	0.805	0.570	8.969	10.634

Table 6: Results of the ligand-based virtual screen of the MUV database.

Descriptor	ROC-AUC	BEDROC20	RIE20	EF5
ours	0.679	0.195	3.826	4.258
ap	0.677	0.176	3.440	3.737
tt	0.670	0.191	3.746	4.066
avalon	0.644	0.174	3.403	3.661
laval	0.643	0.172	3.375	3.659
ecfc4	0.637	0.181	3.540	3.895
rdk5	0.627	0.177	3.473	3.747
ecfc0	0.626	0.130	2.541	2.816
fcfc4	0.615	0.163	3.184	3.471
fcfp4	0.605	0.175	3.420	3.706
lecfp4	0.601	0.178	3.480	3.774
lecfp6	0.599	0.178	3.481	3.768
ecfp4	0.599	0.176	3.447	3.753
ecfp6	0.591	0.175	3.433	3.776
maccs	0.578	0.127	2.482	2.705

Timing

Another point to consider, when comparing our proposed method with the baseline, is the time needed to extract the molecular descriptors and train the machine learning algorithm. Calculating Morgan fingerprints with RDKit takes on a single CPU only a few seconds for a dataset of 10.000 compounds. If ran on a modern GPU, our proposed model takes approximately the same time. On a single CPU core, however, this computational time increases approximately by a factor of 100. Training an RF for this size of dataset on our descriptors is in the order of seconds. An SVM, that scales with number of input features, takes, in the order of minutes for the best baseline fingerprints (ecfpc4 2048), and in the order of seconds for our descriptors (512 dimensional). The best performing graph-convolution model, on the other side, takes approximately 30 minutes to train on a modern GPU (on a single CPU this method would not be computational feasible). Thus, with a GPU at hand, running a QSAR experiment with our proposed descriptors is on the same timescale as with state-of-the-art molecular fingerprints, while being significantly faster than training a graph-convolution model. One way to make our encoder faster would be to replace the GRU cells by one-dimensional convolutional layers. In our experiments however these do not perform as well in the downstream QSAR validation sets.

Generated compounds

Figure 3 and 4 depicts examples of compounds generated for the experiment in section 3.4 in the main article. The compounds shown are randomly picked and had a (ecfp4-) tanimoto distance greater than 0.5 to the starting compound.

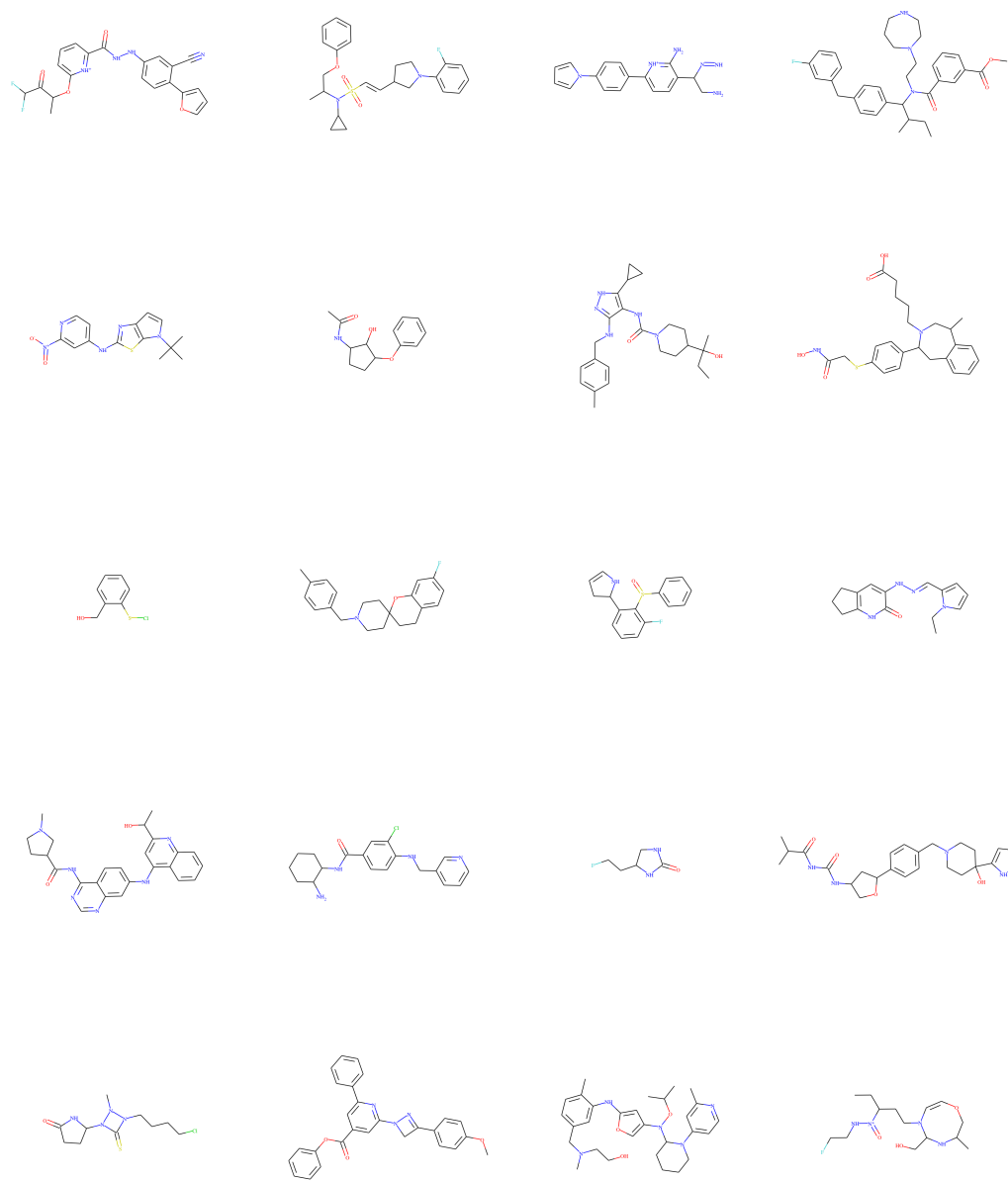


Figure 3: Examples of generated compounds.

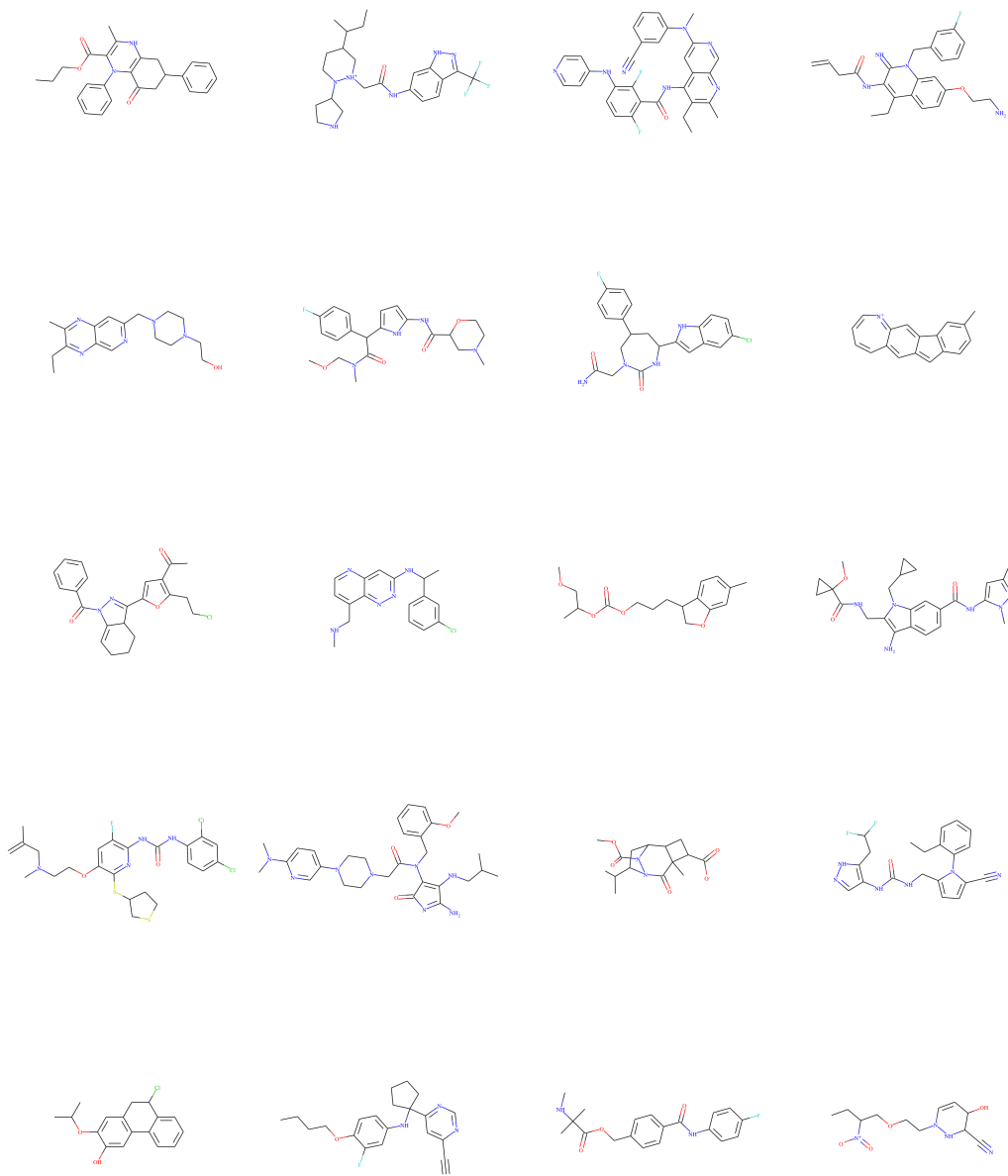


Figure 4: Examples of generated compounds.

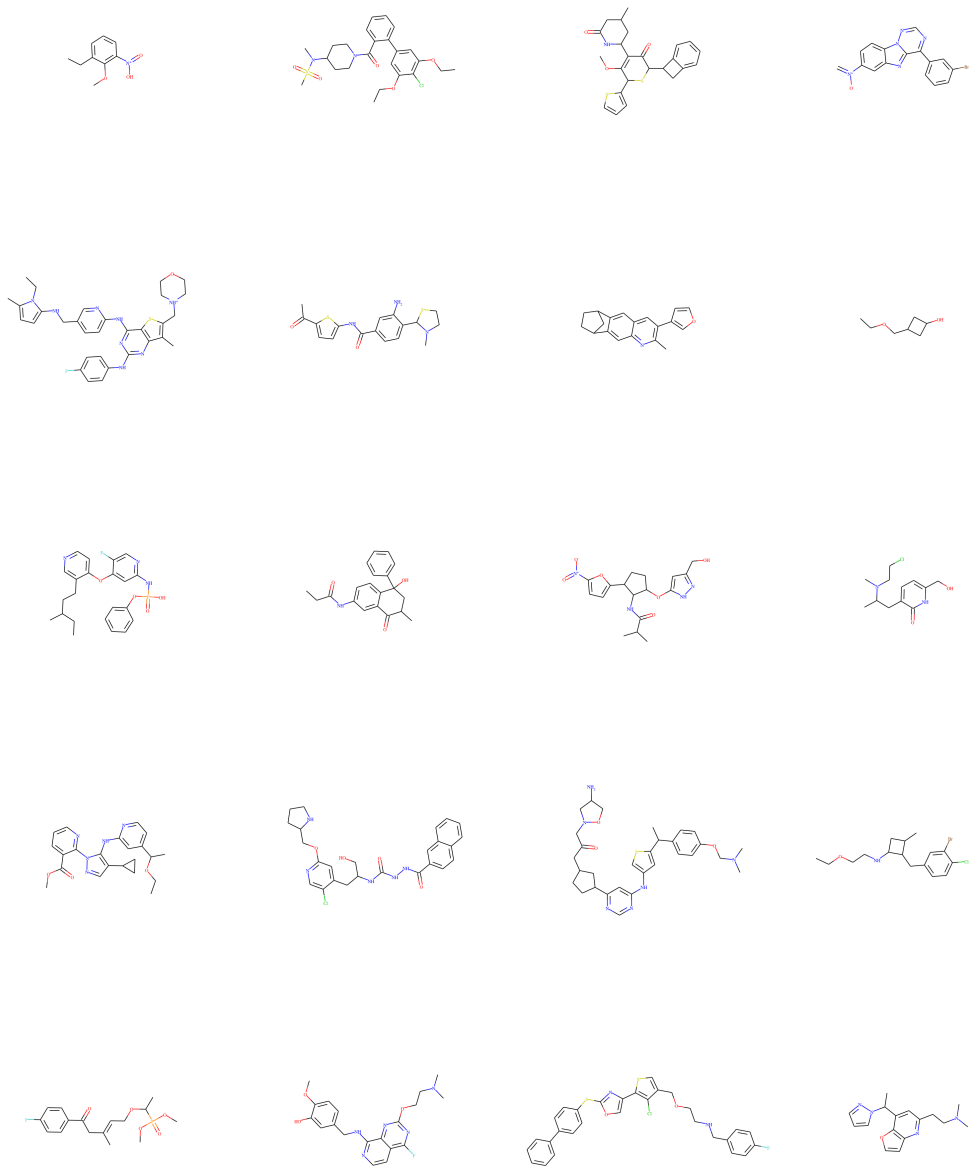


Figure 5: Examples of generated compounds.

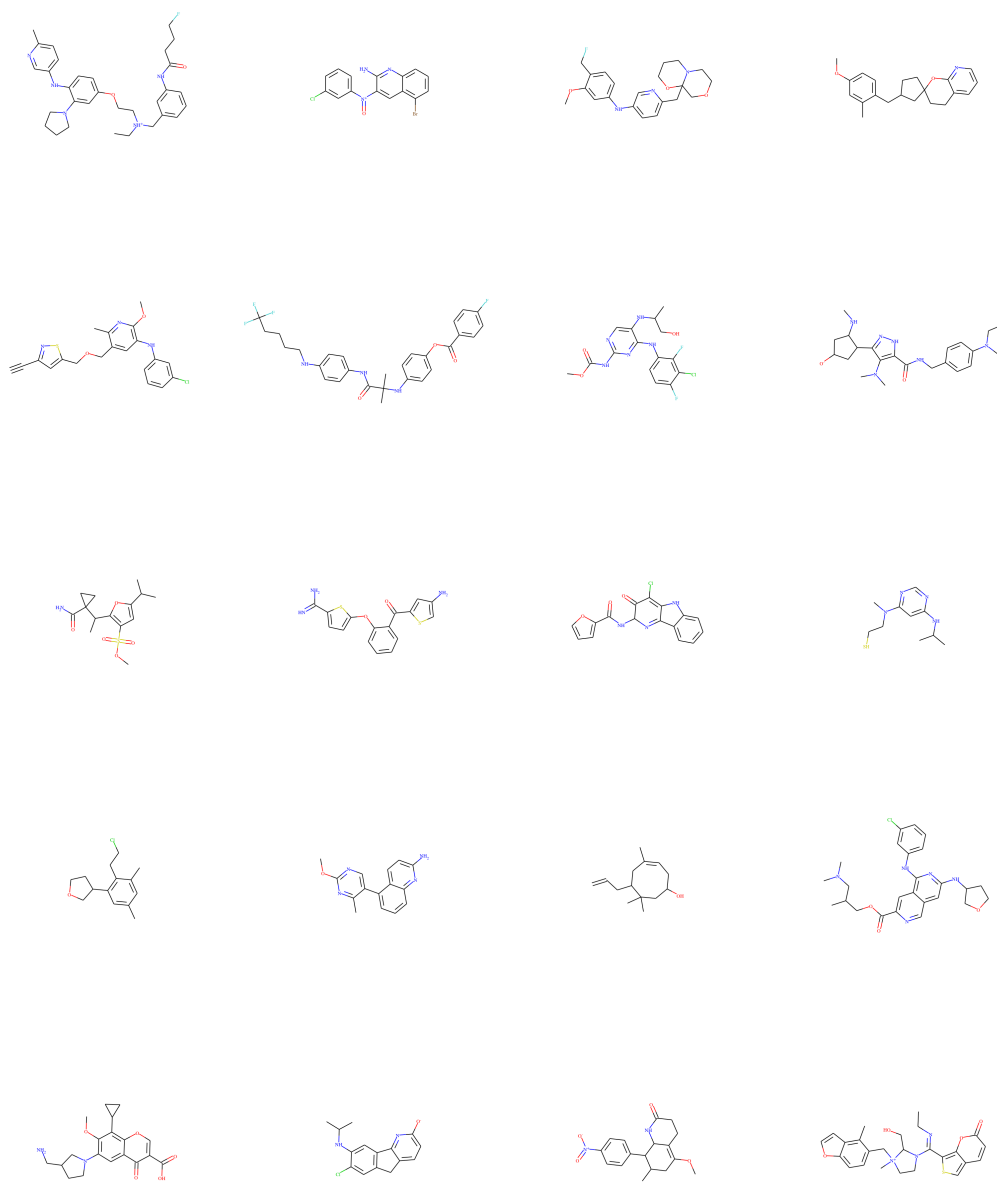


Figure 6: Examples of generated compounds.

References

- (1) Krizhevsky, A.; Sutskever, I.; Hinton, G. E. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*. 2012; pp 1097–1105.
- (2) Gehring, J.; Auli, M.; Grangier, D.; Yarats, D.; Dauphin, Y. N. Convolutional sequence to sequence learning. *arXiv preprint arXiv:1705.03122* **2017**,
- (3) Hochreiter, S.; Bengio, Y.; Frasconi, P.; Schmidhuber, J. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies. 2001.
- (4) Hochreiter, S.; Schmidhuber, J. Long short-term memory. *Neural computation* **1997**, *9*, 1735–1780.
- (5) Chung, J.; Gulcehre, C.; Cho, K.; Bengio, Y. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555* **2014**,
- (6) Riniker, S.; Landrum, G. A. Open-source platform to benchmark fingerprints for ligand-based virtual screening. *J. Cheminf.* **2013**, *5*, 26.
- (7) Kingma, D. P.; Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* **2014**,
- (8) Abadi, M. et al. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. 2015; <https://www.tensorflow.org/>, Software available from tensorflow.org.

2.2 Publication 2: Efficient Multi-Objective Molecular Optimization in a Continuous Latent Space

Full Reference: *Winter, Robin, Montanari, Floriane, Steffen, Andreas, Briem, Hans, Noé, Frank, & Clevert, Djork-Armé (2019). Efficient multi-objective molecular optimization in a continuous latent space. Chemical science, 10(34), 8016-8024.*

DOI: 10.1039/C9SC01928F

Licence: CC-BY-NC

Journal/Conference: Chemical Science (Impact Factor: 9.8)

Source Code: <https://github.com/jrwnter/mso>

Paper's main contributions:

- We propose a novel method for efficient *in silico* optimization of chemical compounds with respect to a multi-objective value function.
- We utilize the continuous molecular representation proposed in our previous work in combination with the efficient heuristic optimization algorithm *Particle Swarm Optimization*.
- We demonstrate the competitive performance of our proposed optimization method in a public benchmark.
- We demonstrate the methods ability to optimize molecules with respect to multiple objectives at the same time.

Author's contribution to the paper:

- Conceptualization of the original idea and its application to molecular optimization and de novo design.
- Implementation of the method.
- Design and evaluation of experiments, data curation and analysis.
- Preparation and creation of initial draft and visualizations.

Acknowledgement: Reproduced from *Chemical Science, 10(34), 8016-8024* with permission from the Royal Society of Chemistry.

Cite this: *Chem. Sci.*, 2019, **10**, 8016

All publication charges for this article have been paid for by the Royal Society of Chemistry

Efficient multi-objective molecular optimization in a continuous latent space†

Robin Winter,^{id}*^{ab} Floriane Montanari,^{id}^a Andreas Steffen,^a Hans Briem,^a Frank Noé,^{id}^b and Djork-Arné Clevert^{id}^a

One of the main challenges in small molecule drug discovery is finding novel chemical compounds with desirable properties. In this work, we propose a novel method that combines *in silico* prediction of molecular properties such as biological activity or pharmacokinetics with an *in silico* optimization algorithm, namely Particle Swarm Optimization. Our method takes a starting compound as input and proposes new molecules with more desirable (predicted) properties. It navigates a machine-learned continuous representation of a drug-like chemical space guided by a defined objective function. The objective function combines multiple *in silico* prediction models, defined desirability ranges and substructure constraints. We demonstrate that our proposed method is able to consistently find more desirable molecules for the studied tasks in relatively short time. We hope that our method can support medicinal chemists in accelerating and improving the lead optimization process.

Received 18th April 2019
Accepted 2nd July 2019

DOI: 10.1039/c9sc01928f

rsc.li/chemical-science

1 Introduction

A key challenge in small molecule drug discovery is to find novel chemical compounds with desirable properties. Computational methods have long been used to guide and accelerate the search through the huge chemical space of druglike molecules. In virtual screening, for instance, computational models can be utilized to rank virtual libraries of chemical structures regarding selected properties such as the predicted activity towards a target of interest.^{2,3} However, given the estimated vast amount of druglike molecules (10^{23} – 10^{60}),⁴ a complete search through this space is computationally infeasible.

An alternative approach is to computationally generate new molecules (*de novo* design) with optimized properties without the need for enumerating large virtual libraries. Heuristic methods such as genetic algorithms were used to optimize selected properties on-the-fly.^{5–7} However, due to the discrete nature of the chemical space, defining rules to transform one molecule into another (*e.g.* mutation and crossover rules for the genetic algorithms) largely depends on human expert knowledge. Moreover, defining a finite set of possible transformation rules limits the optimization possibilities and thereby promising molecules might be missed.

With the recent rise of deep learning^{8,9} in the field of computational chemistry, new approaches for *de novo* drug design have emerged (for a comprehensive review of this field the interested reader is referred to ref. 10 and 11). Segler *et al.* trained a Recurrent Neural Network (RNN) to model a larger set of molecules represented by the Simplified Molecular Input Line Entry Specification (SMILES) notation.¹² The resulting model was not only able to reproduce the molecules in the training set, but also to generate novel structures. By further training on a focused set of structures with a certain property distribution (*e.g.* the activity towards a biological target) the novel generated molecules could be enriched with structures following this desired property distribution. Another strategy for fine-tuning a generative model is Reinforcement Learning.¹³ Reinforcement Learning aims at learning the optimal set of actions to optimize a defined reward in a given environment. In the case of *de novo* design, the reward can *e.g.* be defined by the molecular properties to be optimized. Olivecrona *et al.* utilized this concept to alter the generative process of a pre-trained RNN, in order to generate more molecules with desirable properties.¹⁴

Besides RNNs trained on the SMILES representation, other groups also utilized Generative Adversarial Neural Networks^{15–17} or other molecular representations such as the molecular graph.^{18–20} While these methods differ in how they generate molecules, they all apply Reinforcement Learning to enrich the generated molecules with structures that have desirable properties. The main drawback of such methods is the need to retrain the generative model every time the reward function changes. This becomes impractical in a typical drug discovery project as the optimization criteria usually change over time.

^aDepartment of Digital Technologies, Bayer AG, Berlin, Germany. E-mail: robin.winter@bayer.com

^bDepartment of Mathematics and Computer Science, Freie Universität Berlin, Berlin, Germany

† Electronic supplementary information (ESI) available: Details of the desirability scaling functions, high resolution figures and detailed results of the GuacaMol benchmark. See DOI: 10.1039/c9sc01928f



A method that decouples the generation of molecules from the optimization problem was originally proposed by Gómez-Bombarelli *et al.*²¹ In their work, a variational autoencoder was trained on the SMILES notation of a large set of molecules. As a result, a new continuous vector representation of chemical structures was obtained. Points in this continuous space correspond to molecules in the discrete chemical space (as represented by the SMILES notation) and *vice versa*. Novel structures can be generated by sampling arbitrary points in the continuous space and then transforming them back to the SMILES notation. A molecular transformation can be achieved by a simple shift in the vector representation. Thus, optimizing chemical structures with respect to selected properties can be directly performed by optimizing a reward function in the continuous space. In their work, Gómez-Bombarelli *et al.* utilized Bayesian Optimization to find points in the space that correspond to molecules with a high drug-likeness and synthetic accessibility. More recently, Jin *et al.* also used Bayesian Optimization to optimize molecules generated by a variational autoencoder based on molecular graphs.²² Bayesian Optimization is a powerful method that has proven useful in the optimization of functions that are computationally expensive to evaluate as it needs a comparably low amount of function evaluations.²³ However, its computational complexity increases exponentially with the number of dimensions of the optimization space.²⁴ In the case of molecular optimization, though, function evaluations are relatively cheap (prediction of molecular properties) and the dimensionality of the search space (continuous molecular representation) relatively high.

In this work, we propose the use of a more light weight heuristic optimization method termed Particle Swarm Optimization (PSO). Hartenfeller *et al.* already proposed in 2008 to apply PSO on a discrete chemical space represented by a large library of molecular building blocks and chemical reactions.²⁵ Here, we apply PSO in our continuous chemical representation reported previously.²⁶ As particles of the swarm navigating this representation correspond to actual molecules in the chemical space, we term our method Molecule Swarm Optimization (MSO). In three different experiments we show how our proposed method can be utilized to optimize molecules with respect to a single objective, under constraints with regard to chemical substructures and with respect to a multi-objective value function.

2 Methods

2.1 Continuous chemical representation

Our approach can be used with any continuous representation of the chemical space, such as those found in.^{21,22,27} In this study we build upon the continuous molecular representation framework reported earlier.²⁶ This molecular representation was learned using a Deep Neural Network to translate from one molecular string representation (*e.g.* SMILES) to another. In this way, the model learns the common “essence” between both syntactically different string notations, *i.e.* the molecule which both notations are representing. By introducing a bottleneck in the architecture of the neural network, the molecule is encoded

in a compressed embedding, that can be utilized as latent representation of the chemical space. As the model is trained on a huge dataset of approximately 75 million chemical structures stemming from various sources, the resulting latent space represents a wide range of the chemical space that can be explored.

In this earlier work, we also showed that the learned molecular representation can be utilized as powerful molecular descriptors for quantitative structure activity relationships (QSAR) models. Moreover, transformations in the latent space result, if decoded back, in smooth transformations in the discrete chemical space in regard of structural as well as molecular properties. The interested reader is directed to the original publication for technical details of our framework.²⁶

2.2 Particle swarm optimization

Particle Swarm Optimization (PSO) is a stochastic optimization technique that mimics swarm intelligence to find an optimal point in a search space as defined by an objective function. The particle swarm consists of individual agents (particles) that explore the search space, utilizing information gained during their search and “communicating” with other particles in the swarm.²⁸

This concept can be defined by a few simple equations. The N particles in the swarm are defined by their position x and velocity v . The potential surface of the search space can be evaluated by the objective function f . The movement of the i -th particle at iteration step k is influenced by the best point it has yet discovered

$$x_i^{\text{best}} = \text{argmax}_f(x_i^k) \quad (1)$$

as well as the overall best point yet discovered

$$x^{\text{best}} = \text{argmax}_f(x_i^{\text{best}}). \quad (2)$$

After each iteration, each particle updates its velocity v_i and position x_i in the following way:

$$v_i^{k+1} = wv_i^k + c_1r_1(x_i^{\text{best}} - x_i^k) + c_2r_2(x^{\text{best}} - x_i^k) \quad (3)$$

$$x_i^{k+1} = x_i^k + v_i^{k+1} \quad (4)$$

where c_1 and c_2 are constants that weight the contribution of the particles individual experience *versus* the swarm experience. r_1 and r_2 are random numbers drawn from independent uniform distributions between 0 and 1. The so called inertia weight w is a constant that controls the momentum of the particle from the previous iteration.

2.3 Objective function

The search of the Particle Swarm is guided by the objective function that is defined to be maximized. In drug discovery, the optimized objective can be both complex, conflicting, ill-defined or evolving over time. For example, at the early stages of lead discovery, a higher emphasis is put on increasing biological activity and gaining structure-activity-relationship



knowledge. A set of targets that should not be hit by the compounds can also be introduced at that stage. Later on, when the overall activity landscape is well understood by the team, the focus of the optimization evolves more towards pharmacokinetics-related properties (ADME) such as improving solubility, metabolic stability or cell permeability, *etc.* These different objectives can contradict themselves, for example increasing the solubility of a compound might lead to permeability problems. This makes a multi-parameter optimization notoriously challenging.

In order to keep the method flexible, we propose different individual objective functions that can be combined and weighted:

- Fixed ranges for molecular properties such as molecular weight, number of H-bond donors, octanol–water partition coefficient, stereocenters *etc.*
- Ad-hoc QSAR models to predict the biological activity of the molecules with respect to targets of interest.
- ADME models to predict solubility, metabolic stability, cell permeability and efflux rate.
- Scoring functions for chemical reasonableness like the synthetic accessibility (SA) score²⁹ or drug-likeness (QED).³⁰
- Penalties for unwanted (*e.g.* toxic) or uncommon substructures.
- Rewards for defined substructures (*e.g.* fixing a scaffold) or similarity to a certain compound.

These functions either work directly on the continuous representation (*e.g.* QSAR models, similarity) or on the decoded SMILES representation utilizing the cheminformatics library RDKit.³¹ In this study, we build two biological activity models for prediction of Epidermal Growth Factor Receptor (EGFR) and asparyl protease b-site APP cleaving enzyme-1 (BACE1) activity. These targets were chosen as the QSAR models build for these targets showed reasonable predictive performance in our prior work.²⁶ Compounds with reported IC₅₀ values were extracted from ChEMBL³² and preprocessed as described in this prior work. We encoded the molecules in the continuous representation and trained Support Vector Machine (SVM) regression models (as implemented in the Python library scikit-learn³³) on predicting the IC₅₀ values of the compounds. Moreover, we trained SVMs on solubility, metabolic stability and membrane permeability endpoints utilizing in-house data. SVM hyperparameters were optimized in a 5-fold cross-validation.

In order to filter for unwanted substructures, we extracted known toxic substructures from a published collection by SureChEMBL.^{34,35} Moreover, to filter for possible unstable structures we created a list with extended-connectivity fingerprints (ECFP4) of substructures that occur in more than 5 individual compounds in ChEMBL. Roughly 1% of the compounds in ChEMBL have substructures that occur less often and are here considered as potentially unstable. During the optimization, generated structures are penalized if they contain such known toxic substructures or have uncommon ECFP4 bits.

In order to combine the scores of the different functions in a multi-objective setting, we follow the approach reported in³⁶ and scale each function between 0 and 1 reflecting values of low

to high desirability (see ESI† for more details). The scaled scores of each function are combined as the weighted average, where the weights correspond to priorities in different tasks. The resulting desirability score (dscore) is subsequently used as the objective function for the PSO algorithm.

2.4 Optimization model

The final optimization model combines the parts mentioned above, *i.e.* the continuous molecular representation, the optimization algorithm and the objective functions. Either a query molecule is encoded in the continuous space or a random point is sampled. The PSO algorithm is initialized by generating a fixed amount of particles (in the order of 100) at this position with randomly drawn initial velocities. After the first position update, the objective function is evaluated for each individual particle of the swarm. The search is continued until a certain number of iterations or a break condition (*e.g.* desired value) is met. Since the PSO algorithm is a stochastic optimization technique, multiple restarts are performed.

3 Results and discussion

By combining our encoder-decoder framework, computational models to predict properties and/or biological activities of compounds, and an optimization algorithm, we optimize a query molecule with respect to the objective function resulting in a set of compounds with more desirable (predicted) properties. In the first part we show the optimization of molecules with regard to a single objective. Next, we further restrict the optimization by adding substructure constraints and then demonstrate that our framework can be utilized to perform multi-objective optimization of compounds. In our final experiment we benchmark our proposed model with the GuacaMol¹ package.

3.1 Single-objective optimization

As a first proof-of-principle, we run experiments on the optimization of molecules with respect to single molecular properties. Similar to related works,^{15,20,21,37} we perform individual optimizations on the drug-likeness, the octanol–water partition coefficient logP as well as the synthetic accessibility of a molecule. We utilize the RDKit implementation of the Quantitative Estimate of Druglikeness (QED) score to evaluate the drug-likeness and the penalized logP score³⁷ that combines the partition coefficient and the synthetic accessibility (SA) score of a compound. Moreover, we optimize compounds with respect to their predicted biological activity on EGFR and BACE1.

For each optimization task, we run the PSO algorithm 100 times for 100 steps each. Table 1 shows for each task the highest score achieved. For the drug-likeness and the penalized logP tasks, we compare our method to the best results of state-of-the-art optimization models as reported by You *et al.*²⁰ Our proposed method achieves the same performance on the drug-likeness task as the best state-of-the-art approach and outperforms all other approaches on the penalized logP task. Moreover, our method consistently generated molecules with very



high predicted binding-affinities ($IC_{50} < 1$ nM) for both biological targets respectively. As the compounds used to train both QSAR models were not included in the pre-training dataset of the encoding-decoding framework, these high scores can not be attributed to an information leakage or bias in the generative model. In fact, we investigate if including these compounds in the pre-training influences the optimization results. We found that both models performed similarly with overlapping confidence intervals, suggesting that time-costly fine-tuning steps are not needed here. Fig. 1 depicts the average scores during the optimization process for the different tasks. To better understand the impact of the starting point for the optimization, we show results for a fixed starting point (benzene) and for variable starting points, randomly sampled from ChEMBL. In all tasks, the model consistently optimizes the respective property, reaching relatively high scores already after a few iterations. Although starting from different points, the variance between the scores at a given optimization step is similar to the variance obtained when repeatedly starting from benzene. As a matter of fact, the variance seems to be higher for the fixed starting point. Moreover, starting from a molecule picked from ChEMBL seems to result in faster and more successful optimizations, except for the optimization of the penalized logP score. This is probably due to the increased size and structural complexity of these molecules compared to a simple benzene ring. Moreover, initializing the particle swarm algorithm with more particles helps finding more optimal points in the search space in less iterations (at the expense of increased computational time).

Similar to related work,^{21,22} we also tried to optimize the latent space of our autoencoder framework with Bayesian Optimization (BO). However, trying different BO frameworks, we were not able to find good solutions within reasonable computation time. This is probably due to the high dimensionality of our search space (512 dimensions), since BO's computational complexity grows exponentially with the number of dimensions.²⁴

It has to be mentioned that the baseline methods in Table 1 were trained on significantly less data ($\approx 250,000$ compounds) which might lead to an unfair advantage for our proposed method. To investigate the impact of a smaller pre-training dataset on our optimization results, we retrained our encoder-decoder framework with the same dataset as used in ref. 20. We find that for this model different runs have a higher variance in performance, however, the best solutions

still have a similar high score as the solutions reported in Table 1. Thus, the performance of our proposed model cannot only be explained by the increased size of the pre-training dataset.

Using one GPU for passing the molecules in the swarm through the encoder-decoder model and one CPU core to perform the PSO algorithm and objective function evaluation, computational time is in the order of a few minutes for a 100-steps run. The run time on a 32-core CPU machine without GPU support is in the order of 10 minutes. This is in contrast to baseline methods in Table 1 which have a reported wall clock running time of multiple hours to one day on a 32-core CPU machine.²⁰ This substantial speed-up is mainly due to the fact that our proposed method separates the training of the generative model (the decoder of the utilized encoder-decoder framework) and the optimization procedure. Since we use the same pre-trained generative model for each optimization task, we do not have to spend computational resources on training this model in every new run. This is in contrast to the methods used for comparison, as they approach the optimization task by re-training/fine-tuning the generative model for every new task.

Fig. 2 shows a few example molecules randomly picked from the final iteration for each optimization task. It is evident that, by optimizing a single objective function, the model exploits its ability to freely navigate the chemical space and solely focuses on this very objective. While the optimization of drug-likeness obviously results in pleasant-looking molecules, the three other optimization tasks result into comparably "unusual" chemistry. Since long aliphatic chains are both maximizing the partition coefficient while being scored as easy to synthesize, they are the inevitable outcome when optimizing for penalized logP. Moreover, if the objective is solely maximizing the prediction of a QSAR model, the resulting molecules will aggregate the evidence the QSAR model found in the potent molecules of its training set. This, however, will most likely guide the optimizer into parts of the chemical space that are not well covered by training set molecules, leading to inaccurate and overoptimistic predictions. Thus, final molecules are likely to be artifacts of the underlying QSAR model.

Summing up, we demonstrated that our method is able to very quickly improve upon a given starting molecule in terms of predicted drug-likeness or predicted biological activity. This confirms that our optimization method is able to navigate the chemical space defined by our pre-trained embedding and perform single-parameter optimization in a timely fashion. However, these examples are far from actual drug design cases. At this stage, we do not apply any structural constraints for guiding the structure generation. This means that the new, optimized compounds might be structurally remote from the defined starting points or contain toxic or unstable moieties (e.g. 1,3-pentadiyne substructure in Fig. 2d). Usually, drug discovery projects are confined within chemical series and closely related analogues. Hence, we propose to add constraints on the chemical structure during optimization in the following section.

Table 1 Best results for the different single-objective optimization tasks. Our Method is compared to the results of three recently published optimizations methods as reported by You *et al.*²⁰

	ORGAN	JT-VAE	GCPN	MSO
Reference	15	22	20	Ours
Penalized logP	3.63	5.30	7.98	26.1
QED	0.896	0.925	0.948	0.948
EGFR [pIC ₅₀]	—	—	—	10.3
BACE1 [pIC ₅₀]	—	—	—	11.5
Run time	~1 d	~1 d	~8 h	~10 m



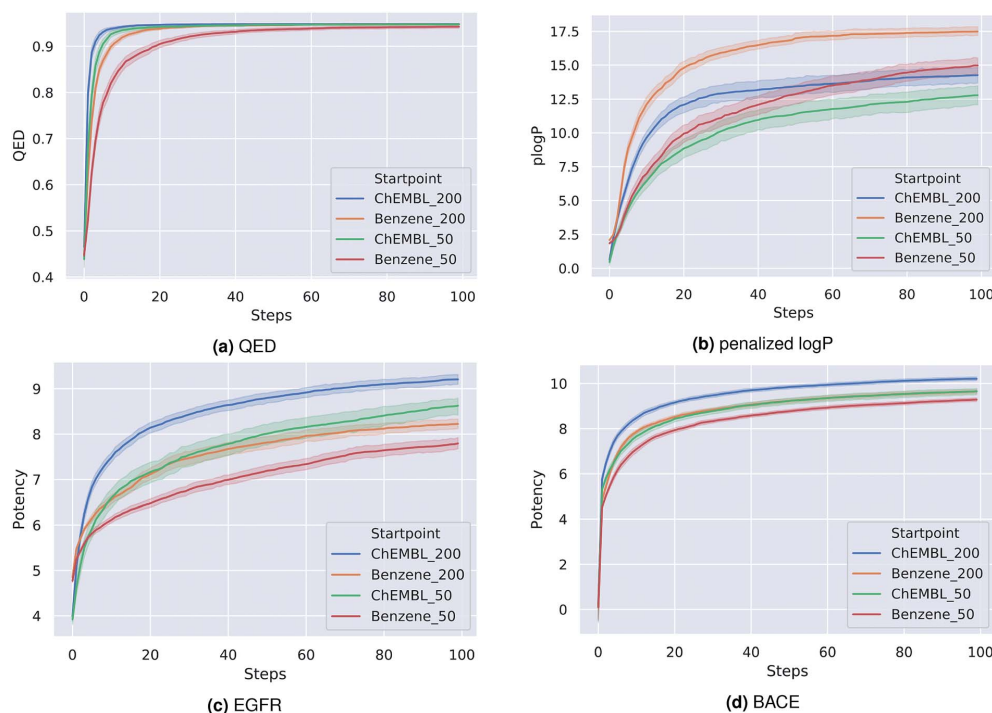


Fig. 1 Best score of the particle swarm over the course of optimization averaged over 100 runs for four optimization tasks: (a) quantitative estimation of drug-likeness, (b) penalized octanol–water partition coefficient, (c) binding affinity (pIC_{50}) to Epidermal Growth Factor Receptor, (d) binding affinity (pIC_{50}) to aspartyl protease b-site APP cleaving enzyme-1. Optimizations were either started from benzene or a random picked compound from ChEMBL with either 50 or 200 particles.

3.2 Constrained single-objective optimization

In this experiment, we perform a single property optimization, while constraining the explorable chemical space using a defined molecular scaffold that needs to be present in the generated molecules. This task is more closely related to a real drug-development process, as it mimics a typical lead optimization task.

We base our experiment on a lead optimization paper by Stamford *et al.* in which an iminopyrimidine lead series was optimized for BACE1 inhibition.³⁸ In order to evaluate whether our method can optimize for biological activity while constraining the explorable chemical space, we start the optimization from the same initial compound as in³⁸ (Fig. 3b). We fix the iminopyrimidinone scaffold (Fig. 3a) by penalizing generated compounds that do not contain this substructure in the objective function. For the prediction of BACE1 activity we retrain the BACE1 model from the previous section, excluding all compounds with an iminopyrimidinone scaffold.

On the 17 iminopyrimidinone compounds reported by Stamford *et al.* the QSAR model achieves a Spearman correlation coefficient of $\rho = 0.7$ (p -value = 0.004) compared to the reported IC_{50} values. This means that although we did not

include compounds with an iminopyrimidinone scaffold in the training, the BACE1 activity prediction model shows a reasonable performance in the part of the chemical space we are interested in (*i.e.* compounds with iminopyrimidinone scaffold).

In addition to fixing the scaffold, we further restrict the chemical space by penalizing compounds with more than 26 heavy atoms (one more heavy atom than the best reported compound by Stamford *et al.* depicted in Fig. 3d). Moreover, we penalize for known toxic and uncommon substructures.

We run the optimization model for 100 steps and restart the optimization 200 times. Fig. 3d–f depicts the compounds with the best scores found in the *in silico* optimization. In the course of the optimization the most active BACE1 inhibitor (compound c) from Stamford *et al.* was passed by in 2 out of the 200 runs but was not part of the final set of proposed molecules. This can be explained as the members of the final set of molecules all have higher predicted activities than compound c. As a proof of principle, we also performed an experiment where the Euclidean distance of a particle to compound c's embedding was used as objective function. In this experiment, we could consistently (200 out of 200 times) recover compound c as the optimal solution. Hence, we



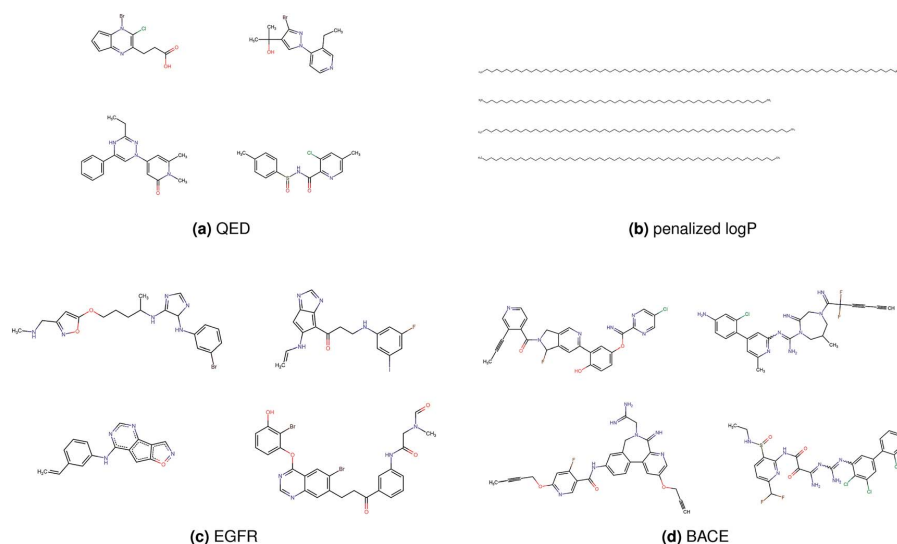


Fig. 2 Compound examples resulting from optimizing (a) quantitative estimation of drug-likeness, (b) penalized octanol-water partition coefficient, (c) binding affinity to Epidermal Growth Factor Receptor, (d) binding affinity to aspartyl protease b-site APP cleaving enzyme-1.

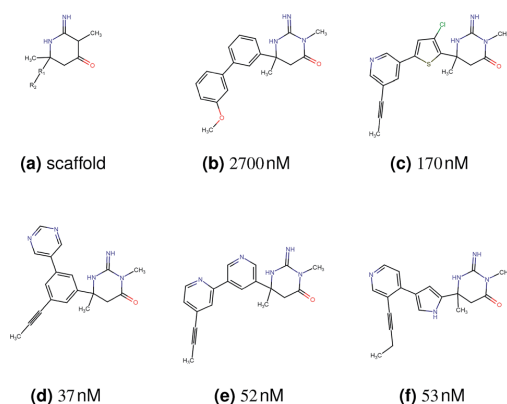


Fig. 3 (a) Iminopyrimidinone scaffold fixed in the optimization. R1 and R2 are aromatic rings. (b) Starting point of the optimization. (c) Best reported compound by Stamford *et al.* (d–f) Top 3 compounds found by our method. All compounds are depicted with their predicted binding affinity to aspartyl protease b-site APP cleaving enzyme-1 (BACE1).

suppose that our approach does not contain compound c within the final set of molecules because this region of chemical space is not the most attractive for the applied BACE1 model. Our reported *in silico* solutions do contain the required scaffold and are predicted to have a higher potency, so they might possibly give useful new ideas to medicinal chemists working on BACE1 inhibitors with an iminopyrimidinone scaffold.

3.3 Multi-objective optimization

In this last section, we evaluate the ability of our proposed method to optimize a molecule with respect to a multi-objective value function. In accordance to a typical multi-objective lead optimization process, we define the value function as a combination of multiple molecular properties. We conduct three experiments:

1. Maximizing the predicted binding affinity to EGFR while minimizing the predicted binding affinity to BACE1.
2. Maximizing the predicted binding affinity to BACE1 while minimizing the predicted binding affinity to EGFR.
3. Maximizing the predicted binding affinity to both EGFR and BACE1.

Additionally for all experiments, we maximize the predicted solubility, metabolic stability, cell permeability, drug-likeness as well as the predicted synthetic accessibility (SA) of the molecule and penalize for known toxic or uncommon substructures and molecular weight below 200 or above 600 Dalton. Each of the ten different objectives was scaled by a desirability function between 0 and 1, where low values correspond to undesirable ranges and high values to acceptable or good ranges of a molecular property (see ESI† for details on the scaling functions). The final optimization objective is the weighted average of the different scaled objective functions. In all experiments, we weighted the maximization of binding affinities with a factor of 5, minimization of binding affinity with a factor of 3 and all other properties with 1.

Each of the three optimization tasks was run 100 times for 200 steps, starting from a randomly picked molecule from ChEMBL. The aggregated results for the different properties over the course of the optimization are depicted in Fig. 4. Our

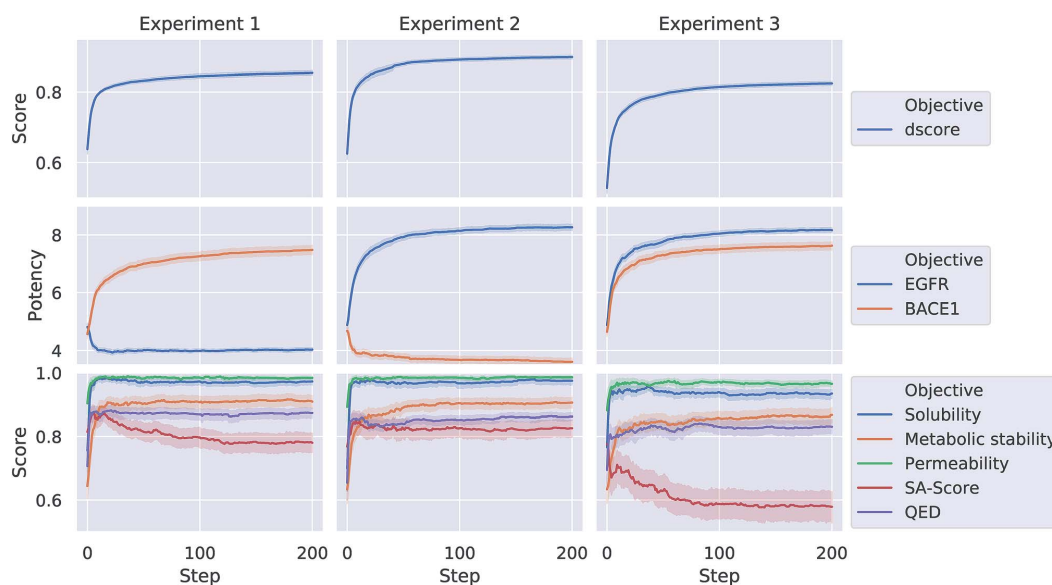


Fig. 4 Results for the best scoring (dscore) particle in the swarm over the course of optimization for 200 steps averaged over 100 runs for the three different optimization tasks. In addition to the objective (dscore) that is optimized, the average predicted potency (pIC₅₀) on both BACE1 and EGFR as well as the average scaled predicted solubility, metabolic stability, permeability, SA score and QED of the best scoring (dscore) particle is shown.

proposed method is consistently able to optimize a query molecule with respect to the defined multi-objective value function. The weighted average of the different objective functions (dscore) increases on average from 0.64, 0.63 and 0.53 to 0.85, 0.9 and 0.82 for the three different experiments respectively. The method is able to find solutions that are predicted to meet the respective activity-profile while keeping desirable ADMET properties as well as QED and SA scores high. In experiment 3, however, the methods finds solutions that on average have a comparably lower SA score in order to meet the desired activity-profile.

Table 2 shows a few example molecules picked from the best final iteration for each of the three optimization tasks. Although the value function consists of many different and partially conflicting individual objectives our proposed method is consistently able to find molecules in the vast chemical space that meet the desirable ranges for all of the defined objectives.

4 GuacaMol benchmark

In order to assess our proposed model's performance in a more standardized framework we utilized the recently published benchmark package GuacaMol proposed by Brown *et al.*¹ The benchmark consists of 20 optimization tasks including a range of single-, constrained and multi-objective optimization tasks (goal-directed benchmarks v2). In contrast to the previous evaluations, this benchmark does not only consider the highest-scoring solution, but also takes up to top-250 solutions for

scoring into account. For a fair comparison, we retrained our encoder-decoder framework with the same subset of ChEMBL that is defined in the benchmark. For each optimization task we used a particle swarm with 200 particles that was run for 250 iteration and 40 restarts.

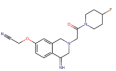
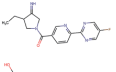
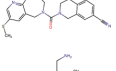
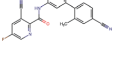
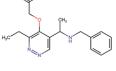
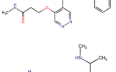
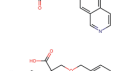
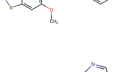
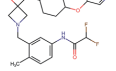
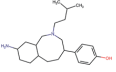
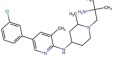
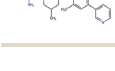
We found that our proposed method achieves a higher average score when compared with the baselines methods included in the benchmark. We were able to outperform these methods in 9 out of the 12 tasks that were not already perfectly solved by the best baseline method (Graph-GA). For detailed (task-wise) solutions we refer the reader to Table 1 in the ESI.†

5 Conclusion

We propose the use of Particle Swarm Optimization (PSO) heuristic to optimize molecules in a continuous latent representation. We show that our model is able to consistently optimize molecules with respect to single objectives such as maximizing the predicted drug-likeness, partition coefficient logP or target binding affinity as modeled by quantitative structure activity relationship (QSAR) model. Not only does our proposed method exhibit competitive or better performance in finding optimal solutions compared to baseline method, it also achieves significant reduction in computational time. In the more standardized benchmark package GuacaMol we outperform the baseline methods in 9 out of 12 tasks that were not already perfectly solved by the best baseline method. In further experiments we show how the proposed method can be utilized



Table 2 Four example compounds as result of the *in silico* optimization for experiment 1, 2 and 3 respectively, separated by blank rows. Each compound is shown with its calculated and predicted molecular properties: binding affinity (pIC₅₀) to target Epidermal Growth Factor Receptor (EGFR), binding affinity (pIC₅₀) to target aspartyl protease b-site APP cleaving enzyme-1 (BACE), metabolic stability (Stab) in percent, solubility from powder (Sol) in mg L⁻¹, cell permeability (Perm) in nm s⁻¹, drug-likeness (QED) and synthetic accessibility (SA)

Compound	EGFR	BACE	Stab	Sol	Perm	QED	SA
	3.7	8.0	86	390	72	0.90	3.0
	4.4	8.5	86	500	130	0.94	3.4
	4.1	8.3	78	570	90	0.73	3.0
	4.3	8.3	86	25	69	0.69	2.8
	8.6	2.4	82	1000	85	0.77	2.9
	9.1	3.1	68	610	100	0.73	2.9
	8.4	3.5	86	280	67	0.82	2.8
	9.0	3.6	80	230	74	0.70	3.2
	8.7	8.2	59	838	80	0.63	3.0
	8.1	8.0	53	531	120	0.85	3.4
	8.1	7.7	85	390	84	0.78	3.3
	7.9	7.9	68	1000	130	0.78	3.6

to support medicinal chemists in a lead optimization process by proposing *in silico* solutions for relevant tasks. Finally, we perform multi-objective optimizations of compounds with respect to relevant properties such as specific target activity profiles and pharmacokinetics-related properties. We demonstrate that our proposed method is consistently able to optimize the joined objective function and results in compounds that exhibit desirable ranges for all included computed properties.

Although we show promising results for optimizing molecular properties in this work, it has to be noted that the optimization cycles are based on predicted values for the properties. This can be particularly problematic for QSAR models that are applied outside their domain of applicability. Hence, we advocate the use of our proposed method only in combination with reasonable constraints to parts of the chemical space that can be modeled by the applied functions with reasonable confidence. An even more suitable approach would be combining the *in silico* optimization with real world experiments in an active learning manner. By doing so, the QSAR model could be refitted while evolving into yet unexplored regions of the chemical space and hopefully remain reasonably accurate in its predictions.

Availability

Source code of the proposed model will be made openly available on GitHub (<https://github.com/jrwnter/mso>) upon publication.

Conflicts of interest

There are no conflicts to declare.

Acknowledgements

This project was supported by several research grant. FN acknowledges funding from the European Commission (ERC CoG 772230 "ScaleCell") and MATH+ (AA1-6). DC and FM acknowledge funding from the Bayer AG Life Science Collaboration ("DeepMinDS"). RW acknowledges Bayer AG's PhD scholarship.

Notes and references

- 1 N. Brown, M. Fiscato, M. H. Segler and A. C. Vaucher, *J. Chem. Inf. Model.*, 2019, **48**, 1096–1108.
- 2 B. K. Shoichet, *Nature*, 2004, **432**, 862.
- 3 W. P. Walters, M. T. Stahl and M. A. Murcko, *Drug Discovery Today*, 1998, **3**, 160–178.
- 4 P. G. Polishchuk, T. I. Madzhidov and A. Varnek, *J. Comput.-Aided Mol. Des.*, 2013, **27**, 675–679.
- 5 M. Reutlinger, T. Rodrigues, P. Schneider and G. Schneider, *Angew. Chem., Int. Ed.*, 2014, **53**, 4244–4248.
- 6 F. Dey and A. Cafilisch, *J. Chem. Inf. Model.*, 2008, **48**, 679–690.
- 7 Y. Yuan, J. Pei and L. Lai, *J. Chem. Inf. Model.*, 2011, **51**, 1083–1091.
- 8 Y. LeCun, Y. Bengio and G. Hinton, *Nature*, 2015, **521**, 436.
- 9 J. Schmidhuber, *Neural Networks*, 2015, **61**, 85–117.
- 10 K. T. Butler, D. W. Davies, H. Cartwright, O. Isayev and A. Walsh, *Nature*, 2018, **559**, 547.
- 11 D. C. Elton, Z. Boukouvalas, M. D. Fuge and P. W. Chung, 2019, arXiv preprint arXiv:1903.04388.
- 12 M. H. Segler, T. Kogej, C. Tyrchan and M. P. Waller, *ACS Cent. Sci.*, 2017, **4**, 120–131.
- 13 R. S. Sutton, A. G. Barto, *et al.*, *Introduction to reinforcement learning*, MIT press Cambridge, 1998, vol. 135.



- 14 M. Olivecrona, T. Blaschke, O. Engkvist and H. Chen, *J. Cheminf.*, 2017, **9**, 48.
- 15 G. L. Guimaraes, B. Sanchez-Lengeling, C. Outeiral, P. L. C. Farias and A. Aspuru-Guzik, 2017, arXiv preprint arXiv:1705.10843.
- 16 B. Sanchez-Lengeling, C. Outeiral, G. L. Guimaraes and A. Aspuru-Guzik, 2017, arXiv preprint arXiv:1705.10843.
- 17 E. Putin, A. Asadulaev, Q. Vanhaelen, Y. Ivanenkov, A. V. Aladinskaya, A. Aliper and A. Zhavoronkov, *Mol. Pharmaceutics*, 2018, **15**, 4386–4397.
- 18 N. De Cao and T. Kipf, 2018, arXiv preprint arXiv:1805.11973.
- 19 Y. Li, L. Zhang and Z. Liu, 2018, arXiv preprint arXiv:1801.07299.
- 20 J. You, B. Liu, R. Ying, V. Pande and J. Leskovec, 2018, arXiv preprint arXiv:1806.02473.
- 21 R. Gómez-Bombarelli, J. N. Wei, D. Duvenaud, J. M. Hernández-Lobato, B. Sánchez-Lengeling, D. Sheberla, J. Aguilera-Iparraguirre, T. D. Hirzel, R. P. Adams and A. Aspuru-Guzik, *ACS Cent. Sci.*, 2018, **4**, 268–276.
- 22 W. Jin, R. Barzilay and T. Jaakkola, 2018, arXiv preprint arXiv:1802.04364.
- 23 J. Snoek, H. Larochelle and R. P. Adams, *Advances in neural information processing systems*, 2012, pp. 2951–2959.
- 24 Z. Wang, M. Zoghi, F. Hutter, D. Matheson and N. De Freitas, *Twenty-Third International Joint Conference on Artificial Intelligence*, 2013.
- 25 M. Hartenfeller, E. Proschak, A. Schüller and G. Schneider, *Chem. Biol. Drug Des.*, 2008, **72**, 16–26.
- 26 R. Winter, F. Montanari, F. Noé and D.-A. Clevert, *Chem. Sci.*, 2019, **10**, 1692–1701.
- 27 E. Bjerrum and B. Sattarov, *Biomolecules*, 2018, **8**, 131.
- 28 J. Kennedy, *Encyclopedia of machine learning*, Springer, 2011, pp. 760–766.
- 29 P. Ertl and A. Schuffenhauer, *J. Cheminf.*, 2009, **1**, 8.
- 30 G. R. Bickerton, G. V. Paolini, J. Besnard, S. Muresan and A. L. Hopkins, *Nat. Chem.*, 2012, **4**, 90.
- 31 G. Landrum, *et al.*, *RDKit: Open-source cheminformatics*, 2006.
- 32 A. Gaulton, L. J. Bellis, A. P. Bento, J. Chambers, M. Davies, A. Hersey, Y. Light, S. McGlinchey, D. Michalovich, B. Al-Lazikani, *et al.*, *Nucleic Acids Res.*, 2011, **40**, D1100–D1107.
- 33 F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, *et al.*, *J. Mach. Learn. Res.*, 2011, **12**, 2825–2830.
- 34 *Non MedChem-Friendly SMARTS*, <https://www.surechembl.org/knowledgebase/169485-non-medchem-friendly-smarts>, accessed: 2019-04-03.
- 35 I. Sushko, E. Salmina, V. A. Potemkin, G. Poda and I. V. Tetko, *ToxAlerts: a web server of structural alerts for toxic chemicals and compounds with potential adverse reactions*, 2012.
- 36 D. J. Cummins and M. A. Bell, *J. Med. Chem.*, 2016, **59**, 6999–7010.
- 37 M. J. Kusner, B. Paige and J. M. Hernández-Lobato, 2017, arXiv preprint arXiv:1703.01925.
- 38 A. W. Stamford, J. D. Scott, S. W. Li, S. Babu, D. Tadesse, R. Hunter, Y. Wu, J. Misiaszek, J. N. Cumming, E. J. Gilbert, *et al.*, *ACS Med. Chem. Lett.*, 2012, **3**, 897–902.



Supporting Information: Efficient Multi-Objective Molecular Optimization in a Continuous Latent Space

Robin Winter, Floriane Montanari, Andreas Steffen,
Hans Briem, Frank Noé, Djork-Arné Clevert

1 Desirability Scaling

We followed the approach proposed by Cummins and Bell [1] and scale each molecular property function prediction between 0 and 1, reflecting values of low to high desirability. For each property we defined a range of values with a respective desirability score. Between these points we interpolated linearly. The resulting scaling functions are depicted in Figure 1.

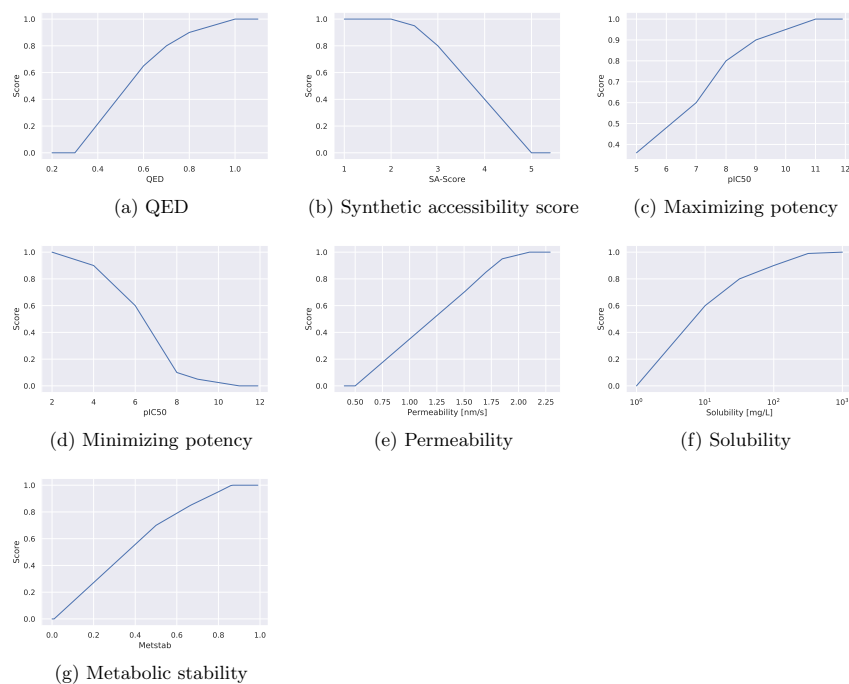


Figure 1: Scaling functions for different molecular properties.

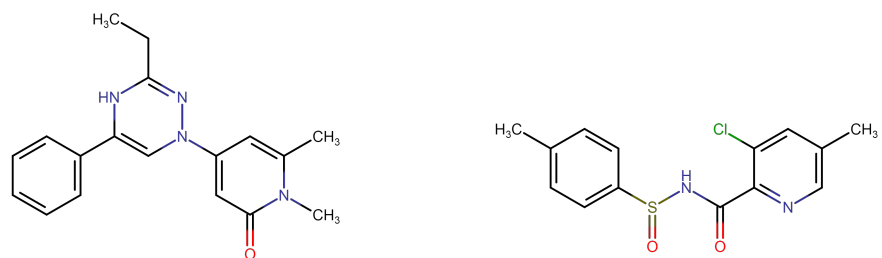
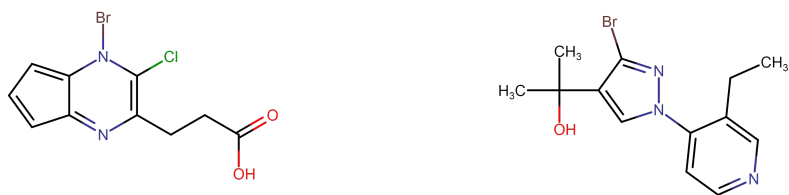


Figure 2: High resolution version of Figure 2(a).

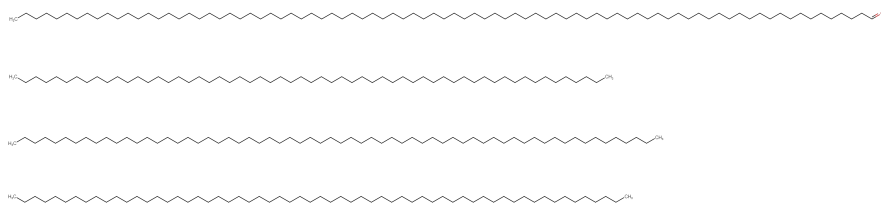


Figure 3: High resolution version of Figure 2(b).

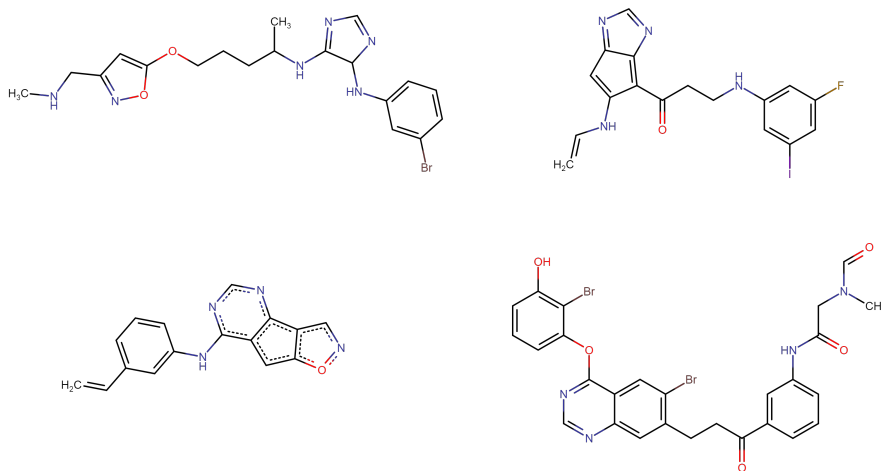


Figure 4: High resolution version of Figure 2(c).

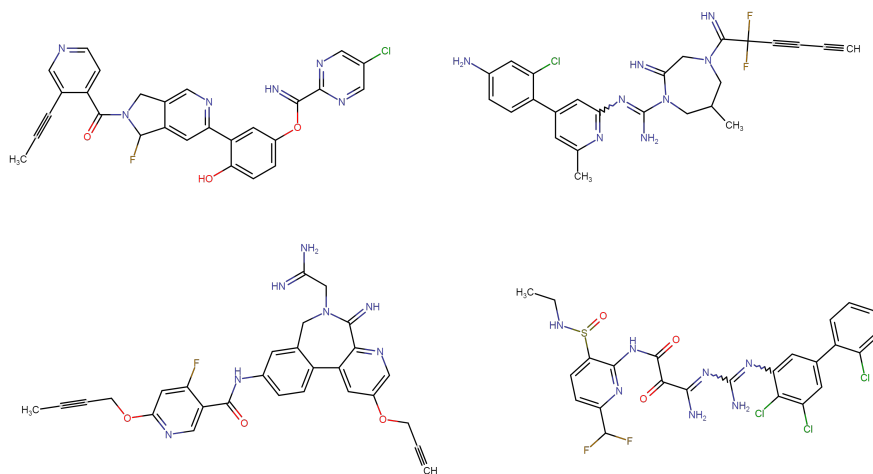


Figure 5: High resolution version of Figure 2(d).

Table 1: Results of the GuacaMol benchmark for baseline methods and our proposed method

Benchmark	Best of Data Set	SMILES GA	Graph MCTS	Graph GA	SMILES LSTM	MSO (ours)
Celecoxib rediscovery	0.505	0.732	0.355	1.000	1.000	1.000
Troglitazone rediscovery	0.419	0.515	0.311	1.000	1.000	1.000
Thiothixene rediscovery	0.456	0.598	0.311	1.000	1.000	1.000
Aripiprazole similarity	0.595	0.834	0.380	1.000	1.000	1.000
Albuterol similarity	0.719	0.907	0.749	1.000	1.000	1.000
Mestranol similarity	0.629	0.790	0.402	1.000	1.000	1.000
C11H24	0.684	0.829	0.410	0.971	0.993	0.997
C9H10N2O2PF2Cl	0.747	0.889	0.631	0.982	0.879	1.000
Median molecules 1	0.334	0.334	0.225	0.406	0.438	0.437
Median molecules 2	0.351	0.380	0.170	0.432	0.422	0.395
Osimertinib MPO	0.839	0.886	0.784	0.953	0.907	0.966
Fexofenadine MPO	0.817	0.931	0.695	0.998	0.959	1.000
Ranolazine MPO	0.792	0.881	0.616	0.920	0.855	0.931
Perindopril MPO	0.575	0.661	0.385	0.792	0.808	0.834
Amlodipine MPO	0.696	0.722	0.533	0.894	0.894	0.900
Sitagliptin MPO	0.509	0.689	0.458	0.891	0.545	0.868
Zaleplon MPO	0.547	0.413	0.488	0.754	0.669	0.764
Valsartan SMARTS	0.259	0.552	0.040	0.990	0.978	0.994
Scaffold Hop	0.933	0.970	0.590	1.000	0.996	1.000
Deco Hop	0.738	0.885	0.478	1.000	0.998	1.000
Total	12.144	14.396	9.009	17.983	17.340	18.086

References

- [1] D. J. Cummins and M. A. Bell, *Journal of medicinal chemistry*, 2016, **59**, 6999–7010.

2.3 Publication 3: grünifai: Interactive Multiparameter Optimization of Molecules in a Continuous Vector Space

Full Reference: *Winter, Robin, Retel, Joren, Noé, Frank, Clevert, Djork-Arné & Steffen, Andreas (2020). grünifai: interactive multiparameter optimization of molecules in a continuous vector space. Bioinformatics, 36(13), 4093-4094.*

DOI: 10.1093/bioinformatics/btaa271

Journal/Conference: Bioinformatics (Impact Factor: 6.9)

Source Code: <https://github.com/jrwnter/gruenifai>

Paper's main contributions:

- We develop a web application for interactive molecular optimization.
- We utilize the continuous molecular representation and molecular optimization algorithm proposed in our previous works and make them accessible to a broader user group through a web application.
- We implement various ways for the user to actively steer the optimization process.

Author's contribution to the paper:

- Conceptualization of the original idea.
- Design of tools for active steering of the optimization.
- Implementation of the web application's backend.
- Preparation and creation of initial draft and visualizations.

2.4 Publication 4: Auto-Encoding Molecular Conformations

Full Reference: *Winter, Robin, Noé, Frank & Clevert, Djork-Arné (2021). Auto-encoding molecular conformations. arXiv preprint arXiv:2101.01618*

DOI: 10.48550/arXiv.2101.01618

Licence: CC-BY

Journal/Conference: Machine Learning for Molecules Workshop at NeurIPS 2020

Source Code: NA

Paper's main contributions:

- We propose a novel method for representing molecular conformations in a continuous and fixed-size space and generation of novel conformations from molecular topology only.
- We demonstrate how different conformations can be represented in a continuous space, where similar conformations cluster together.
- We show how our proposed method can be used to sample energetically reasonable conformations for a given molecular topology.

Author's contribution to the paper:

- Conceptualization of the original idea and its application to molecular conformation representation and generation.
- Development of the methodology and implementation.
- Design and evaluation of experiments, data curation and analysis.
- Preparation and creation of initial draft and visualizations.

Auto-Encoding Molecular Conformations

Robin Winter

Machine Learning Research
Bayer AG
Berlin
robin.winter[at]bayer.com

Frank Noé

Mathematics and Computer Science
Freie Universität Berlin
Berlin

Djork-Arné Clevert

Machine Learning Research
Bayer AG
Berlin
djork-arne.clevert[at]bayer.com

Abstract

In this work we introduce an Autoencoder for molecular conformations. Our proposed model converts the discrete spatial arrangements of atoms in a given molecular graph (conformation) into and from a continuous fixed-sized latent representation. We demonstrate that in this latent representation, similar conformations cluster together while distinct conformations split apart. Moreover, by training a probabilistic model on a large dataset of molecular conformations, we demonstrate how our model can be used to generate diverse sets of energetically favorable conformations for a given molecule. Finally, we show that the continuous representation allows us to utilize optimization methods to find molecules that have conformations with favourable spatial properties.

1 Introduction

Representing chemical matter in a meaningful and expressive way plays a crucial role when it comes to computer-aided modeling in the field of chemistry (Todeschini and Consonni, 2009). Recently, substantial progress has been made in many molecule-related tasks, such as bio- and physicochemical property prediction (Montanari et al., 2020), inverse design of desirable molecules (Gómez-Bombarelli et al., 2018; Winter et al., 2019; Le et al., 2020), retrosynthetic analysis and synthesis planning (Segler et al., 2018). Most of these advancements can be attributed to the use of deep neural networks that enable representation learning of chemical matter. While traditional methods are mostly based on features, generated by rule-based algorithms extracting structural information (e.g. extended-connectivity fingerprints), these novel methods are directly trained on a discrete but more comprehensive representation of molecules. In particular, Graph Neural Networks utilizing the molecular graph with atoms as nodes and bonds as edges (Duvenaud et al., 2015) or Recurrent Neural Networks utilizing the one-dimensional line notation of molecules, namely SMILES (Segler et al., 2018). Still, the underlying molecular representation of the aforementioned methods are limited in the sense that they do not reflect the spatial arrangement of the atoms in the molecule. However, many interesting molecular properties, such as its ability to fit inside a protein binding pocket, inducing a pharmacological effect, are driven by its possible (energetically stable) spatial arrangements (conformations). Recently, work has been done to apply neural networks directly on specific conformations of molecules to predict properties such as the equilibrium energy or the *HOMO-LUMO* gap (Schütt et al., 2018). Moreover, models have been proposed to generate molecular conformations for a given molecular graph (Mansimov et al., 2019; Simm and Hernández-Lobato,

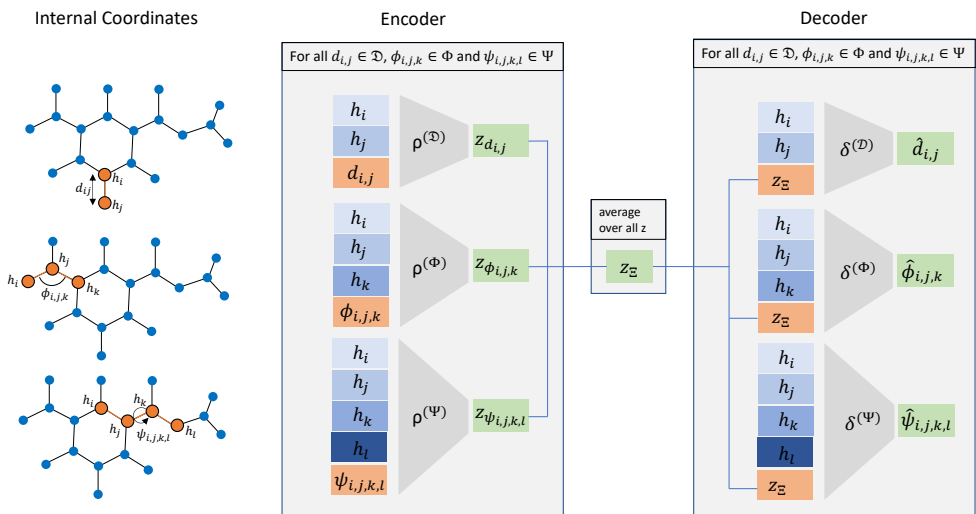


Figure 1: Model architecture of the conformation encoder (middle) and decoder (right). The encoding functions $\rho^{(\mathcal{D})}$, $\rho^{(\Phi)}$ and $\rho^{(\Psi)}$ encode their respective internal coordinates into a latent representation. The decoding functions $\delta^{(\mathcal{D})}$, $\delta^{(\Phi)}$ and $\delta^{(\Psi)}$ are conditioned on the averaged latent representations (conformer embedding) to reconstruct their respective internal coordinates, given a set of node embeddings $h_i \in \mathcal{H}$. On the left hand side, the definition of the internal coordinates, bond length $d_{i,j} \in \mathcal{D}$, bond angle $\phi_{i,j,k} \in \Phi$ and dihedral angle $\psi_{i,j,k,l} \in \Psi$, is visualized.

2019) or chemical formula (Hoffmann and Noé, 2019).

In this work we propose a novel model that converts a molecular conformation to and from a fixed-sized latent representation (conformation embedding), independent of the number of atoms and bonds of a molecule. Moreover, training the model in a probabilistic setting, we show that we can model the conformational distributions of molecules. We demonstrate that sampling from this model results into a diverse set of energetically reasonable conformers. Finally, combining the conformation embedding with a continuous molecular structure embedding, we demonstrate how molecules can be optimized with respect to spatial properties.

2 Methods

2.1 Representing Molecular Conformations

The three-dimensional arrangement of atoms of a molecule can be represented in many different ways. Annotating each atom with a Cartesian coordinate is probably the most straight-forward way, however, does not reflect a molecules invariance to rigid translations and rotations. In this work we utilize the *internal coordinate representation*, also known as *Z-matrix*. In this notation, a molecules spatial arrangement (conformation) Ξ is defined by the set of distances $\mathcal{D} = \{d_1, \dots, d_{N_{\mathcal{D}}}\}$ between bonded atoms (bond length), the angles $\Phi = \{\phi_1, \dots, \phi_{N_{\Phi}}\}$ of three connected atoms (bond angles) and the torsion angles (dihedral angles) $\Psi = \{\psi_1, \dots, \psi_{N_{\Psi}}\}$ of three consecutive bonds (see Figure 1). This representation is invariant to rotations and rigid translations and can always be transformed to and from Cartesian coordinates.

2.2 Conformation Autoencoder

The overall goal of the proposed model is to find functions f_{Θ} and g_{Θ} that map a conformation $\Xi_{\mathcal{G}}$ of a molecule \mathcal{G} to and from a fixed-sized latent representation $z_{\Xi} \in \mathbb{R}^{F_z}$, respectively. This introduces two major challenges. Firstly, the model has to map molecules with a different number of atoms and bonds to the same fixed-sized space. Secondly, f_{Θ} has to invariant with respect to the ordering of

atoms in the input. Our proposed model consist of a conformation-independent and a conformation-dependent part. The conformation-independent part comprises a Graph Neural Network utilizing the molecular graph to extract node-level features for a given molecule. The conformation-dependent part utilizes these extracted node-level features either to encode the internal coordinates of a specific molecular conformation into a latent representation (conformation embedding) or to reconstruct a conformation by predicting the internal coordinates of sets of connected atoms, given their respective node features and a conformation embedding. The whole model is trained on minimizing the reconstruction error of the internal coordinates Ξ for a given molecule:

$$\mathcal{C}_{\Xi} = \frac{1}{N_{\mathcal{D}}} \sum_{d \in \mathcal{D}} \|d, \hat{d}\|_2^2 + \frac{1}{N_{\Phi}} \sum_{\phi \in \Phi} \|\phi, \hat{\phi}\|_2^2 + \frac{1}{N_{\Psi}} \sum_{\psi \in \Psi} \min(\|\psi, \hat{\psi}\|_2^2, 2\pi - \|\psi, \hat{\psi}\|_2^2), \quad (1)$$

where the last term accounts for the periodicity of the dihedral angle. Our proposed model can easily be extended to a probabilistic generative model by employing the ideas from Kingma and Welling (2013), effectively defining the model as a variational auto encoder.

2.2.1 Molecular Graph Encoder

In this work we define the conformation-independent molecular graph as an undirected Graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, with vertices or nodes $v_i \in \mathcal{V}$ and edges $e_{ij} = (v_i, v_j) \in \mathcal{E}$ connecting v_i and v_j . Where nodes $v_i \in \mathbb{R}^{F_v}$ represent atoms and encode atom features such as element type and charge. Edges $e_{ij} \in \mathbb{R}^{F_e}$ represent bonds between atoms and encode the bond type (i.e. single, double, triple or aromatic bond).

We utilize a Graph Neural Network (GNN) to extract a node-level representation of a molecular graph. Given a molecular graph with initial node and edge features defined by the atoms and bonds of the molecule, a GNN iteratively updates node embeddings by aggregating localized information of connected nodes respectively.

In particular, we use the Graph Attention Network (GAT) (Veličković et al., 2017) framework which updates the node embeddings \mathbf{h}_i by the following rule:

$$\mathbf{h}'_i = \alpha_{i,j} \Theta \mathbf{h}_i + \sum_{j \in \mathcal{N}(i)} \alpha_{i,j} \Theta \mathbf{h}_j, \quad (2)$$

with neighbouring node indices $\mathcal{N}(i) = \{j \in (0, \dots, N) | \forall j((v_i, v_j) \in \mathcal{E})\}$. The attention coefficients $\alpha_{i,j}$ are computed as

$$\alpha_{i,j} = \frac{\exp(\sigma(\mathbf{a}^T [\Theta \mathbf{h}_i \| \Theta \mathbf{h}_j]))}{\sum_{k \in \mathcal{N}(i) \cup \{i\}} \exp(\sigma(\mathbf{a}^T [\Theta \mathbf{h}_i \| \Theta \mathbf{h}_k]))}, \quad (3)$$

where \cdot^T is the transposition and $\|$ the concatenation operation. The attention function a is defined as a single-layer feedforward neural network.

To incorporate edge attributes $e_{i,j}$ (bond-type information) in the model we also utilize the so-called edge-conditioned graph convolution (EConv) layer (Simonovsky and Komodakis, 2017), defined by the following update rule:

$$\mathbf{h}'_i = \Theta \mathbf{h}_i + \sum_{j \in \mathcal{N}(i)} \mathbf{h}_j \cdot \mathbf{f}_{\Theta}(e_{i,j}), \quad (4)$$

where \mathbf{f}_{Θ} denotes a multi layer perceptron.

In particular, we utilized edge-conditioned graph convolution (EConv) (Simonovsky and Komodakis, 2017) and Graph Attention Network (GAT) (Veličković et al., 2017) layers. To aggregate information about higher-order neighbours, we combine one EConv (to encode edge information) with multiple consecutive GAT layers:

$$\mathbf{h}_i^l = \text{GAT}^{l-1} \circ \dots \circ \text{GAT}^1 \circ \text{EConv}(\mathbf{h}_i^0). \quad (5)$$

where $\mathbf{h}_i^0 = v_i \in \mathbb{R}^{F_v}$ are the atom features of the input molecular graph.

2.2.2 Conformation Encoder

To this end, we define the molecular conformation representation learning task as a learning task on *sets*. In particular, our proposed model learns to extract a latent representation z_{Ξ} of a set of internal coordinates Ξ for a given molecule:

$$z_{\Xi} = f_{\Theta}(\mathcal{H}, \Xi) = f_{\Theta}(\mathcal{H}, \mathcal{D}, \Phi, \Psi), \quad (6)$$

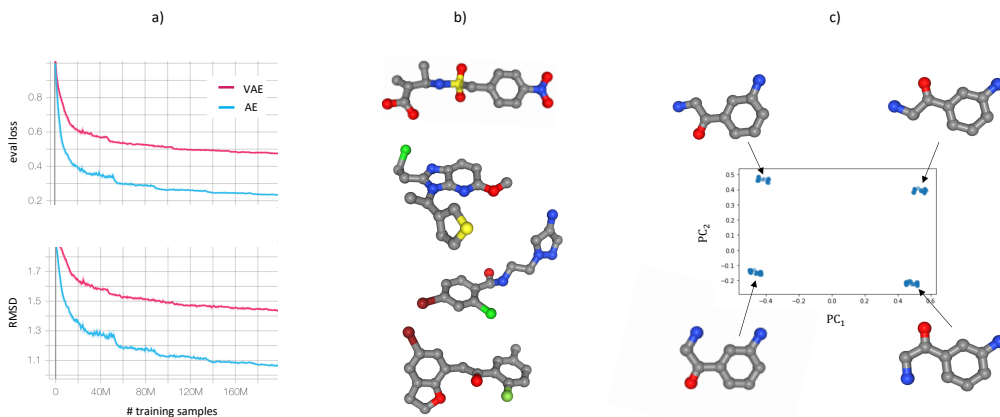


Figure 2: a) Learning curves for the autoencoder (AE) and variational autoencoder (VAE) during training, with evaluation loss as defined in (1) and root means squared deviation (RMSD) between predicted and input conformations on a holdout set. b) Four example conformations generated by our proposed model. c) First two principle components of the latent representation (conformation embedding) of 200 conformations with a corresponding representative conformation for each cluster.

with the permutation invariant function f_{Θ} , parameterized by a neural network and conditioned on the node embeddings $\mathcal{H} = \{h_1, \dots, h_N\}$, extracted by the molecular graph encoder defined in the previous section. Following Zaheer et al. (2017), we define the permutation invariant function f_{Θ} as

$$\begin{aligned}
 z_{\Xi} &= f_{\Theta}(\mathcal{H}, \Xi) = \sigma \left(\sum_{\xi \in \Xi} \rho(\mathcal{H}, \xi) \right) = \frac{1}{N_{\Xi}} \sum_{\xi \in \Xi} \rho_{\Theta}(\mathcal{H}, \xi) \\
 &= \frac{1}{N_{\mathcal{D}} + N_{\Phi} + N_{\Psi}} \left(\sum_{d \in \mathcal{D}} \rho_{\Theta}^{(\mathcal{D})}(\mathcal{H}, d) + \sum_{\phi \in \Phi} \rho_{\Theta}^{(\Phi)}(\mathcal{H}, \phi) + \sum_{\psi \in \Psi} \rho_{\Theta}^{(\Psi)}(\mathcal{H}, \psi) \right).
 \end{aligned} \tag{7}$$

The functions $\rho_{\Theta}^{(\mathcal{D})}$, $\rho_{\Theta}^{(\Phi)}$ and $\rho_{\Theta}^{(\Psi)}$ are defined as feed-forward neural networks that take bond lengths, bond angles and dihedral angles as input respectively. Additionally, to put an internal coordinate into context of the graph, ρ_{Θ} is conditioned on corresponding node embeddings \mathcal{H} as well. This means, if $\rho_{\Theta}^{(\Phi)}$ encodes the angle $\phi_{i,j,k}$ between atoms v_i and v_k connected by atom v_j , function $\rho_{\Theta}^{(\Phi)}$ takes also node embeddings h_i , h_j and h_k as argument (see Figure 1). Most importantly, equation (7) is invariant to the order of internal coordinates and node indices and the dimensionality of the resulting z_{Ξ} is invariant of the size of the input molecule.

2.2.3 Conformation Decoder

To reconstruct a molecular conformation back from its latent representation, we train three additional neural networks $\delta_{\Theta}^{(\mathcal{D})}$, $\delta_{\Theta}^{(\Phi)}$ and $\delta_{\Theta}^{(\Psi)}$ for each type of internal coordinate respectively (see Figure 1). Each neural network is conditioned on the conformation embedding and takes the node embeddings of the corresponding internal coordinate as input. For example, to predict the bond length $d_{i,j}$ between atoms v_i and v_j , the network takes h_i , h_j and z_{Ξ} as input.

3 Results and Discussion

We trained our model on the public PubChem3D dataset (Bolton et al., 2011), which comprises molecules (organic, up to 50 heavy atoms) with multiple conformations generated by the forcefield software OMEGA (Hawkins et al., 2010). Upon convergence, our model is able to predict internal coordinates for a given molecule that result into conformations that are similar (with respect to the RMSD) to the input conformations (see Figure 2). To quantitatively analyze how energetically reasonable the reconstructed conformations are, we calculated their internal energy with the MMFF94 forcefield (Halgren, 1996) as implemented in the Python package RDKit (Landrum et al., 2006).

The median energetic difference between the input and reconstructed conformation is approximately 80 kcal/mol, which corresponds to small deviations from local minimas, without e.g. clashes of atoms (see example molecules in Figure 2). Moreover, since the model does not only reconstruct any possible conformation for a molecule but is trained on reconstructing a specific input conformation, differences between these conformations have to be encoded in the latent representation. On the right side of Figure 2, we show this for a simple example of a small molecule in four different conformations.

As described in Section 2.2, we can easily extend the proposed model to a variational autoencoder, which can be used to sample conformations from the learned distribution. A major challenge in conformation generation is to efficiently sample diverse conformers. Therefore, we analyzed the average interconformer RMSD (icRMSD) for a set of 200 sampled conformers per molecule for the holdout set. Comparing the icRMSD of our proposed model with a state-of-the-art conformation generation algorithm ETKDG (Riniker and Landrum, 2015) as implemented in RDKit, we see a similar performance, with our model having a slightly higher average icRMSD of 0.07 Å.

Since the proposed model gives means to directly infer conformations for a given molecule, it is possible to optimize molecules in the continuous conformation embedding with respect to spatial properties. When combined with a latent representation of the molecular structure (Winter et al., 2019), optimization of molecules can even be performed with respect to both the molecular graph and its conformation. As a proof of principle, we optimized molecules with respect to a combination of the conformation-independent *quantitative estimate of drug-likeness* (QED) score (values between 0 and 1) (Bickerton et al., 2012) and the conformation-dependent property *asphericity* (Todeschini and Consonni, 2009) (values between 0 and 1), which quantifies a molecules deviation from a spherical shape. We utilized the genetic *Particle Swarm Optimization* algorithm (Kennedy and Eberhart, 1995), to optimize both latent representations at the same time. Starting from the already drug-like molecule aspirin with a combined score of 0.76, we could already find after 50 iterations molecules with a score of 1.82. In general this method could also be used to optimize molecules for other interesting spatial properties, such as fitting pharmacophores or the shape of known bio active molecule.

References

- Todeschini, R.; Consonni, V. *Molecular descriptors for chemoinformatics: volume I: alphabetical listing/volume II: appendices, references*; John Wiley & Sons, 2009; Vol. 41.
- Montanari, F.; Kuhnke, L.; Ter Laak, A.; Clevert, D.-A. Modeling physico-chemical admet endpoints with multitask graph convolutional networks. *Molecules* **2020**, *25*, 44.
- Gómez-Bombarelli, R.; Wei, J. N.; Duvenaud, D.; Hernández-Lobato, J. M.; Sánchez-Lengeling, B.; Sheberla, D.; Aguilera-Iparraguirre, J.; Hirzel, T. D.; Adams, R. P.; Aspuru-Guzik, A. Automatic chemical design using a data-driven continuous representation of molecules. *ACS central science* **2018**, *4*, 268–276.
- Winter, R.; Montanari, F.; Steffen, A.; Briem, H.; Noé, F.; Clevert, D.-A. Efficient multi-objective molecular optimization in a continuous latent space. *Chemical science* **2019**, *10*, 8016–8024.
- Le, T.; Winter, R.; Noé, F.; Clevert, D.-A. Neuraldecipher – reverse-engineering extended-connectivity fingerprints (ECFPs) to their molecular structures. *Chem. Sci.* **2020**, –.
- Segler, M. H. S.; Preuss, M.; Waller, M. P. Planning chemical syntheses with deep neural networks and symbolic AI. **2018**, *555*, 604–610.
- Duvenaud, D. K.; Maclaurin, D.; Iparraguirre, J.; Bombarell, R.; Hirzel, T.; Aspuru-Guzik, A.; Adams, R. P. Convolutional networks on graphs for learning molecular fingerprints. *Advances in neural information processing systems*. 2015; pp 2224–2232.
- Segler, M. H.; Kogej, T.; Tyrchan, C.; Waller, M. P. Generating focused molecule libraries for drug discovery with recurrent neural networks. *ACS central science* **2018**, *4*, 120–131.
- Schütt, K. T.; Sauceda, H. E.; Kindermans, P.-J.; Tkatchenko, A.; Müller, K.-R. SchNet–A deep learning architecture for molecules and materials. *The Journal of Chemical Physics* **2018**, *148*, 241722.

- Mansimov, E.; Mahmood, O.; Kang, S.; Cho, K. Molecular geometry prediction using a deep generative graph neural network. *Scientific reports* **2019**, *9*, 1–13.
- Simm, G. N.; Hernández-Lobato, J. M. A generative model for molecular distance geometry. *arXiv preprint arXiv:1909.11459* **2019**,
- Hoffmann, M.; Noé, F. Generating valid Euclidean distance matrices. *arXiv preprint arXiv:1910.03131* **2019**,
- Kingma, D. P.; Welling, M. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114* **2013**,
- Veličković, P.; Cucurull, G.; Casanova, A.; Romero, A.; Lio, P.; Bengio, Y. Graph attention networks. *arXiv preprint arXiv:1710.10903* **2017**,
- Simonovsky, M.; Komodakis, N. Dynamic edge-conditioned filters in convolutional neural networks on graphs. Proceedings of the IEEE conference on computer vision and pattern recognition. 2017; pp 3693–3702.
- Zaheer, M.; Kottur, S.; Ravanbakhsh, S.; Poczos, B.; Salakhutdinov, R. R.; Smola, A. J. Deep sets. Advances in neural information processing systems. 2017; pp 3391–3401.
- Bolton, E. E.; Chen, J.; Kim, S.; Han, L.; He, S.; Shi, W.; Simonyan, V.; Sun, Y.; Thiessen, P. A.; Wang, J., et al. PubChem3D: a new resource for scientists. *Journal of cheminformatics* **2011**, *3*, 32.
- Hawkins, P. C.; Skillman, A. G.; Warren, G. L.; Ellingson, B. A.; Stahl, M. T. Conformer generation with OMEGA: algorithm and validation using high quality structures from the Protein Databank and Cambridge Structural Database. *Journal of chemical information and modeling* **2010**, *50*, 572–584.
- Halgren, T. A. Merck molecular force field. I. Basis, form, scope, parameterization, and performance of MMFF94. *Journal of computational chemistry* **1996**, *17*, 490–519.
- Landrum, G., et al. RDKit: Open-source cheminformatics. **2006**,
- Riniker, S.; Landrum, G. A. Better informed distance geometry: using what we know to improve conformation generation. *Journal of chemical information and modeling* **2015**, *55*, 2562–2574.
- Winter, R.; Montanari, F.; Noé, F.; Clevert, D.-A. Learning continuous and data-driven molecular descriptors by translating equivalent chemical representations. *Chemical science* **2019**, *10*, 1692–1701.
- Bickerton, G. R.; Paolini, G. V.; Besnard, J.; Muresan, S.; Hopkins, A. L. Quantifying the chemical beauty of drugs. *Nature chemistry* **2012**, *4*, 90–98.
- Kennedy, J.; Eberhart, R. Particle swarm optimization. Proceedings of ICNN'95-International Conference on Neural Networks. 1995; pp 1942–1948.

2.5 Publication 5: Permutation-Invariant Variational Autoencoder for Graph-Level Representation Learning

Full Reference: *Winter, Robin, Noé, Frank & Clevert, Djork-Arné (2021). Permutation-invariant variational autoencoder for graph-level representation learning. Advances in Neural Information Processing Systems, 34.*

DOI: 10.48550/arXiv.2104.09856

Licence: CC-BY

Journal/Conference: Neural Information Processing Systems (NeurIPS)
(h5-index: 309)

Source Code: <https://github.com/jrwnter/pigvae>

Paper's main contributions:

- We propose a novel method for representing graphs on a graph-level in a permutation invariant way.
- We discuss why permutation invariant graph-level representations are desirable and demonstrate their superiority over alternatives.
- We benchmark our proposed method in a variety of graph representation, classification, regression, generation and interpolation tasks.

Author's contribution to the paper:

- Conceptualization of the original idea and its application to graph representation learning.
- Development of the methodology and implementation.
- Design and evaluation of experiments, data curation and analysis.
- Preparation and creation of initial draft and visualizations.

Permutation-Invariant Variational Autoencoder for Graph-Level Representation Learning

Robin Winter
Bayer AG
Freie Universität Berlin
robin.winter@bayer.com

Frank Noé
Freie Universität Berlin
frank.noe@fu-berlin.de

Djork-Arné Clevert
Bayer AG
djork-arne.clevert@bayer.com

Abstract

Recently, there has been great success in applying deep neural networks on graph structured data. Most work, however, focuses on either node- or graph-level supervised learning, such as node, link or graph classification or node-level unsupervised learning (e.g., node clustering). Despite its wide range of possible applications, graph-level unsupervised representation learning has not received much attention yet. This might be mainly attributed to the high representation complexity of graphs, which can be represented by $n!$ equivalent adjacency matrices, where n is the number of nodes. In this work we address this issue by proposing a permutation-invariant variational autoencoder for graph structured data. Our proposed model indirectly learns to match the node order of input and output graph, without imposing a particular node order or performing expensive graph matching. We demonstrate the effectiveness of our proposed model for graph reconstruction, generation and interpolation and evaluate the expressive power of extracted representations for downstream graph-level classification and regression.

1 Introduction

Graphs are an universal data structure that can be used to describe a vast variety of systems from social networks to quantum mechanics [1]. Driven by the success of Deep Learning in fields such as Computer Vision and Natural Language Processing, there has been an increasing interest in applying deep neural networks on non-Euclidean, graph structured data as well [2, 3]. Most notably, generalizing Convolutional Neural Networks and Recurrent Neural Networks to arbitrarily structured graphs for supervised learning has lead to significant advances on task such as molecular property prediction [4] or question-answering [5]. Research on unsupervised learning on graphs mainly focused on node-level representation learning, which aims at embedding the local graph structure into latent node representations [6, 7, 8, 9, 10]. Usually, this is achieved by adopting an autoencoder framework where the encoder utilizes e.g., graph convolutional layers to aggregate local information at a node level and the decoder is used to reconstruct the graph structure from the node embeddings. Graph-level representations are usually extracted by aggregating node-level features into a single vector, which is common practice in supervised learning on graph-level labels [4].

Unsupervised learning of graph-level representations, however, has not yet received much attention, despite its wide range of possible applications, such as feature extraction, pre-training for graph-level classification/regression tasks, graph matching or similarity ranking. This might be mainly attributed to the high representation complexity of graphs arising from their inherent invariance with respect

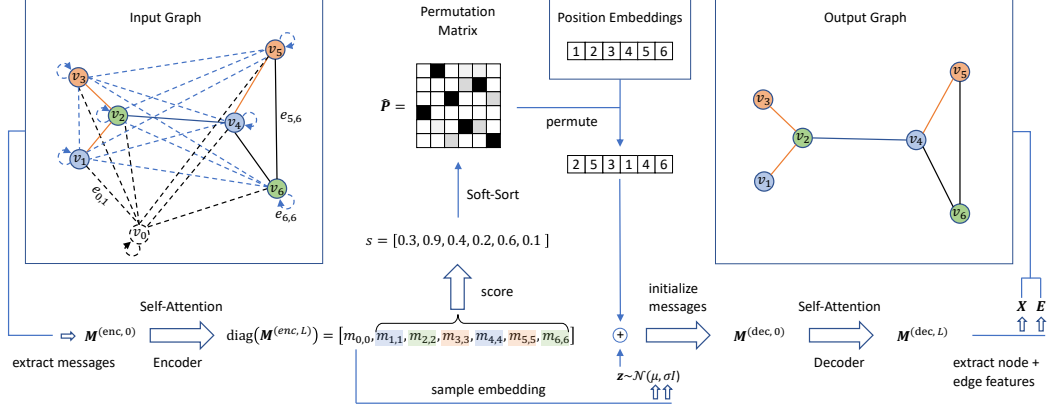


Figure 1: Network architecture of the proposed model. Input graph is depicted as fully connected graph (dashed lines for not direct neighbours) with the additional embedding node v_0 and edges to it (black color). Different node and edge types are represented by different colors and real edges by solid lines between nodes. Transformations parameterized by a neural network are represented by block arrows.

to the order of nodes within the graph. In general, a graph with n nodes, can be represented by $n!$ equivalent adjacency matrices, each corresponding to a different node order. Since the general structure of a graph is invariant to the order of their individual nodes, a graph-level representation should not depend on the order of the nodes in the input representation of a graph, i.e. two isomorphic graphs should always be mapped to the same representation. This poses a problem for most neural network architectures which are by design not invariant to the order of their inputs. Even if carefully designed in a permutation invariant way (e.g., Graph Neural Networks with a final node aggregation step), there is no straight-forward way to train an autoencoder network, due to the ambiguous reconstruction objective, requiring the same discrete order of input and output graphs to compute the reconstruction loss.

How can we learn a permutation-invariant graph-level representation utilizing a permutation-variant reconstruction objective? In this work we tackle this question proposing a graph autoencoder architecture that is by design invariant to the order of nodes in a graph. We address the order ambiguity issue by training alongside the encoder and decoder model an additional *permuter* model that assigns to each input graph a permutation matrix to align the input graph node order with the node order of the reconstructed graph.

2 Method

2.1 Notations and Problem Definition

An undirected Graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is defined by the set of n nodes $\mathcal{V} = \{v_1, \dots, v_n\}$ and edges $\mathcal{E} = \{(v_i, v_j) | v_i, v_j \in \mathcal{V}\}$. We can represent a graph in matrix form by its node features $\mathbf{X}_\pi \in \mathbb{R}^{n \times d_v}$ and adjacency matrix $\mathbf{A}_\pi \in \{0, 1\}^{n \times n}$ in the node order $\pi \in \Pi$, where Π is the set of all $n!$ permutations over \mathcal{V} . We define the permutation matrix \mathbf{P} that reorders nodes from order π to order π' as $\mathbf{P}_{\pi \rightarrow \pi'} = (p_{ij}) \in \{0, 1\}^{n \times n}$, with $p_{ij} = 1$ if $\pi(i) = \pi'(j)$ and $p_{ij} = 0$ everywhere else. Since Graphs are invariant to the order of their nodes, note that

$$\mathcal{G}_\pi = \mathcal{G}(\mathbf{X}_\pi, \mathbf{A}_\pi) = \mathcal{G}(\mathbf{P}_{\pi \rightarrow \pi'} \mathbf{X}_\pi, \mathbf{P}_{\pi \rightarrow \pi'} \mathbf{A}_\pi \mathbf{P}_{\pi \rightarrow \pi'}^\top) = \mathcal{G}(\mathbf{X}_{\pi'}, \mathbf{A}_{\pi'}) = \mathcal{G}_{\pi'}, \quad (1)$$

where \top is the transpose operator. Let us now consider a dataset of graphs $\mathbf{G} = \{\mathcal{G}^{(i)}\}_{i=0}^N$ we would like to be represented in a low-dimensional continuous space. We can adopt a latent variable approach and assume that the data is generated by a process $p_\theta(\mathcal{G} | \mathbf{z})$, involving an unobserved continuous random variable \mathbf{z} . Following the work of Kingma and Welling [11], we approximate the intractable posterior by $q_\phi(\mathcal{G} | \mathbf{z}) \approx p_\theta(\mathcal{G} | \mathbf{z})$ and minimize the lower bound on the marginal likelihood of graph $\mathcal{G}^{(i)}$:

$$\log p_\theta(\mathcal{G}^{(i)}) \geq \mathcal{L}(\phi, \theta; \mathcal{G}^{(i)}) = -\text{KL} \left[q_\phi(\mathbf{z} | \mathcal{G}^{(i)}) || p_\theta(\mathbf{z}) \right] + \mathbb{E}_{q_\phi(\mathbf{z} | \mathcal{G}^{(i)})} \left[\log p_\theta(\mathcal{G}^{(i)} | \mathbf{z}) \right], \quad (2)$$

where the Kullback–Leibler (KL) divergence term regularizes the encoded latent codes of graphs $\mathcal{G}^{(i)}$ and the second term enforces high similarity of decoded graphs to their encoded counterparts. As graphs can be completely described in matrix form by their node features and adjacency matrix, we can parameterize q_ϕ and p_θ in Eq. (2) by neural networks that encode and decode node features $\mathbf{X}_\pi^{(i)}$ and adjacency matrices $\mathbf{A}_\pi^{(i)}$ of graphs $\mathcal{G}_\pi^{(i)}$. However, as graphs are invariant under arbitrary node re-ordering, the latent code \mathbf{z} should be invariant to the node order π :

$$q_\phi(\mathbf{z}|\mathcal{G}_\pi) = q_\phi(\mathbf{z}|\mathcal{G}_{\pi'}), \text{ for all } \pi, \pi' \in \Pi. \quad (3)$$

This can be achieved by parameterizing the encoder model q_ϕ by a permutation invariant function. However, if the latent code \mathbf{z} does not encode the input node order π , input graph \mathcal{G}_π and decoded graph $\hat{\mathcal{G}}_{\pi'}$ are no longer necessarily in the same order, as the decoder model has no information about the node order of the input graph. Hence, the second term in Eq. (2) cannot be optimized anymore by minimizing the reconstruction loss between encoded graph \mathcal{G}_π and decoded graph $\hat{\mathcal{G}}_{\pi'}$ in a straight-forward way. They need to be brought in the same node order first. We can rewrite the expectation in Eq. (2) using Eq. (1):

$$\mathbb{E}_{q_\phi(\mathbf{z}|\mathcal{G}_\pi^{(i)})} \left[\log p_\theta(\mathcal{G}_{\pi'}^{(i)}|\mathbf{z}) \right] = \mathbb{E}_{q_\phi(\mathbf{z}|\mathcal{G}_\pi^{(i)})} \left[\log p_\theta(\hat{\mathbf{P}}_{\pi \rightarrow \pi'} \mathcal{G}_\pi^{(i)}|\mathbf{z}) \right]. \quad (4)$$

Since the ordering of the decoded graph π' is subject to the learning process of the decoder and thus unknown in advance, finding $\hat{\mathbf{P}}_{\pi \rightarrow \pi'}$ is not trivial. In [12], the authors propose to use approximate graph matching to find the permutation matrix $\mathbf{P}_{\pi \rightarrow \pi'}$ that maximizes the similarity $s(X_{\pi'}, \mathbf{P}_{\pi \rightarrow \pi'} \hat{X}_\pi; A_{\pi'}, \mathbf{P}_{\pi \rightarrow \pi'} \hat{A}_\pi \mathbf{P}_{\pi \rightarrow \pi'}^\top)$, which involves up to $O(n^4)$ re-ordering operations at each training step in the worst case [13].

2.2 Permutation-Invariant Variational Graph Autoencoder

In this work we propose to solve the reordering problem in Eq. (4) implicitly by inferring the permutation matrix $\mathbf{P}_{\pi' \rightarrow \pi}$ from the input graph \mathcal{G}_π by a model $g_\psi(\mathcal{G}_\pi)$ that is trained to bring input and output graph in the same node order and is used by the decoder model to permute the output graph. We train this *permuter* model jointly with the *encoder* model $q_\phi(\mathbf{z}|\mathcal{G}_\pi)$ and *decoder* model $p_\theta(\mathcal{G}_\pi|\mathbf{z}, \mathbf{P}_{\pi' \rightarrow \pi})$, optimizing:

$$\mathcal{L}(\phi, \theta, \psi; \mathcal{G}_\pi^{(i)}) = -\text{KL} \left[q_\phi(\mathbf{z}|\mathcal{G}_\pi^{(i)}) || p_\theta(\mathbf{z}) \right] + \mathbb{E}_{q_\phi(\mathbf{z}|\mathcal{G}_\pi^{(i)})} \left[\log p_\theta(\mathcal{G}_\pi^{(i)}|\mathbf{z}, g_\psi(\mathcal{G}_\pi^{(i)})) \right]. \quad (5)$$

Intuitively, the permuter model has to learn how the ordering of nodes in the graph generated by the decoder model will differ from a specific node order present in the input graph. During the learning process, the decoder will learn its own canonical ordering that, given a latent code z , it will always reconstruct a graph in. The permuter learns to transform/permute this canonical order to a given input node order. For this, the permuter predicts for each node i of the input graph a score s_i corresponding to its probability to have a low node index in the decoded graph. By sorting the input nodes indices by their assigned scores we can infer the output node order and construct the respective permutation matrix $\mathbf{P}_{\pi \rightarrow \pi'} = (p_{ij}) \in \{0, 1\}^{n \times n}$, with

$$p_{ij} = \begin{cases} 1, & \text{if } j = \text{argsort}(s)_i \\ 0, & \text{else} \end{cases} \quad (6)$$

to align input and output node order. Since the argsort operation is not differentiable, we utilize the continuous relaxation of the argsort operator proposed in [14, 15]:

$$\mathbf{P} \approx \hat{\mathbf{P}} = \text{softmax} \left(\frac{-d(\text{sort}(s) \mathbf{1}^\top, \mathbf{1} s^\top)}{\tau} \right), \quad (7)$$

where the softmax operator is applied row-wise, $d(x, y)$ is the L_1 -norm and $\tau \in \mathbb{R}_+$ a temperature-parameter. By utilizing this continuous relaxation of the argsort operator, we can train the permuter model g_ψ in Eq. (5) alongside the encoder and decoder model with stochastic gradient descent. In order to push the relaxed permutation matrix towards a real permutation matrix (only one 1 in every row and column), we add to Eq. (5) a row- and column-wise entropy term as additional penalty term:

$$C(\mathbf{P}) = \sum_i H(\bar{\mathbf{p}}_{i,\cdot}) + \sum_j H(\bar{\mathbf{p}}_{\cdot,j}), \quad (8)$$

with Shannon entropy $H(x) = -\sum_i x_i \log(x_i)$ and normalized probabilities $\bar{p}_{i,\cdot} = \frac{\mathbf{P}_{i,\cdot}}{\sum_j \mathbf{P}_{i,j}}$.

Propositions 1. A square matrix \mathbf{P} is a real permutation matrix if and only if $C(\mathbf{P}) = 0$ and the doubly stochastic constraint $p_{ij} \geq 0 \forall (i, j)$, $\sum_i p_{ij} = 1 \forall j$, $\sum_j p_{ij} = 1 \forall i$ holds.

Proof. See Appendix A.

By enforcing $\hat{\mathbf{P}} \rightarrow \mathbf{P}$, we ensure that no information about the graph structure is encoded in $\hat{\mathbf{P}}$ and decoder model $p_\theta(\mathcal{G}_\pi | \mathbf{z}, \mathbf{P})$ can generate valid graphs during inference, without providing a specific permutation matrix \mathbf{P} (e.g., one can set $\mathbf{P} = \mathbf{I}$ and decode the learned canonical node order). At this point it should also be noted, that our proposed framework can easily be generalized to arbitrary sets of elements, although we focus this work primarily on sets of nodes and edges defining a graph.

Graph Isomorphism Problem. Equation (5) gives us means to train an autoencoder framework with a permutation invariant encoder that maps a graph $f: \mathcal{G} \rightarrow \mathcal{Z}$ in an efficient manner. Such an encoder will always map two topologically identical graphs (even with different node order) to the same representation z . Consequently, the question arises, if we can decide for a pair of graphs whether they are topologically identical. This is the well-studied *graph isomorphism problem* for which no polynomial-time algorithm is known yet [16, 17]. As mentioned above, in our framework, two isomorphic graphs will always be encoded to the same representation. Still, it might be that two non-isomorphic graphs will be mapped to the same point (non-injective). However, if the decoder is able to perfectly reconstruct both graphs (which is easy to check since the permuter can be used to bring the decoded graph in the input node order), two non-isomorphic graphs must have a different representation z . If two graphs have the same representation and the reconstruction fails, the graphs might still be isomorphic but with no guarantees. Hence, our proposed model can solve the graph isomorphism problem at least for all graphs it can reconstruct.

2.3 Details of the Model Architecture

In this work we parameterize the encoder, decoder and permuter model in Eq. (5) by neural networks utilizing the self-attention framework proposed by Vaswani et al. [18] on directed messages representing a graph. Figure 1, visualizes the architecture of the proposed permutation-invariant variational autoencoder. In the following, we describe the different parts of the model in detail¹.

Graph Representation by Directional Messages. In general, most graph neural networks can be thought of as so called Message Passing Neural Networks (MPNN) [19]. The key idea of MPNNs is the aggregation of neighbourhood information by passing and receiving messages of each node to and from neighbouring nodes in a graph. We adopt this view and represent graphs by its messages between nodes. We represent a graph $\mathcal{G}(\mathbf{X}, \mathbf{E})$, with node features $\mathbf{X} \in \mathbb{R}^{n \times d_v}$ and edge features $\mathbf{E} \in \mathbb{R}^{n \times n \times d_e}$, by its message matrix $\mathbf{M} = (\mathbf{m}_{ij}) \in \mathbb{R}^{n \times n \times d_m}$:

$$\mathbf{m}_{ij} = \sigma([\mathbf{x}_i || \mathbf{x}_j || \mathbf{e}_{ij}] \mathbf{W} + \mathbf{b}), \quad (9)$$

with non-linearity σ , concatenation operator $||$ and trainable parameters \mathbf{W} and \mathbf{b} . Note, that nodes in this view are represented by *self-messages* $\text{diag}(\mathbf{M})$, messages between non-connected nodes exists, although the presence or absence of a connection might be encoded in \mathbf{e}_{ij} , and if \mathbf{M} is not symmetric, edges have an inherent direction.

Self-Attention on Directed Messages. We follow the idea of aggregating messages from neighbours in MPNNs, but utilize the self-attention framework proposed by Vaswani et al. [18] for sequential data. Our proposed model comprises multiple layers of multi-headed scaled-dot product attention. One attention head is defined by:

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d_k}}\right) \mathbf{V} \quad (10)$$

with queries $\mathbf{Q} = \mathbf{M}\mathbf{W}^Q$, keys $\mathbf{K} = \mathbf{M}\mathbf{W}^K$, and values $\mathbf{V} = \mathbf{M}\mathbf{W}^V$ and trainable weights $\mathbf{W}^Q \in \mathbb{R}^{d_m \times d_q}$, $\mathbf{W}^K \in \mathbb{R}^{d_m \times d_k}$ and $\mathbf{W}^V \in \mathbb{R}^{d_m \times d_v}$. For multi-headed self-attention we concatenate multiple attention heads together and feed them to a linear layer with d_m output features. Since the message matrix \mathbf{M} of a graph with n nodes comprises n^2 messages, attention of all messages to all messages would lead to a $O(n^4)$ complexity. We address this problem by letting messages

¹Code available at <https://github.com/jrwnter/pigvae>

\mathbf{m}_{ij} only attend on incoming messages \mathbf{m}_{ki} , reducing the complexity to $O(n^3)$. We achieve this by representing \mathbf{Q} as a $(m \times n \times d)$ tensor and \mathbf{K} and \mathbf{V} by a transposed $(n \times m \times d)$ tensor, resulting into a $(m \times n \times m)$ attention tensor, with number of nodes $m = n$ and number of features d . That way, we can efficiently utilize batched matrix multiplications in Eq. (10), in contrast to computing the whole $(n^2 \times n^2)$ attention matrix and masking attention on not incoming messages out.

Encoder To encode a graph into a fixed-sized, permutation-invariant, continuous latent representation, we add to input graphs a dummy node v_0 , acting as an embedding node. To distinguish the embedding node from other nodes, we add an additional node and edge type to represent this node and edges to and from this node. After encoding this graph into a message matrix $\mathbf{M}^{(\text{enc}, 0)}$ as defined in Eq. (9), we apply L iterations of self-attention to update $\mathbf{M}^{(\text{enc}, L)}$, accumulating the graph structure in the embedding node, represented by the self-message $\mathbf{m}_{0,0}^{(\text{enc}, L)}$. Following [11], we utilize the reparameterization trick and sample the latent representation \mathbf{z} of a graph by sampling from a multivariate normal distribution:

$$\mathbf{z} \sim \mathcal{N}(f_\mu(\mathbf{m}_{0,0}^{(\text{enc}, L)}), f_\sigma(\mathbf{m}_{0,0}^{(\text{enc}, L)})\mathbf{I}), \quad (11)$$

with $f_\mu : \mathbf{m}_{0,0} \rightarrow \mu \in \mathbb{R}^{d_z}$ and $f_\sigma : \mathbf{m}_{0,0} \rightarrow \sigma \in \mathbb{R}^{d_z}$, parameterized by a linear layer.

Permuter To predict how to re-order the nodes in the output graph to match the order of nodes in the input graph, we first extract node embeddings represented by self-messages on the main diagonal of the encoded message matrix $\mathbf{m}_{i,i}^{(\text{enc}, L)} = \text{diag}(\mathbf{M}^{(\text{enc}, L)})$ for $i > 0$. We score these messages by a function $f_s : \mathbf{m}_{i,i} \rightarrow s \in \mathbb{R}$, parameterized by a linear layer and apply the soft-sort operator (see Eq. (7)) to retrieve the permutation matrix $\hat{\mathbf{P}}$.

Decoder We initialize the message matrix for the decoder models input with the latent representation \mathbf{z} at each entry. To break symmetry and inject information about the relative position/order of nodes to each other, we follow [18] and define position embeddings in dimension k

$$\text{PE}(i)_k = \begin{cases} \sin(i/10000^{2k/d_z}), & \text{for even } k \\ \cos(i/10000^{2k/d_z}), & \text{for odd } k \end{cases} \quad (12)$$

It follows for the initial decoder message matrix $\mathbf{M}^{(\text{dec}, 0)}$:

$$\mathbf{m}_{ij}^{(\text{dec}, 0)} = \sigma([\mathbf{z} + [\text{PE}(i)||\text{PE}(j)]] \mathbf{W} + \mathbf{b}), \quad (13)$$

Since the self-attention based decoder model is permutation equivariant, we can move the permutation operation in Eq. (5) in front of the decoder model and directly apply it to the position embedding sequence (see Figure 1). After L iterations of self-attention on the message matrix \mathbf{M} , we extract node features $\mathbf{x}_i \in \mathbf{X}$ and edge features $\mathbf{e}_{i,j} \in \mathbf{E}$ by a final linear layer:

$$\mathbf{x}_i = \mathbf{m}_{i,i} \mathbf{W}^v + \mathbf{b}^v \quad \mathbf{e}_{i,j} = 0.5 \cdot (\mathbf{m}_{i,j} + \mathbf{m}_{j,i}) \mathbf{W}^e + \mathbf{b}^e, \quad (14)$$

with learnable parameters $\mathbf{W}^v \in \mathbb{R}^{d_m \times d_v}$, $\mathbf{W}^e \in \mathbb{R}^{d_m \times d_e}$, $\mathbf{b}^v \in \mathbb{R}^{d_v}$ and $\mathbf{b}^e \in \mathbb{R}^{d_e}$.

Overall Architecture We now describe the full structure of our proposed method using the ingredients above (see Figure 1). Initially, the input graph is represented by the directed message matrix $\mathbf{M}^{(\text{enc}, 0)}$, including an additional graph embedding node v_0 . The encoder model performs L iterations of self-attention on incoming messages. Next, diagonal entries of the resulting message matrix $\mathbf{M}^{(\text{enc}, L)}$ are extracted. Message $\mathbf{m}_{0,0}^{(\text{enc}, L)}$, representing embedding node v_0 , is used to condition the normal distribution, graph representation \mathbf{z} is sampled from. The other diagonal entries $\mathbf{m}_{i,i}^{(\text{enc}, L)}$ are transformed into scores and sorted by the Soft-Sort operator to retrieve the permutation matrix $\hat{\mathbf{P}}$. Next, position embeddings (in Figure 1 represented by single digits) are re-ordered by applying $\hat{\mathbf{P}}$ and added by the sampled graph embedding \mathbf{z} . The resulting node embeddings are used to initialize message matrix $\mathbf{M}^{(\text{dec}, 0)}$ and fed into the decoding model. After L iterations of self-attention, diagonal entries are transformed to node features \mathbf{X} and off-diagonal entries to edge features \mathbf{E} to generate the output graph. In order to train and infer on graphs of different size, we pad all graphs in a batch with empty nodes to match the number of nodes of the largest graph. Attention on empty nodes is masked out at all time. To generate graphs of variable size, we train alongside the variational autoencoder an additional multi-layer perceptron to predict the number of atoms of graph from its latent representation \mathbf{z} . During inference, this model informs the decoder on how many nodes to attend to.

Table 1: Negative log likelihood (NLL) and area under the receiver operating characteristics curve (ROC-AUC) for reconstruction of the adjacency matrix of graphs from different families. We compare our proposed method (PIGAE) with Graph Autoencoder (GAE) [8] and results of Graphite and Graph Autoencoder (GAE*) reported in [20]. PIGAE* utilize topological distances of nodes in a graph as edge feature.

MODELS	ERDOS-RENYI		BARABASI-ALBERT		EGO	
	NLL	ROC-AUC	NLL	ROC-AUC	NLL	ROC-AUC
PIGAE	20.5 ± 0.9	98.3 ± 0.1	27.2 ± 0.9	96.7 ± 0.2	23.4 ± 0.5	97.8 ± 0.3
PIGAE*	19.5 ± 0.8	99.4 ± 0.1	15.2 ± 0.8	99.5 ± 0.1	22.4 ± 0.5	98.8 ± 0.3
GAE	186 ± 3	57.9 ± 0.1	199 ± 3	57.4 ± 0.1	191 ± 4	59.1 ± 0.1
GAE*	222 ± 8	-	236 ± 15	-	197 ± 2	-
GRAPHITE	196 ± 1	-	192 ± 2	-	183 ± 1	-

Key Architectural Properties Since no position embeddings are added to the input of the encoders self-attention layers, accumulated information in the single embedding node $v_0(\mathbf{m}_{0,0})$ is invariant to permutations of the input node order. Hence, the resulting graph embedding \mathbf{z} is permutation invariant as well. This is in stark contrast to classical graph autoencoder frameworks [8, 20, 21], that encode whole graphs effectively by concatenating all node embeddings, resulting in a graph-level representation that is different for isomorphic graphs, as the sequence of node embeddings permutes equivalently with the input node order. As no information about the node order is encoded in the graph embedding \mathbf{z} , the decoder learns its own (canonical) node order, distinct graphs are deterministically decoded in. The input node order does not influence this decoded node order. As the decoder is based on permutation equivariant self-attention layers, this canonical order is solely defined with respect to the sequence of position embeddings used to initialize the decoders input. If the sequence of position embeddings is permuted, the decoded node order permutes equivalently. Thus, by predicting the right permutation matrix, input and output order can be aligned to correctly calculate the reconstruction loss. Input to the permuter model $[\mathbf{m}_{1,1}, \dots, \mathbf{m}_{n+1,n+1}]$ is equivariant to permutations in the input node order (due to the equivariant self-attention layers in the encoder). Since the permuter model itself (i.e., the scoring function) is also permutation equivariant (node-wise linear layer), resulting permutation matrices \mathbf{P} are equivariant to permutations in the input node order. Consequently, if the model can correctly reconstruct a graph in a certain node order, it can do it for all $n!$ input node orders, and the learning process of the whole model is independent to the node order of graphs in the training set.

3 Related Works

Most existing research on unsupervised graph representation learning focuses on *node-level* representation learning and can be broadly categorized in either shallow methods based on matrix factorization [22, 23, 24, 25] or random walks [26, 27], and deep methods based on Graph Neural Networks (GNN) [2, 3]. Kipf and Welling [8] proposed a graph autoencoder (GAE), reconstructing the adjacency matrix by taking the dot product between the latent node embeddings encoded by a GNN. Grover et al. [20] build on top of the GAE framework by parameterizing the decoder with additional GNNs, further refining the decoding process. Although *graph-level* representations in GAE-like approaches can be constructed by concatenating all node-level representations, note, that as a consequence they are only permutation equivariant and not permutation invariant. Permutation invariant representations could be extracted only after training by aggregating node embeddings into a single-vector representation. However, such a representation might miss important global graph structure. Samanta et al. [21] proposed a GAE-like approach, which parameters are trained in a permutation invariant way, following [28] utilizing breadth-first-traversals with randomized tie breaking during the child-selection step. However, as graph-level representations are still constructed by concatenation of node-level embeddings, this method still encodes graphs only in a permutation-equivariant way. A different line of work utilized Normalizing Flows [29] to address variational inference on graph structured data based on node-level latent variables [30, 31, 32].

Research on *graph-level* representations has mainly focused on supervised learning, e.g., graph-level classification by applying a GNN followed by a global node feature aggregation step (e.g., mean or max pooling) [4] or a jointly learned aggregation scheme [33]. Research on graph-level unsupervised

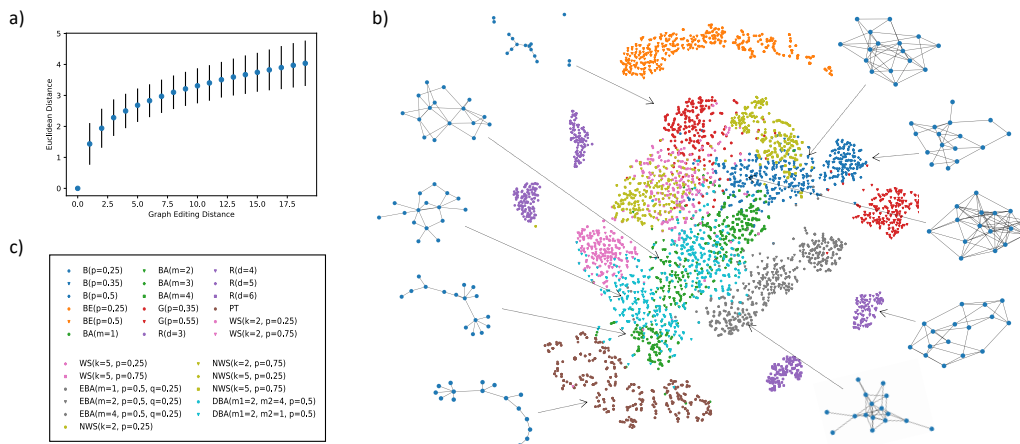


Figure 2: a) Euclidean distance over graph editing distance, averaged over 1000 Barabasi-Albert graphs with $m = 2$. b) t-SNE projection of representations from ten different graph families with different parameters. Example graphs are shown for some of the clusters. c) Legend of t-SNE plot explaining colours and symbols. Graph family abbreviations: Binomial (B), Binomial Ego (BE), Barabasi-Albert (BA), Geometric (G), Regular (R), Powerlaw Tree (PT), Watts-Strogatz (WA), Extended Barabasi-Albert (EBA), Newman-Watts-Strogatz (NWA), Dual-Barabasi-Albert (DBA).

representation learning has not yet received much attention and existing work is mainly based on contrastive learning approaches. Narayanan et al. [34] adapted the *doc2vec* method from the field of natural language processing to represent whole graphs by a fixed size embedding, training a *skipgram* method on rooted subgraphs (*graph2vec*). Bai et al. [35] proposed a *Siamese* network architecture, trained on minimizing the difference between the Euclidean distance of two encoded graphs and their graph editing distance. Recently, Sun et al. [36], adapted the *Deep InfoMax* architecture [37] to graphs, training on maximizing the mutual information between graph-level representations and representations of sub-graphs of different granularity (*InfoGraph*). Although those contrastive learning approaches can be designed to encode graphs in a permutation invariant way, they cannot be used to reconstruct or generate graphs from such representations.

Another line of related work concerns itself with *generative models* for graphs. Besides methods based on variational autoencoders [8, 20, 12] and Normalizing Flows [30, 31, 32], graph generative models have also been recently proposed based on generative adversarial neural networks [38, 39] and deep auto-regressive models [40, 28, 41]. Moreover, due to its high practical value for drug discovery, many graph generating methods have been proposed for molecular graphs [42, 43, 44, 45, 21]. Although graph generative models can be trained in a permutation invariant way [42, 21], those models can not be used to extract permutation invariant graph-level representations.

Recently, Yang et al. [46] proposed a GAE-like architecture with a node-feature aggregation step to extract permutation invariant graph-level representations that can also be used for graph generation. They tackle the discussed ordering issue of GAEs in the reconstruction by training alongside the GAE a Generative Adversarial Neural Network, which's permutation invariant discriminator network is used to embed input and output graph into a latent space. That way, a permutation invariant reconstruction loss can be defined as a distance in this space. However, as this procedure involves adversarial training of the reconstruction metric, this only approximates the exact reconstruction loss used in our work and might lead to undesirable graph-level representations.

4 Experimental Evaluation

We perform experiments on synthetically generated graphs and molecular graphs from the public datasets QM9 and PubChem. At evaluation time, predicted permutation matrices are always discretized to ensure their validity. For more details on training, see Appendix C.

Table 2: Classification Accuracy of our method (PIGAE), classical GAE, InfoGraph (IG), Shortest Path Kernel (SP) and Weisfeiler-Lehman Sub-tree Kernel (WL) on graph class prediction.

PIGAE	GAE	IG	SP	WL
0.83 ± 0.01	0.65 ± 0.01	0.75 ± 0.02	0.50 ± 0.02	0.73 ± 0.01

4.1 Synthetic Data

Graph Reconstruction In the first experiment we evaluate our proposed method on the reconstruction performance of graphs from graph families with a well-defined generation process. Namely, Erdos-Renyi graphs [47], with an edge probability of $p = 0.5$, Barabasi-Albert graphs [48], with $m = 4$ edges preferentially attached to nodes with high degree and Ego graphs. For each family we uniformly sample graphs with 12-20 nodes. The graph generation parameters match the ones reported in [20], enabling us to directly compare to Graphite. As additional baseline we compare against the Graph Autoencoder (GAE) proposed by Kipf and Welling [8]. As Grover et al. [20] only report negative log-likelihood estimates for their method Graphite and baseline GAE we also reevaluate GAE and report both negative log-likelihood (NLL) estimates for GAE to make a better comparison to Graphite possible (accounting for differences in implantation or the graph generation process). In Table 1 we show the evaluation metrics on a fixed test set of 25000 graphs for each graph family. On all four graph datasets our proposed model significantly outperforms the baseline methods, reducing the NLL error in three of the four datasets by approximately one magnitude. Utilizing the topological distance instead of just the connectivity as edge feature (compare [49]) further improves the reconstruction performance.

Qualitative Evaluation To evaluate the representations learned by our proposed model, we trained a model on a dataset of randomly generated graphs with variable number of nodes from ten different graph families with different ranges of parameters (see Appendix B for details). Next, we generated a test set of graphs with a fixed number of nodes from these ten different families and with different fixed parameters. In total we generated graphs in 29 distinct settings. In Figure 2, we visualized the t-SNE projection [50] of the graph embeddings, representing different families by colour and different parameters within each family by different symbols. In this 2-D projection, we can make out distinct clusters for the different graph sets. Moreover, clusters of similar graph sets tend to cluster closer together. For example, Erdos-Renyi graphs form for each edge probability setting (0.25, 0.35, 0.5) a distinct cluster, while clustering in close proximity. As some graph families with certain parameters result in similar graphs, some clusters are less separated or tend to overlap. For example, the Dual-Barabasi-Albert graph family, which attaches nodes with either m_1 or m_2 other nodes, naturally clusters in between the two Barabasi-Albert graph clusters with $m = m_1$ and $m = m_2$.

Graph Editing Distance A classical way of measuring graph similarity is the so called *graph editing distance* (GED) [51]. The GED between two graphs measures the minimum number of graph editing operations to transform one graph into the other. The set of operations typically includes inclusion, deletion and substitution of nodes or edges. To evaluate the correlation between similarity in graph representation and graph editing distance, we generated a set of 1000 Barabasi-Albert ($m = 3$) graphs with 20 nodes. For each graph we successively substitute randomly an existing edge by a new one, creating a set of graphs with increasing GED with respect to the original graph. In Figure 2, we plot the mean Euclidean distance between the root graphs and their 20 derived graphs with increasing GED. We see a strong correlation between GED and Euclidean distance of the learned representations. In contrast to classical GAEs, random permutations of the edited graphs have no effect on this correlation (see Appendix D for comparison).

Graph Isomorphism and Permutation Matrix To empirical analyse if our proposed method detects isomorphic graphs, we generated for 10000 Barabasi-Albert graphs with up to 28 nodes a randomly permuted version and a variation only one graph editing step apart. For all graphs the Euclidean distance between original graph and edited graph was at least greater than 0.3. The randomly permuted version always had the same embedding. Even for graphs out of training domain (geometric graphs with 128 nodes) all isomorphic and non-isomorphic graphs could be detected. Additionally, we investigated how well the permuter model can assign a permutation matrix to graph.

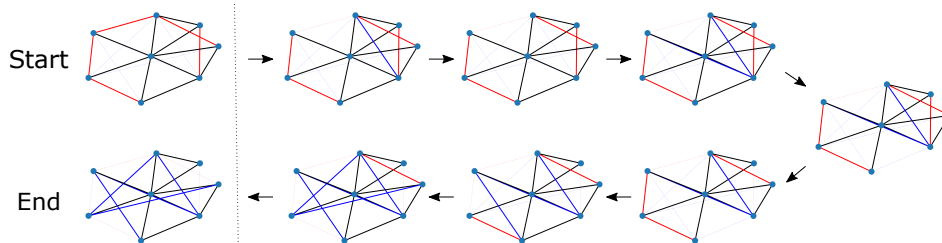


Figure 3: Linear interpolation between two graphs in the embedding space. Start graph’s edges are colored red. End graph’s edges are colored blue. Edges present in both graphs are colored black. Thick lines are present in a decoded graph, thin lines are absent.

As the high reconstruction performance in Table 1 suggest, most of the time, correct permutation matrices are assigned (See Appendix E for a more detailed analysis). We find that permutation matrices equivalently permute with the permutation of the input graph (See Appendix E).

Graph Classification In order to quantitatively evaluate how meaningful the learned representations are, we evaluate the classification performance of a Support Vector Machine (SVM) on predicting the correct graph set label (29 classes as defined above) from the graph embedding. As baseline we compare against SVMs based on two classical graph kernel methods, namely Shortest Path Kernel (SP) [52] and Weisfeiler-Lehman Sub-tree Kernel (WL) [53] as well as embeddings extracted by the recently proposed contrastive learning model InfoGraph (IG) [36] and averaged node embeddings extracted by a classical GAE. Our model, IG and GAE were trained on the same synthetic dataset. In Table 2 we report the accuracy for each model and find the SVM based on representations from our proposed model to significantly outperform all baseline models. Notably, representations extracted from a classical GAE model, by aggregating (averaging) node-level embeddings into a graph-level embedding, perform significantly worse compared to representations extracted by our method. This finding is consistent with our hypothesis that aggregation of unsupervised learned node-level (local) features might miss important global features, motivating our work on graph-level unsupervised representation learning.

Graph Interpolation The permutation invariance property of graph-level representations also enables the interpolation between two graphs in a straight-forward way. With classical GAEs such interpolations cannot be done in a meaningful way, as interpolations between permutation dependent graph embeddings would affect both graph structure as well as node order. In Figure 3 we show how successive linear interpolation between the two graphs in the embedding space results in smooth transition in the decoded graphs, successively deleting edges from the start graph (red) and adding edges from the end graph (blue). To the best of our knowledge, such graph interpolations have not been report in previous work yet and might show similar impact in the graph generation community as interpolation of latent spaces did in the field of Natural Language Processing and Computer Vision.

4.2 Molecular Graphs

Next, we evaluate our proposed model on molecular graphs from the QM9 dataset [54, 55]. This datasets contains about 134 thousand organic molecules with up to 9 heavy atoms (up to 29 atoms/nodes including Hydrogen). Graphs have 5 different atom types (C, N, O, F and H), 3 different formal charge types (-1, 0 and 1) and 5 differ bond types (no-, single-, double-, triple- and aromatic bond). Moreover, the dataset contains an energetically favorable conformation for each molecule in form of Cartesian Coordinates for each atom. We transform these coordinates to an (rotation-invariant) Euclidean distance matrix and include the distance information as additional edge feature to the graph representation (More details in Appendix F).

Graph Reconstruction and Generation We define a holdout set of 10,000 molecules and train the model on the rest. Up on convergence, we achieve on the hold out set a balanced accuracy of 99.93% for element type prediction, 99.99% for formal charge type prediction and 99.25% for edge type prediction (includes prediction of non-existence of edges). Distances between atoms are reconstructed with a root mean squared error of 0.33\AA and a coefficient of determination of $R^2 = 0.94$.

Dataset	PIGAE (ours)	ECFP
Classification (ROC-AUC \uparrow)		
BACE	0.824 ± 0.005	0.82 ± 0.02
BBBP	0.81 ± 0.04	0.78 ± 0.03
Regression (MSE \downarrow)		
ESOL	0.10 ± 0.01	0.25 ± 0.02
LIPO	0.34 ± 0.02	0.39 ± 0.02

Table 3: Downstream performance for molecular property prediction tasks.

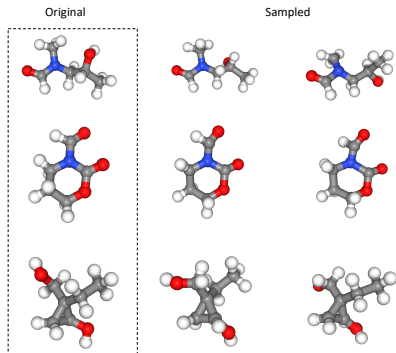


Figure 4: Example Molecular conformations sampled around original graph representation.

In the field of computational chemistry and drug discovery, the generation of energetically reasonable conformations for molecules is a topic of great interest [56]. Since our model is trained on approximating the data generating process for molecules and their conformations, we can utilize the trained model to sample molecular conformations. To retrieve a set of conformations, we encode a query molecule into its latent representation and randomly sample around this point by adding a small noise and decode the resulting representations. We utilize *Multidimensional Scaling* (MDS) [57] to transform a decoded distance matrix back to Cartesian Coordinates. In Figure 4, we show examples of molecular conformations sampled from the trained model. Under visual inspection, we find that sampled conformations differ from encoded conformations, while still being energetically reasonable (e.g., rotation along rotatable bonds while avoiding clashes between atoms).

Molecular Property Prediction Finally, we evaluate the learned representations for molecular property prediction. In order to accurately represent molecular graphs from different parts of the chemical space, we train our proposed model on a larger dataset retrieved from the public PubChem database [58]. We extracted organic molecules with up to 32 heavy atoms, resulting into a set of approximately 67 million compounds (more details in Appendix F). We evaluate the representations of the trained model based on the predictive performance of a SVM on two classification and two regression tasks from the MoleculeNet benchmark [59]. We compare representations derived from our pre-trained model with the state-of-the-art Extended Connectivity Fingerprint molecular descriptors (radius=3, 2048 dim) [60]. For each task and descriptor, the hyperparameter C of the SVM was tuned in a nested cross validation. The results are presented in Table 3. Descriptors derived from our pre-trained model seem to represent molecular graphs in a meaningful way as they outperform the baseline on average in three out of four tasks.

5 Conclusion, Limitations and Future Work

In this work we proposed a permutation invariant autoencoder for graph-level representation learning. By predicting the relation (permutation matrix) between input and output graph order, our proposed model can directly be trained on node and edge feature reconstruction, while being invariant to a distinct node order. This poses, to the best of our knowledge, the first method for non-contrastive and non-adversarial learning of permutation invariant graph-level representations that can also be used for graph generation and might be an important step towards more powerful representation learning methods on graph structured data or sets in general. We demonstrate the effectiveness of our method in encoding graphs into meaningful representations and evaluate its competitive performance in various experiments. Although we propose a way of reducing the computational complexity by only attending on incoming messages in our directed message self-attention framework, in its current state, our proposed model is limited in the number of nodes a graph can consist of. However, recently, much work has been done on more efficient and sparse self-attention frameworks [61, 62, 63]. In future work, we aim at building up on this work to scale our proposed method to larger graphs. Moreover, we will investigate further into the generative performance of the proposed model, as this work was mainly concerned with its representation learning capability.

Funding Disclosures

D.A.C. received financial support from European Commission grant numbers 963845 and 956832 under the Horizon2020 Framework Program for Research and Innovation. F.N. acknowledges funding from the European Commission (ERC CoG 772230 “ScaleCell”) and MATH+ (AA1-6). F.N. is advisor for Relay Therapeutics and scientific advisory board member of 1qbit and Redesign Science.

References

- [1] William L Hamilton, Rex Ying, and Jure Leskovec. Representation learning on graphs: Methods and applications. *arXiv preprint arXiv:1709.05584*, 2017.
- [2] Michael M Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. Geometric deep learning: going beyond euclidean data. *IEEE Signal Processing Magazine*, 34(4):18–42, 2017.
- [3] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems*, 2020.
- [4] David K Duvenaud, Dougal Maclaurin, Jorge Iparraguirre, Rafael Bombarell, Timothy Hirzel, Alán Aspuru-Guzik, and Ryan P Adams. Convolutional networks on graphs for learning molecular fingerprints. *Advances in neural information processing systems*, 28:2224–2232, 2015.
- [5] Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard Zemel. Gated graph sequence neural networks. *arXiv preprint arXiv:1511.05493*, 2015.
- [6] Shaosheng Cao, Wei Lu, and Qionгкаi Xu. Deep neural networks for learning graph representations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 30, 2016.
- [7] Daixin Wang, Peng Cui, and Wenwu Zhu. Structural deep network embedding. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1225–1234, 2016.
- [8] Thomas Kipf and Max Welling. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308v*, 2016.
- [9] Jiezhong Qiu, Yuxiao Dong, Hao Ma, Jian Li, Kuansan Wang, and Jie Tang. Network embedding as matrix factorization: Unifying deepwalk, line, pte, and node2vec. In *Proceedings of the eleventh ACM international conference on web search and data mining*, pages 459–467, 2018.
- [10] Shirui Pan, Ruiqi Hu, Guodong Long, Jing Jiang, Lina Yao, and Chengqi Zhang. Adversarially regularized graph autoencoder for graph embedding. *arXiv preprint arXiv:1802.04407*, 2018.
- [11] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.
- [12] Martin Simonovsky and Nikos Komodakis. Graphvae: Towards generation of small graphs using variational autoencoders. In *International Conference on Artificial Neural Networks*, pages 412–422. Springer, 2018.
- [13] Minsu Cho, Jian Sun, Olivier Duchenne, and Jean Ponce. Finding matches in a haystack: A max-pooling strategy for graph matching in the presence of outliers. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2083–2090, 2014.
- [14] Sebastian Prillo and Julian Eisenschlos. Softsort: A continuous relaxation for the argsort operator. In *International Conference on Machine Learning*, pages 7793–7802. PMLR, 2020.
- [15] Aditya Grover, Eric Wang, Aaron Zweig, and Stefano Ermon. Stochastic optimization of sorting networks via continuous relaxations. *arXiv preprint arXiv:1903.08850*, 2019.

- [16] Michael R Garey. A guide to the theory of np-completeness. *Computers and intractability*, 1979.
- [17] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*, 2018.
- [18] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.
- [19] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. *arXiv preprint arXiv:1704.01212*, 2017.
- [20] Aditya Grover, Aaron Zweig, and Stefano Ermon. Graphite: Iterative generative modeling of graphs. In *International conference on machine learning*, pages 2434–2444. PMLR, 2019.
- [21] Bidisha Samanta, Abir De, Gourhari Jana, Vicenç Gómez, Pratim Kumar Chattaraj, Niloy Ganguly, and Manuel Gomez-Rodriguez. Nevae: A deep generative model for molecular graphs. *Journal of machine learning research*. 2020 Apr; 21 (114): 1-33, 2020.
- [22] Amr Ahmed, Nino Shervashidze, Shравan Narayanamurthy, Vanja Josifovski, and Alexander J Smola. Distributed large-scale natural graph factorization. In *Proceedings of the 22nd international conference on World Wide Web*, pages 37–48, 2013.
- [23] Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps and spectral techniques for embedding and clustering. In *Nips*, volume 14, pages 585–591, 2001.
- [24] Shaosheng Cao, Wei Lu, and Qionghai Xu. Grarep: Learning graph representations with global structural information. In *Proceedings of the 24th ACM international on conference on information and knowledge management*, pages 891–900, 2015.
- [25] Mingdong Ou, Peng Cui, Jian Pei, Ziwei Zhang, and Wenwu Zhu. Asymmetric transitivity preserving graph embedding. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1105–1114, 2016.
- [26] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 701–710, 2014.
- [27] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 855–864, 2016.
- [28] Jiaxuan You, Rex Ying, Xiang Ren, William Hamilton, and Jure Leskovec. Graphrnn: Generating realistic graphs with deep auto-regressive models. In *International Conference on Machine Learning*, pages 5708–5717. PMLR, 2018.
- [29] Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. In *International conference on machine learning*, pages 1530–1538. PMLR, 2015.
- [30] Jenny Liu, Aviral Kumar, Jimmy Ba, Jamie Kiros, and Kevin Swersky. Graph normalizing flows. *arXiv preprint arXiv:1905.13177*, 2019.
- [31] Chengxi Zang and Fei Wang. Moflow: an invertible flow model for generating molecular graphs. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 617–626, 2020.
- [32] Tony Duan and Juho Lee. Graph embedding vae: A permutation invariant model of graph structure. *arXiv preprint arXiv:1910.08057*, 2019.
- [33] Rex Ying, Jiaxuan You, Christopher Morris, Xiang Ren, William L Hamilton, and Jure Leskovec. Hierarchical graph representation learning with differentiable pooling. *arXiv preprint arXiv:1806.08804*, 2018.

- [34] Annamalai Narayanan, Mahinthan Chandramohan, Rajasekar Venkatesan, Lihui Chen, Yang Liu, and Shantanu Jaiswal. graph2vec: Learning distributed representations of graphs. *arXiv preprint arXiv:1707.05005*, 2017.
- [35] Yunsheng Bai, Hao Ding, Yang Qiao, Agustin Marinovic, Ken Gu, Ting Chen, Yizhou Sun, and Wei Wang. Unsupervised inductive graph-level representation learning via graph-graph proximity. *arXiv preprint arXiv:1904.01098*, 2019.
- [36] Fan-Yun Sun, Jordan Hoffmann, Vikas Verma, and Jian Tang. Infograph: Unsupervised and semi-supervised graph-level representation learning via mutual information maximization. *arXiv preprint arXiv:1908.01000*, 2019.
- [37] R Devon Hjelm, Alex Fedorov, Samuel Lavoie-Marchildon, Karan Grewal, Phil Bachman, Adam Trischler, and Yoshua Bengio. Learning deep representations by mutual information estimation and maximization. *arXiv preprint arXiv:1808.06670*, 2018.
- [38] Hongwei Wang, Jia Wang, Jialin Wang, Miao Zhao, Weinan Zhang, Fuzheng Zhang, Xing Xie, and Minyi Guo. Graphgan: Graph representation learning with generative adversarial nets. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.
- [39] Nicola De Cao and Thomas Kipf. Molgan: An implicit generative model for small molecular graphs. *arXiv preprint arXiv:1805.11973*, 2018.
- [40] Yujia Li, Oriol Vinyals, Chris Dyer, Razvan Pascanu, and Peter Battaglia. Learning deep generative models of graphs. *arXiv preprint arXiv:1803.03324*, 2018.
- [41] Chenhao Niu, Yang Song, Jiaming Song, Shengjia Zhao, Aditya Grover, and Stefano Ermon. Permutation invariant graph generation via score-based generative modeling. In *International Conference on Artificial Intelligence and Statistics*, pages 4474–4484. PMLR, 2020.
- [42] Jiaxuan You, Bowen Liu, Rex Ying, Vijay Pande, and Jure Leskovec. Graph convolutional policy network for goal-directed molecular graph generation. *arXiv preprint arXiv:1806.02473*, 2018.
- [43] Wengong Jin, Regina Barzilay, and Tommi Jaakkola. Junction tree variational autoencoder for molecular graph generation. In *International Conference on Machine Learning*, pages 2323–2332. PMLR, 2018.
- [44] Rafael Gómez-Bombarelli, Jennifer N Wei, David Duvenaud, José Miguel Hernández-Lobato, Benjamín Sánchez-Lengeling, Dennis Sheberla, Jorge Aguilera-Iparraguirre, Timothy D Hirzel, Ryan P Adams, and Alán Aspuru-Guzik. Automatic chemical design using a data-driven continuous representation of molecules. *ACS central science*, 4(2):268–276, 2018.
- [45] Rocío Mercado, Tobias Rastemo, Edvard Lindelöf, Günter Klambauer, Ola Engkvist, Hongming Chen, and Esben Jannik Bjerrum. Graph networks for molecular design. *Machine Learning: Science and Technology*, 2020.
- [46] Carl Yang, Peiye Zhuang, Wenhan Shi, Alan Luu, and Pan Li. Conditional structure generation through graph variational generative adversarial nets. In *NeurIPS*, pages 1338–1349, 2019.
- [47] Paul Erdős and Alfréd Rényi. On the evolution of random graphs.
- [48] Albert-László Barabási and Réka Albert. Emergence of scaling in random networks. *science*, 286(5439):509–512, 1999.
- [49] Pan Li, Yanbang Wang, Hongwei Wang, and Jure Leskovec. Distance encoding: Design provably more powerful neural networks for graph representation learning. *arXiv preprint arXiv:2009.00142*, 2020.
- [50] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.
- [51] Alberto Sanfeliu and King-Sun Fu. A distance measure between attributed relational graphs for pattern recognition. *IEEE transactions on systems, man, and cybernetics*, (3):353–362, 1983.

- [52] Karsten M Borgwardt and Hans-Peter Kriegel. Shortest-path kernels on graphs. In *Fifth IEEE international conference on data mining (ICDM'05)*, pages 8–pp. IEEE, 2005.
- [53] Nino Shervashidze, Pascal Schweitzer, Erik Jan Van Leeuwen, Kurt Mehlhorn, and Karsten M Borgwardt. Weisfeiler-lehman graph kernels. *Journal of Machine Learning Research*, 12(9), 2011.
- [54] Lars Ruddigkeit, Ruud Van Deursen, Lorenz C Blum, and Jean-Louis Reymond. Enumeration of 166 billion organic small molecules in the chemical universe database gdb-17. *Journal of chemical information and modeling*, 52(11):2864–2875, 2012.
- [55] Raghunathan Ramakrishnan, Pavlo O Dral, Matthias Rupp, and O Anatole Von Lilienfeld. Quantum chemistry structures and properties of 134 kilo molecules. *Scientific data*, 1(1):1–7, 2014.
- [56] Paul CD Hawkins. Conformation generation: the state of the art. *Journal of Chemical Information and Modeling*, 57(8):1747–1756, 2017.
- [57] Joseph B Kruskal. *Multidimensional scaling*. Number 11. Sage, 1978.
- [58] Sunghwan Kim, Paul A Thiessen, Evan E Bolton, Jie Chen, Gang Fu, Asta Gindulyte, Lianyi Han, Jane He, Siqian He, Benjamin A Shoemaker, et al. Pubchem substance and compound databases. *Nucleic acids research*, 44(D1):D1202–D1213, 2016.
- [59] Zhenqin Wu, Bharath Ramsundar, Evan N Feinberg, Joseph Gomes, Caleb Geniesse, Aneesh S Pappu, Karl Leswing, and Vijay Pande. Moleculenet: a benchmark for molecular machine learning. *Chemical science*, 9(2):513–530, 2018.
- [60] David Rogers and Mathew Hahn. Extended-connectivity fingerprints. *Journal of chemical information and modeling*, 50(5):742–754, 2010.
- [61] Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*, 2019.
- [62] Juho Lee, Yoonho Lee, Jungtaek Kim, Adam Kosiorek, Seungjin Choi, and Yee Whye Teh. Set transformer: A framework for attention-based permutation-invariant neural networks. In *International Conference on Machine Learning*, pages 3744–3753. PMLR, 2019.
- [63] Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. Reformer: The efficient transformer. *arXiv preprint arXiv:2001.04451*, 2020.

Checklist

1. For all authors...
 - (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
 - (b) Did you describe the limitations of your work? [Yes] See Section 5
 - (c) Did you discuss any potential negative societal impacts of your work? [No]
 - (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
 - (a) Did you state the full set of assumptions of all theoretical results? [Yes] See Section 2.1
 - (b) Did you include complete proofs of all theoretical results? [Yes] See Appendix A
3. If you ran experiments...
 - (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes]
 - (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Appendix C
 - (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]
 - (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See Appendix C
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
 - (a) If your work uses existing assets, did you cite the creators? [Yes]
 - (b) Did you mention the license of the assets? [N/A]
 - (c) Did you include any new assets either in the supplemental material or as a URL? [No]
 - (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
 - (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
 - (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
 - (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
 - (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

Appendix:

Permutation-Invariant Variational Autoencoder for Graph-Level Representation Learning

Robin Winter
Bayer AG
Freie Universität Berlin
robin.winter@bayer.com

Frank Noé
Freie Universität Berlin
frank.noe@fu-berlin.de

Djork-Arné Clevert
Bayer AG
djork-arne.clevert@bayer.com

Appendix A

Propositions 1. A square matrix \mathbf{P} is a real permutation matrix if and only if $C(\mathbf{P}) = 0$ and the doubly stochastic constraint $p_{ij} \geq 0 \forall (i, j)$, $\sum_i p_{ij} = 1 \forall j$, $\sum_j p_{ij} = 1 \forall i$ holds.

Proof. A permutation matrix is a doubly stochastic matrix with only one 1 in every row and column. (\Rightarrow) If P is a permutation matrix, $C(\mathbf{P}) = 0$ and the doubly stochastic constraint are satisfied by definition.

(\Leftarrow) If $C(\mathbf{P}) = 0 \Rightarrow \sum_i H(\bar{p}_{ij}) = 0 \forall j \wedge \sum_j H(\bar{p}_{ij}) = 0 \forall i$. Thus, if doubly stochastic constraint are satisfied, \bar{p}_{ij} can only have one non-zero element in each row i and column j equal to 1.

Remark Since we apply the row-wise softmax in Eq. (7), $\sum_j p_{ij} = 1 \forall i$ and $p_{ij} \geq 0 \forall (i, j)$ is always fulfilled. If $C(\mathbf{P}) = 0$, all but one entry in a column $p_{i\cdot}$ are 0 and the other entry is 1. Hence, $\sum_i p_{ij} = 1 \forall j$ is fulfilled.

Appendix B

Synthetic random graph generation To generate train and test graph datasets we utilized the python package *NetworkX* [1]. We sampled from ten different graph families with different parameter ranges, namely:

- Binominal graphs with edge probability $p \in (0.2, 0.6)$.
- Ego graphs extracted from Binominal graphs ($p \in (0.2, 0.6)$) selecting all neighbours of one random node.
- Watts-Strogatz small-world graphs with $k \in (2, 6)$ nearest neighbours and edge probability $p \in (0.2, 0.6)$.
- Newman-Watts-Strogatz small-world graphs with $k \in (2, 6)$ nearest neighbours and edge probability $p \in (0.2, 0.6)$.
- Random Regular graphs with degree $d \in (3, 6)$.
- Barabási–Albert graphs with $m \in (1, 6)$ edges preferentially attached to high degree nodes.
- Dual-Barabási–Albert graphs with $m_1 \in (1, 6)$ and $m_2 \in (1, 6)$ edges preferentially attached to high degree nodes and probability $p \in (0.1, 0.9)$ for sampling m_1 edges.

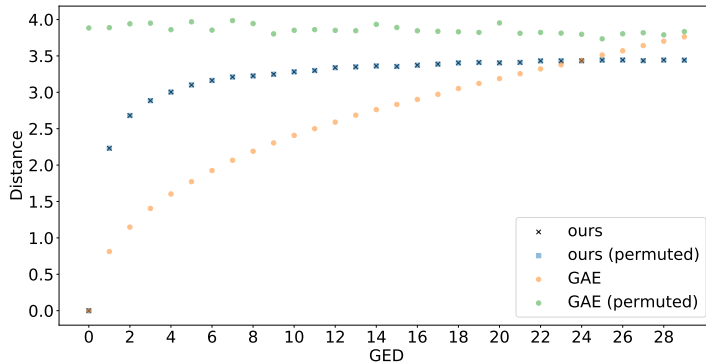


Figure 1: Distance in embedding space over increasing (approximated) graph editing distance (GED) average over 1000 Barabási–Albert graphs with 16 nodes. We compare our method against a classical graph autoencoder (GAE) for either the same graph ordering or random permutations of the same/edited graph.

- Extended-Barabási–Albert graphs with $m \in (1, 6)$ edges preferentially attached to high degree nodes and probability $p \in (0.1, 0.5)$ for adding an edge between existing nodes and probability $q \in (0.1, 0.5)$ for rewiring of existing edges.
- Random-Powerlaw-Tree graph with $\gamma = 3$.
- Random Geometric graph with edges between nodes in a unit square less than $r \in (0.35, 0.65)$ units apart.

Appendix C

Training Details We did not perform an extensive hyperparameter evaluation for the different experiments and mostly followed [2] for hyperparameter selection. We applied $L = 16$ layers of self-attention in both encoder and decoder, with a hidden dim chosen from $(256, 512)$ with either 16 or 32 attention heads with 64 hidden dimensions. Each self attention layer was followed by a point-wise fully connected neural network with two layers (1024 hidden dim) and a residual connection. We set the graph embedding dimension to 64. We tried different weightings of reconstruction and permutation matrix penalty loss to maximize the reconstruction accuracy with a discretized permutation matrix, while enabling stable training. In some settings we found it beneficial, to slowly decay the temperature constant τ of the Soft-Sort operator during training. We performed all experiments on a NVIDIA DGX-2 system, parallelizing models on up to 10 GPUs and training up to 5 days, depending on task and model size.

Appendix D

Graph Editing Distance In section 4.1 we describe how distances in the graph embedding space of our proposed model correlates with the graph editing distance (GED). One important property of the GED is its invariance to the node ordering of graphs that are compared. Thus, two similar graphs should be assigned the same small distance irregardless of the specific node ordering both graphs are represented in. Since embeddings produced by our proposed model are invariant to node permutations, distances between graphs in this embedding space, like the GED, are invariant to the node ordering as well. This is not the case for graph-level representations extracted by a classical graph autoencoder (GAE) [3]. In Figure 1 we plot the Euclidean distance between a reference graph representation and representations of graphs with increasing amount of variations applied to it. We followed the protocol described in Section 4.1 and created variations of graphs by successively substituting existing edges in the reference graph by new edges not present in the reference graph. We compare representations encoded by our method with representations encoded by a classical GAE.

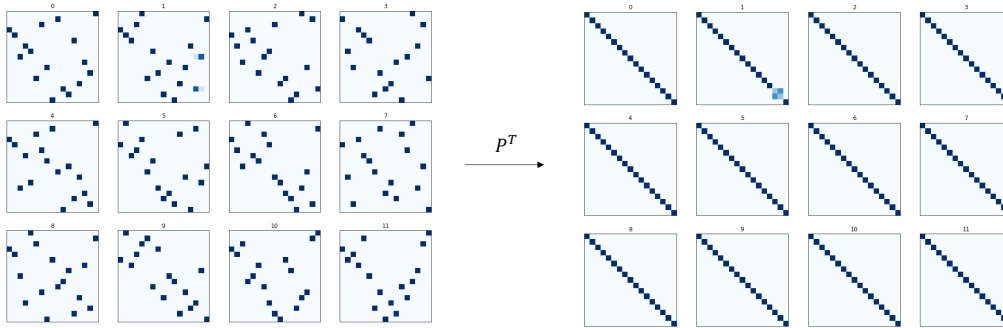


Figure 2: Left: Predicted permutation matrices for 12 random Barabási–Albert graphs. Right: Predicted permutation matrices after permuting the same input graphs by the transpose of the previously predicted permutation matrices.

We can see a clear correlation between the distance in embedding space and the amount of editing steps performed to the reference graph. However, in the case of the GAE, this correlation breaks, if we change the order of nodes in the edited graphs. This is expected, since the graph-level embedding of a GAE should permute equivalently with changes in the graph order. Since our graph-level representation is permutation-invariant, random permutations of the edited graphs have no effect on the Euclidean distance.

Interestingly, the distance of embeddings extracted by our method seem to converge after a certain amount of graph editing steps, while distances of embeddings extracted by the GAE seem to increase further. The reason for this might be that successively substituting edges of the reference graph with new edges must not necessarily result into more dissimilar graphs at some point. It is for instance possible to recreate the same graph again, just in a different node order. With increasing amounts of editing steps a graph should be on average as far away as the average distance between any pair of graphs (of the same size and family). In our case (Barabási–Albert graphs with $m=4$ and 16 nodes) this average pairwise distance between 1000 randomly generated graphs is approximately 3.3, which matches well with the value distances are converging against in Figure 1.

Also note, how random permutations of the same graph (isomorphic graphs) always resulted in a Euclidean distance of 0 and all graphs one editing step away (non-isomorphic) had a distance greater than 0.

Appendix E

Permutation Matrix As discussed in section 2.2 (Key architectural properties), we carefully designed our proposed model to make it invariant to permutations of the input graphs. By utilizing the permutation equivariant property of self-attention layers, the encoder and permuter model produce permutation matrices that equivalently permute with permutations in the input graph. In Figure 2, we show this for some example graphs (synthetic Barabási–Albert graphs). To better visually analyze the impact of the permutation of the input graph on the permutation matrix, we did the following: We first predicted permutation matrices for a set of random graphs (left hand side of Figure 2, note that we did not discretize these permutation matrices for this experiment) and then permuted the input graphs with permutation matrices that are defined by the transpose of these predicted permutation matrices. As a consequence, the predicted permutation matrices for these permuted graphs, are approximately (not discretized) equal to the identity matrix. This is exactly what we would expect. By permuting the input graphs by the transposed predicted permutation matrices, we brought the input graph in the canonical order learned by the decoder. Thus, no permutation is necessary and the permuter model predicts the identity matrix. Moreover, we can see how the permutation matrices equivalently permuted with the permutation of the input graph. A permutation of the input by $\mathbf{P}^T \mathcal{G}$ resulted in an equivalent permutation of the predicted permutation matrices $\mathbf{P}^T \mathbf{P} = \mathbb{1}$. Interestingly, in our experiments, this is not always the case. In Figure 2, image 1 shows one deviation from this property (not perfectly on the main diagonal). Inspecting the corresponding predicted permutation matrix (image 1 on the left hand side), we can see two entries (nodes) that are not assigned unambiguously,

resulting in bad approximation of a permutation matrix (which should only have one 1 in every row and column and zeros everywhere else). During inference, we would usually discretize such a permutation matrix to receive a valid permutation matrix. In fact, we find that for Barabási–Albert graphs with 20 nodes, 98 % of nodes have a value (confidence) assigned greater than 0.9. Thus, around 2 % of nodes cannot be assigned to the canonical order with a high confidence. Further investigation of these nodes has shown that they tend to have similar neighbourhoods. Approximately half of them are perfectly symmetric, meaning that they have the exact same neighbourhood, as Weisfeiler-Lehman graph hashing of their ego graphs revealed. Since very similar nodes (with respect to their neighbourhood) in a graph will receive similar node embeddings in the encoder model, the permuter model will score them similarly, resulting in ambiguous assignment in the predicted permutation matrix.

Still, graphs without such symmetric nodes, will receive a unambiguous permutation matrix. For such graphs we found in further experiments where we constructed for 10000 graphs 64 random permutations, that, as expected, the permutation matrix always permuted equivalently with the random permutation of the input graph.

Appendix F

Molecular graph datasets Molecular graphs were constructed utilizing following information for nodes:

- Atom type: 5 (C, N, O, F and H) for the QM9 dataset, 11 (C, N, O, F, S, Si, P, Cl, Br, I and H) for the PubChem dataset.
- Charge type: 3 (-1, 0, 1) for the QM9 dataset, 5 (-2, -1, 0, 1, 2) for the PubChem dataset.
- Ring membership

and following information for the edges:

- Bond type: 4 (single-, double-, triple- and aromatic bond) + 1 (no bond)
- Topological distance: normalized topological distance between two connected nodes.
- Euclidean Distance: For the QM9 dataset we transformed Cartesian coordinates of atoms to a normalized Euclidean distance matrix and used each entry as an additional edge feature.

References

- [1] Aric Hagberg, Pieter Swart, and Daniel S Chult. Exploring network structure, dynamics, and function using networkx. Technical report, Los Alamos National Lab.(LANL), Los Alamos, NM (United States), 2008.
- [2] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *arXiv preprint arXiv:1706.03762*, 2017.
- [3] Thomas Kipf and Max Welling. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308v*, 2016.

2.6 Publication 6: Unsupervised Learning of Group Invariant and Equivariant Representations

Full Reference: *Winter, Robin, Bertolini, Marco, Le, Tuan, Noé, Frank & Clevert, Djork-Arné (2022). Unsupervised Learning of Group Invariant and Equivariant Representations. Advances in Neural Information Processing Systems 35, 31942–31956*

DOI: 10.48550/arXiv.2202.07559

Licence: CC-BY

Journal/Conference: Neural Information Processing Systems (NeurIPS)
(h5-index: 309)

Source Code: <https://github.com/jrwnter/group-invariant-ae>

Paper's main contributions:

- We propose a novel method for learning group invariant representations. Our construction is by design disentangled in a an invariant and equivariant (group function) component.
- We characterize the mathematical conditions of the group function component and we propose an explicit construction suitable for any group G .
- We show in various experiments the validity and flexibility of our framework by learning representations of diverse data types with different network architectures.

Author's contribution to the paper:

- Conceptualization of the original idea and its application to different data types and groups.
- Development of the methodology and implementation.
- Design and evaluation of experiments, data curation and analysis.
- Preparation and creation of main parts of the initial draft and visualizations.

Unsupervised Learning of Group Invariant and Equivariant Representations

Robin Winter*
Bayer AG
Freie Universität Berlin
robin.winter@bayer.com

Marco Bertolini*
Bayer AG
marco.bertolini@bayer.com

Tuan Le
Bayer AG
Freie Universität Berlin
tuan.le2@bayer.com

Frank Noé
Freie Universität Berlin
frank.noe@fu-berlin.de

Djork-Arné Clevert
Bayer AG
djork-arne.clevert@bayer.com

Abstract

Equivariant neural networks, whose hidden features transform according to representations of a group G acting on the data, exhibit training efficiency and an improved generalisation performance. In this work, we extend group invariant and equivariant representation learning to the field of unsupervised deep learning. We propose a general learning strategy based on an encoder-decoder framework in which the latent representation is separated in an invariant term and an equivariant group action component. The key idea is that the network learns to encode and decode data to and from a group-invariant representation by additionally learning to predict the appropriate group action to align input and output pose to solve the reconstruction task. We derive the necessary conditions on the equivariant encoder, and we present a construction valid for any G , both discrete and continuous. We describe explicitly our construction for rotations, translations and permutations. We test the validity and the robustness of our approach in a variety of experiments with diverse data types employing different network architectures.

1 Introduction

An increasing body of work has shown that incorporating knowledge about underlying symmetries in neural networks as inductive bias can drastically improve the performance and reduce the amount of data needed for training Cohen & Welling (2016a); Bronstein et al. (2021). For example, the equivariant design with respect to the translation symmetry of objects in images proper of convolutional neural networks (CNNs) has revolutionized the field of image analysis LeCun et al. (1995). Message Passing neural networks, respecting permutation symmetries in graphs, have enabled powerful predictive models on graph-structured data Gilmer et al. (2017); Defferrard et al. (2016). Recently, much work has been done utilizing 3D rotation and translation equivariant neural networks for point clouds and volumetric data, showing great success in predicting molecular ground state energy levels with high fidelity Miller et al. (2020); Anderson et al. (2019); Klicpera et al. (2020); Schütt et al. (2021). Invariant models take advantage of the fact that often properties of interest, such as the class label of an object in an image or the ground state energy of a molecule, are invariant to certain group actions (e.g., translations or rotations), while the data itself is not (e.g., pixel values, atom coordinates).

There are several approaches to incorporate invariance into the learned representation of a neural network. The most common approach consists of teaching invariance to the model by data augmenta-

*equal contribution

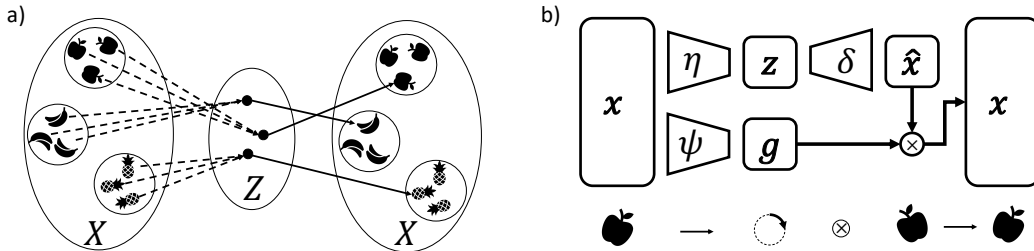


Figure 1: a) Schematic of the learning task this work is concerned with. Data points $x \in X$ are encoded to and decoded from latent space Z . Points in the same orbit in X are mapped to the same point (orbit) $z \in Z = X/G$. Latent points z are mapped to canonical elements $\hat{x} \in \{\rho_X(g)x | \forall g \in G\}$. b) Schematic of our proposed framework with data points x , encoding function η , decoding function δ , canonical elements \hat{x} , group function ψ and group action g

tion: during training, the model must learn that a group transformation on its input does not affect its label. While this approach can lead to improved generalization performance, it reduces training efficiency and quickly becomes impractical for higher dimensional data Thomas et al. (2018). A second technique, known as feature averaging, consists of averaging model predictions over group transformations of the input Puny et al. (2021). While feasible with finite groups, this method requires, for instance, sampling for infinite groups Lyle et al. (2020). A third approach is to impose invariance as a model architectural design. The simplest option is to restrict the function to be learned to be a composition of symmetric functions only Schütt et al. (2018). Such choice, however, can significantly restrict the functional form of the network. A more expressive variation of this approach consists of an equivariant neural network, followed by a symmetric function. This allows the network to leverage the benefits of invariance while having a larger capacity due to the less restrictive nature of equivariance. In fact, in many real-world application, equivariance is beneficial if not necessary Smidt (2020); Miller et al. (2020). For example, the interaction of a molecule (per se rotational invariant) with an external magnetic field is an intrinsically equivariant problem.

All aforementioned considerations require some sort of supervision to extract invariant representations from data. Unsupervised learning of group invariant representations, despite its potential in the field of representation learning, has been impaired by the fact that the representation of the data in general does not manifestly exhibit the group as a symmetry. For instance, in the case of an encoder-decoder framework in which the bottleneck layer is invariant, the reconstruction is only possible up to a group transformation. Nevertheless, the input data is typically parametrized in terms of coordinates in some vector space X , and the reconstruction task can only succeed by employing knowledge about the group action on X .

Following this line of thought, this work is concerned with the question: *Can we learn to extract both the invariant and equivariant representations of data in an unsupervised way?*

To this end, we introduce a group-invariant representation learning method that encodes data in a group-invariant latent code and a group action. By separating the embedding in a group-invariant and a group-equivariant part, we can learn expressive lower-dimensional group-invariant representations utilizing the power of autoencoders (AEs). In particular, all our results trivially extend to variational AEs (VAEs), thus advancing generative approaches of group-invariant data. We can summarize the main contributions of this work as follows:

- We introduce a novel framework for learning group equivariant representations. Our representations are *by construction* separated in an invariant and group action function (equivariant) component.
- We characterize the mathematical conditions of the group action function component and we propose an explicit construction suitable for *any* group G . To the best of our knowledge, this is the first method for unsupervised learning of separated invariant-equivariant representations valid for any group.
- We show in various experiments the validity and flexibility of our framework by learning representations of diverse data types with different network architectures.

2 Method

2.1 Background

We begin this section by introducing the basic concepts which will be central in our work.

A group G is a set equipped with an operation (here denoted \cdot) which is associative as well as having an identity element e and inverse elements. In the context of data, we are mainly interested in how groups represent geometric transformations by acting on spaces and, in particular, how they describe the symmetries of an object or of a set. In either case, we are interested in how groups act on spaces. This is represented by a **group action**: given a set X and a group G , a (left) action of G on X is a map $\rho : G \times X \rightarrow X$ such that it respects the group property of associativity and identity element. If X is a vector space, which we will assume for the remainder of the text, we refer to group actions of the form $\rho_X : G \rightarrow \text{GL}(X)$ as **representations** of G , where the general linear group of degree n $\text{GL}(X)$ is represented by the set of $n \times n$ invertible matrices. Given a group action, a concept which will play an important role in our discussion is given by the fixed points of such an action. Formally, given a point $x \in X$ and an action (representation) ρ_X of G on X , the **stabilizer** of G with respect to x is the subgroup $G_x = \{g \in G | \rho_X(g)x = x\} \subset G$.

In the context of representation learning, we assume our data to be defined as the space of representation-valued functions on some set V , i.e., $X = \{f | f : V \rightarrow W\}$. For instance, a point cloud in three dimensions can be represented as the set of functions $f : \mathbb{R}^3 \rightarrow \mathbb{Z}_2$, assigning to every point $\mathbf{r} \in \mathbb{R}^3$ the value $f(\mathbf{r}) = 0$ (the point is not included in the cloud) or $f(\mathbf{r}) = 1$ (the point is included in the cloud). Representations ρ_V of a group G on V can be extended to representations on f , and therefore on X , $\rho_X : G \rightarrow \text{GL}(X)$, as follows

$$[\rho_X(g)f](x) \equiv \rho_W(g)f(\rho_V(g^{-1})x). \quad (1)$$

In what follows, we will then only refer to representations for the space X , implicitly referring to equation (1) for mapping back to how the various components transform. A map $\varphi : V \rightarrow W$ is said to be **G -equivariant** with respect to the actions (representations) ρ_V, ρ_W if $\varphi(\rho_V(g)v) = \rho_W(g)\varphi(v)$ for every $g \in G$ and $v \in V$. Note that G -invariance is a particular case of the above, where we take ρ_V, ρ_W to be the trivial representations. An element $x \in X$ can be described in terms of a G -invariant component and a group element $g \in G$, as follows: let $\varphi_{\text{inv}} : X \rightarrow X/G$, be an invariant map mapping each element $x \in X$ to a corresponding canonical element \hat{x} in the orbit in the quotient space X/G . Then for each $x \in X$ there exist a $g \in G$ such that $x = \rho_X(g)\varphi_{\text{inv}}(x)$.

2.2 Problem Definition

We consider a classical autoencoder framework with encoding function $\eta : X \rightarrow Z$ and decoding function $\delta : Z \rightarrow X$, mapping between the data domain X , and latent domain Z , minimizing the reconstruction objective $d(\delta(\eta(x)), x)$, with a difference measure d (e.g., L_p norm). As discussed above, we wish to learn the invariant map η (φ_{inv} in the previous paragraph), thus

Property 2.1. *The encoding function $\eta : X \rightarrow Z$ is G -invariant, i.e., $\eta(\rho_X(g)x) = \eta(x) \forall x \in X, \forall g \in G$.*

The decoding function δ maps the G -invariant representation $z \in Z$ back to the data domain X . However, as z is G -invariant, δ can at best map $\eta(x) \in Z$ back to an element $\hat{x} \in X$ such that $\hat{x} \in \{\rho_X(g)x | \forall g \in G\}$, i.e., an element in the orbit of x through G . This is depicted in Figure 1a. Thus, the task of the decoding function $\delta : Z \rightarrow X$ is to map encoded elements $z = \eta(x) \in Z$ to an element $\hat{x} \in X$ such that $\exists \hat{g}_x \in G$ such that

$$\delta(\eta(x)) = \hat{x} = \rho_X(\hat{g}_x)x. \quad (2)$$

We call \hat{x} the **canonical** element of the decoder δ . We can rewrite the reconstruction objective with a G -invariant encoding function η as $d(\rho_X(\hat{g}^{-1})\delta(\eta(x)), x)$. One of the main results of this work consists in showing that \hat{x} and \hat{g}_x can be *simultaneously* learned by a suitable neural network. That is, we have the following property of our learning scheme:

Property 2.2. *There exists a **learnable** function $\psi : X \rightarrow G$ such that, given suitable η, δ as described above the relation $\rho_X(\psi(x))\delta(\eta(x)) = x$, holds for all $x \in X$.*

We call any function ψ satisfying (2.2) a **suitable** group function. Figure 1b describes schematically our proposed framework. In what follows, we will first characterize the defining properties of suitable group functions. Subsequently, we will describe our construction, valid for any group G .

2.3 Predicting Group Actions

In the following we further characterize the properties of ψ . We begin by stating two key results, while we refer to the Appendix A for the proofs.

Proposition 2.3. *Any suitable group function $\psi : X \rightarrow G$ is G -equivariant at a point $x \in X$ up the stabilizer G_x , i.e., $\psi(\rho_X(g)x) \subseteq g \cdot \psi(x)G_x$.*

Proposition 2.4. *The image of any suitable group function $\psi : X \rightarrow G$ is surjective into $\frac{G}{G_x}$, where G_x is the stabilizers of all the points of X .*

Let us briefly discuss an example. Suppose $X = \{x = (x_0, x_1, x_2, x_3) \in \mathbb{R}^{4 \times 2} | x_i = \rho_{\mathbb{R}^2}(g_{\theta=\pi/2})^i x_0, x_0 \in \mathbb{R}^2\}$ and $G = \text{SO}(2)$. X describes all collections of vertices of squares centered at the origin of \mathbb{R}^2 , and it is easy to check that $g_X = \mathbb{Z}_4$, generated by a $\pi/2$ rotation around the origin. In this case, any such square can be brought to any other square (of the same radius) by a rotation of an angle $\theta < \pi/2$, thus $\text{Im}\psi \supseteq \{g_\theta \in \text{SO}(2) | 0 \leq \theta \leq \pi/2\} = \text{SO}(2)/\mathbb{Z}_4$.

Combining the two propositions above we have the following

Lemma 2.5. *Any suitable group function ψ is an isomorphism $O_x \simeq G/G_x$ for any $x \in X$, where $O_x \subset X$ is the orbit of x with respect to G in X .*

2.4 Proposed Construction

Next, we turn to our proposed construction of a class of suitable group functions that satisfy Property 2.2 for any data space X and group G . As we described above, these functions must be learnable.

Property 2.6 (Proposed construction). *Without loss of generality, we write our target function $\psi = \xi \circ \mu$, where $\mu : X \rightarrow Y$ is a learnable map between the data space X and the embedding space Y , while $\xi : Y \rightarrow G$ is a deterministic map. Our construction is further determined by the following properties:*

- We impose $\mu : X \rightarrow Y$ to be G -equivariant, that is, $\mu(\rho_X(g)x) = \rho_Y(g)\mu(x)$ for all $x \in X$ and $g \in G$.
- We ask that Y is an homogeneous space, that is, given any element $y_0 \in Y$, every element $y \in Y$ can be written as $y = \rho_Y(g)y_0$ for some $g \in G$.
- The map $\xi : Y \rightarrow G$ is defined as follows: $\xi(y) = g$ such that $y = \rho_Y(g)y_0$ for any chosen point $y_0 \in Y$.

In what follows we will show that our construction satisfies the properties of the previous section. For proofs see Appendix. We begin with the following

Proposition 2.7. *Let $\psi = \xi \circ \mu$ be a suitable group function and let $\mu : X \rightarrow Y$ be G -equivariant. Then, $G_x = G_{\mu(x)}$ for all $x \in X$.*

The result of the above proposition is crucial for our desired decomposition of the learned embedding, as it ensures that no information about the group action on X is lost through the map μ : if a group element acts non-trivially in X , it will also act non-trivially in Y .

Proposition 2.8. *Given y, y_0 , the element g such that $y \equiv \rho_Y(g)y_0$ is unique up to the stabilizer G_{y_0} .*

This proposition establishes the equivariant properties of the map ξ . Finally, we have

Proposition 2.9. *Let $\psi = \xi \circ \mu$ where μ and ξ are as described above. Then, ψ is a suitable group function.*

2.5 Intuition Behind the Proposed Framework

We conclude this rather technical section with a comment on the intuition behind our construction. Assuming for simplicity that the domain set V admits the structure of vector space, Y represents the space spanned by **all** basis vectors of V . The point y_0 represent a canonical orientation of such basis, and the element $\xi(y) = g$ is the group element corresponding to a basis transformation. As

all elements can be expressed in terms of coordinates with respect to a given basis, it is natural to consider a canonical basis for all orbits, justifying the assumption of homogeneity of the space Y .

Further, let us assume that the invariant autoencoder correctly solves its task, $x \sim \delta(\eta(x))$. Now let $\hat{x} \in O_x$ such that $\hat{x} = \delta(\eta(x))$, and by definition, $\hat{x} = \rho_V(g)x$ for some $g \in G$. Now, the correct orbit element is identified when $\psi(\hat{x}) = e$, since $\psi(x) = g^{-1} \cdot \psi(\hat{x}) = g^{-1}$ and thus $\rho_X(g^{-1})\delta(\eta(x)) = \rho_X(g^{-1})x' = x$. Hence, during training ψ needs to learn which orbit elements are decoded as ‘‘canonical’’, i.e., without the need of an additional group transformation. To clarify, here ‘‘canonical’’ does not reflect any specific property of the element, but it simply refers to the orientation learned from the decoder during training. In fact, different decoder architectures or initializations will lead to different canonical elements.

Finally, note how the different parts of our proposed framework, as visualized in Figure 1b, can be jointly trained by minimizing the objective $d(\rho_X(\psi(x))\delta(\eta(x)), x)$, which is *by construction* group invariant, i.e., not susceptible to potential group-related bias in the data (e.g. data that only occurs in certain orientations).

3 Application to Common Groups

In this section we describe how our framework applies to a variety of common groups which we will then implement in our experiments. As discussed in Section 2.2 and visualized in Figure 1b, the main components of our proposed framework are the encoding function η , the decoding function δ and the group function ψ . As stated in Property 2.1, the only constraint for the encoding function η is that it has to be group invariant. This is in general straightforward to achieve for different groups as we will demonstrate in Section 5. Our proposed framework does not constrain the decoding function δ other than that it has to map elements from the latent space Z to the data domain X . Hence, δ can be designed independently of the group of interest. The main challenge is in defining the group function $\psi = \xi \circ \mu$ such that it satisfies Property 2.2. Following Property 2.6 we now turn to describing our construction of ξ , μ and Y for a variety of common groups.

Orthogonal group $\text{SO}(2)$. The Lie group $\text{SO}(2)$ is defined as the set of all rotations about the origin in \mathbb{R}^2 . We take Y to be the circle $S^1 \subset \mathbb{R}^2$, that is, the space spanned by unit vectors in \mathbb{R}^2 . Now, S^1 is a homogeneous space: any two points $s_0, s_1 \in S^1$ are related by a rotation. Without loss of generality, we take the reference vector y_0 to be the vector $(1, 0) \in S^1$. Then given a vector $y \in S^1$, we can write

$$y = \begin{pmatrix} y_x \\ y_y \end{pmatrix} = \begin{pmatrix} y_x & -y_y \\ y_y & y_x \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix}. \quad (3)$$

thus, the function $\xi : S^1 \rightarrow \text{SO}(2)$ is determined by $\xi(y) = g_\theta$ such that $\theta = \arccos(y_x) = \arcsin(y_y)$.

Orthogonal group $\text{SO}(3)$. We assume that X has no fixed points, as this is usually the case for generic shapes (point clouds) in \mathbb{R}^3 . It would be tempting to take Y to be the sphere $S^2 \subset \mathbb{R}^3$, that is, the space spanned by unit vectors in \mathbb{R}^3 . While this space is homogeneous, it does not satisfy the condition that the stabilizers of G are trivial. In fact, given any vector $y_1 \in S^2$, we have $G_{y_1} = \{g \in \text{SO}(3) | g \text{ is a rotation about } y_1\}$.

In order to construct a space with the desired property, consider a second vector $w_2 \in S^2$ orthogonal to y_1 , $y_2 \in y_1^\perp$. Taking Y to be the space spanned by $y_1, y_2 \in S^2$, it is easy to see that now all the stabilizers are trivial. Finally, let $y_3 = y_1 \times y_2 \in S^2$, then we construct the rotation matrix

$$R = \begin{pmatrix} y_{1,x} & y_{2,x} & y_{3,x} \\ y_{1,y} & y_{2,y} & y_{3,y} \\ y_{1,z} & y_{2,z} & y_{3,z} \end{pmatrix}, \text{ which satisfies } \begin{pmatrix} y_1 \\ y_2 \end{pmatrix} = R \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}^\top = Ry_0.$$

Symmetric group S_n . A suitable space Y is the set of ordered collections of unique elements of the set $M = \{1, 2, \dots, n\}$. For instance, for $n = 3$, we have $Y = \{(1, 2, 3), (1, 3, 2), (2, 1, 3), (2, 3, 1), (3, 1, 2), (3, 2, 1)\}$. It is trivial to see that the action of the permutation group on the set Y is free, that is, all the stabilizers are trivial. Explicitly, given any permutation-equivariant vector $w \in \mathbb{R}^n$, we obtain an element $y = \text{argsort}(w) \in Y$. Moreover, it is

also obvious that any element in Y can be written as $P_\sigma(1, 2, \dots, n) = (\sigma(1), \sigma(2), \dots, \sigma(n))$, that is, a group element acting on the canonical $y_0 = (1, 2, \dots, n)$.

Translation group T_n . Here we take $Y = \mathbb{R}^n$, which is homogeneous with respect to the translation group. In fact, any vector $y \in Y$ can be trivially written as $y = y + \mathbf{0} = y + y_0$, where $\mathbf{0}$ is the origin of \mathbb{R}^n . Our group function takes therefore the form $\xi(y) = y$.

Euclidean group $\text{SE}(n)$. A generic transformation of the Euclidean group on a n -dimensional representation $v \in V$ is

$$v \mapsto Av + b, \quad A \in \text{SO}(n), b \in T_n. \quad (4)$$

Let $\mu = (\mu_1, \mu_2, \dots, \mu_{n+1})$ be a collection of $n + 1$ n -dimensional $\text{SE}(n)$ -equivariant vectors, that is, $\mu_i(\rho_X(g)x) = \rho_Y(g)\mu(x)$, $i = 1, \dots, n$. We construct $\hat{y}_a = (\mu_a - \mu_{n+1}) / \|\mu_a - \mu_{n+1}\| \in S^n$, $a = 1, \dots, n$, where S^n is the unit n -dimensional sphere. These n ortho-normal vectors are **translation invariant** but rotation equivariant, and are suitable to construct the rotation matrix

$$R = (\hat{y}_1 \quad \hat{y}_2 \quad \dots \quad \hat{y}_n), \quad (5)$$

while the extra vector $\hat{y}_{n+1} = \mu_{n+1}$ can be used to predict the translation action. Putting all together, the space Y is described by n vectors $y_a = \hat{y}_a + \hat{y}_{n+1}$, and $y_0 = I_n$ is the $n \times n$ unit matrix, as

$$(R + \hat{y}_{n+1})I_n = (\hat{y}_1 \quad \dots \quad \hat{y}_n)^\top + \hat{y}_{n+1}I_n = (y_1 \quad \dots \quad y_n)^\top. \quad (6)$$

4 Related Work

Group equivariant neural networks. Group equivariant neural networks have shown great success for various groups and data types. There are two main approaches to implement equivariance in a layer and, hence, in a neural network. The first, and perhaps the most common, imposes equivariance on the space of functions and features learned by the network. Thus, the parameters of the model are constrained to satisfy equivariance Thomas et al. (2018); Weiler & Cesa (2019a); Weiler et al. (2018a); Esteves et al. (2020). The disadvantage of this approach consists in the difficulty of designing suitable architectures for all components of the model, transforming correctly under the group action Xu et al. (2021). The second approach to equivariance consists in lifting the map from the space of features to the group G , and equivariance is defined on functions on the group itself Romero & Hoogendoorn (2020); Romero et al. (2020); Hoogeboom et al. (2018). Although this strategy avoids the architectural constraints, applicability is limited to homogeneous spaces Hutchinson et al. (2021) and involves an increased dimensionality of the feature space, due to the lifting to G . Equivariance has been explored in a variety of architecture and data structures: Convolutional Neural Networks Cohen & Welling (2016a); Worrall et al. (2017); Weiler et al. (2018c); Bekkers et al. (2018); Thomas et al. (2018); Dieleman et al. (2016); Kondor & Trivedi (2018); Cohen & Welling (2016b); Cohen et al. (2018); Finzi et al. (2020), Transformers Vaswani et al. (2017); Fuchs et al. (2020); Hutchinson et al. (2021); Romero & Cordonnier (2020), Graph Neural Networks Defferrard et al. (2016); Bruna et al. (2013); Kipf & Welling (2016); Gilmer et al. (2017); Satorras et al. (2021) and Normalizing Flows Rezende & Mohamed (2015); Köhler et al. (2019, 2020); Boyda et al. (2021). These methods are usually trained in a supervised manner and combined with a symmetric function (e.g. pooling) to extract group-invariant representations.

Group equivariant autoencoders. Another line of related work is concerned with group equivariant autoencoders. Such models utilize specific network architectures to encode and decode data in an equivariant way, resulting into equivariant representations only Hinton et al. (2011); Sabour et al. (2017); Kosiorek et al. (2019); Guo et al. (2019). Feige (2019) use weak supervision in an AE to extract invariant and equivariant representations. Winter et al. (2021) implement a permutation-invariant AE to learn graph embeddings, in which the permutation matrix for graph matching is learned during training. In that sense, the present work can be seen as a generalization of their approach for a generic data type and any group.

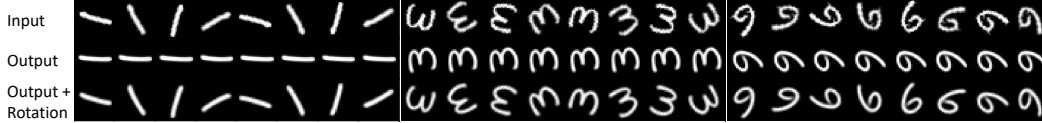


Figure 2: Input and predicted output for rotated versions of three MNIST images. Top row shows the input image successively rotated by 45° . Middle row shows the decoded (canonical) image and bottom row shows the decoded image after applying the predicted rotation.

5 Experiments

In this section we present a diverse set of experiments for the various symmetry groups discussed in Section 3.²

5.1 Rotated MNIST

In the first experiment, we train an $SO(2)$ -invariant autoencoder on the original (non-rotated) MNIST dataset and validate the trained model on the rotated MNIST dataset mni which consists of randomly rotated versions of the original MNIST dataset. For the functions η and ψ we utilize $SO(2)$ -Steerable Convolutional Neural Networks Weiler & Cesa (2019b). For more details about the network architecture and training, we refer to Appendix B. In Figure 2 we show images in different rotations and the respective reconstructed images by the trained model. The model decodes the different rotated versions of the same image (i.e., elements from the same orbit) to the same canonical output orientation (second row in Figure 2). The trained model manages to predict the right rotation matrix (group action) to align the decoded image with the input image, resulting in an overall low reconstruction error. Note that the model never saw rotated images during training but still manages to encode and reconstruct them due to its inherent equivariant design. We find that the encoded latent representation is indeed rotation invariant (up to machine precision), but only for rotations of an angle $\theta = \frac{n\pi}{2}$, $n \in \mathbb{N}$. For all other rotations, we see slight variations in the latent code, which, however, is to be expected due to interpolation artifacts for rotations on a discretized grid. Still, inspecting the 2d-projection of the latent code of our proposed model in Figure 3, we see distinct clusters for each digit class for the different images from the test dataset, independent of the orientation of the digits in the images. In contrast, the latent code of a classical autoencoder exhibits multiple clusters for different orientations of the same digit class.

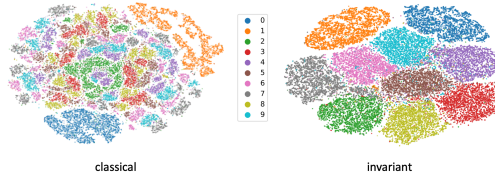


Figure 3: TSNE embedding of the encoded test dataset for a classical and our proposed $SO(2)$ -invariant autoencoder.

5.2 Set of Digits

Next, we train a permutation-invariant autoencoder on sets of digits. A set with N digits is represented by concatenating one-hot vectors of each digit in a $N \times D$ -dimensional matrix, where we take $D = 10$. Notice that this matrix-representation of a set is *not* permutation invariant. We randomly sampled 1.000.000 different sets for training and 100.000 for the final evaluation with $N = 20, 30, 40, 100$, respectively, removing all permutation equivariant sets (i.e., there are no two sets that are the same up to a permutation). For comparison, we additionally trained a classical non-permutation-invariant autoencoder with the same number of parameters and layers as our permutation-invariant version. For more details on the network architecture and training we refer to Appendix C. Here, we demonstrate how the separation of the permutation-invariant information of the set (i.e., the composition of the set) from the (irrelevant) order-information results in a significant reduction of the space needed to encode the set. In Figure 4a, we plot the element-wise reconstruction accuracy of different sized sets for both

²Source code for the different implementations is publicly available at <https://github.com/>... (also see Supplementary Information)

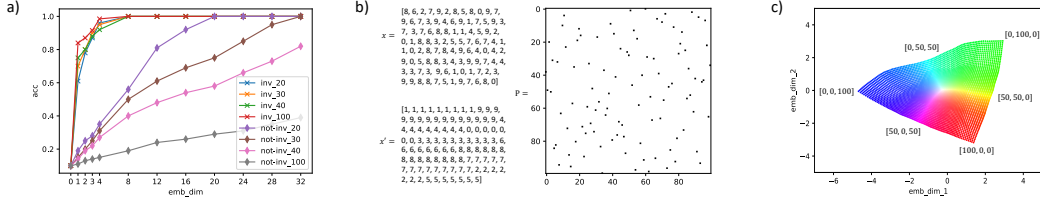


Figure 4: a) Element-wise reconstruction accuracy of our proposed permutation invariant autoencoder (cross) and a classical non-permutation invariant autoencoder (diamond) for different embedding and set sizes. b) Example set x with 100 elements with its canonical reconstruction \hat{x} and the predicted permutation matrix P (resulting into a perfect reconstruction). One can confirm for oneself that, e.g., $x[38] = x'[0]$, matching $P[38, 0] = 1$. c) Best viewed in colour. Visualization of the two-dimensional embedding of a permutation-invariant autoencoder for all 5151 sets of 100 elements with 3 different element classes. Each point represents one set, colours represent set compositions (proportion of each element class, independent of the order). The three element classes are represented by the three colour channels (rgb). E.g., the higher the proportion of the third element class, the bluer the point, while same proportions results into white (point in the center).

models for varying embedding (bottleneck) sizes. As the classical autoencoder has to store both the composition of digits in the set (i.e., number of elements for each of the 10 digits classes) as well as their order in the permutation-dependent matrix representation, the reconstruction accuracy drops for increasing size of the set N for a fixed embedding size. For the same reason, perfect reconstruction accuracy is only achieved if the embedding dimension is at least as large as the number of digits in the set. On the contrary, our proposed permutation invariant autoencoder achieves perfect reconstruction accuracy with a significant lower embedding size. Crucially, as no order information has to be stored in the embedding, this embedding size for perfect reconstruction accuracy also stays the same for increasing size N of the set. In Figure 4b we show one example for a set x with $N = 100$ digits, with the predicted canonical orbit element \hat{x} and the predicted permutation matrix. As perhaps expected, the canonical element clusters together digits with same value, while not using the commonly used order of Arabic numerals. This learned order (here $[1, 9, 4, 0, 3, 6, 8, 7, 2, 5]$) stays fixed for the trained network for different inputs but changes upon re-initialization of the network.

In Figure 4c we show the two-dimensional embedding of a permutation invariant autoencoder trained on set of $N = 100$ elements chosen from $D = 3$ different classes (e.g. digits 0,1,2). As the sets only consists of 3 different elements (but in different compositions and order) we can visualize the $\binom{D+N-1}{N} = \binom{102}{100} = 5151$ elements in the two-dimensional embedding and colour them according to their composition. As our proposed autoencoder only needs to store the information about the set composition and not the order, the embedding is perfectly structured with respect to the composition as can be seen by the colour gradients in the visualization of the embedding.

5.3 Point Cloud

Point clouds are a common way to describe objects in 3D space, such as the atom positions of a molecule or the surface of an object. As such, they usually adhere to 3D translation and rotation symmetries and are unordered, i.e., permutation invariant. Hence, we investigate in the next experiment a combined $SE(3)$ - and S_N -invariant autoencoder for point cloud data. We use the Tetris Shape toy dataset Thomas et al. (2018) which consists of 8 shapes, where each shape includes $N = 4$ points in 3D space, representing the center of each Tetris block. To generate various shapes, we augment the 8 shapes by adding Gaussian noise with $\sigma = 0.01$ standard deviation on each node’s position. Different orientations are obtained by rotating the point cloud with a random rotation matrix $R \in SO(3)$ and further translating all node positions with the same random translation vector $t \in \mathbb{R}^3 \simeq T_3$. For additional details on the network architecture and training we refer to Appendix D. In Figure 5 we visualize the input points and output points before and after applying the predicted rotation. The model successfully reconstructs the input points with high fidelity (mean squared error of $\sim 4 \times 10^{-5}$) for all shapes and arbitrary translations and rotations. Figure 5b shows the two-dimensional embedding of the trained $SE(3)$ - and S_N -invariant autoencoder. Augmenting the points with random noise results into slight variations in the embedding, while samples of the same Tetris shape class still cluster together. The embedding is invariant with respect to rotations, translation and permutations

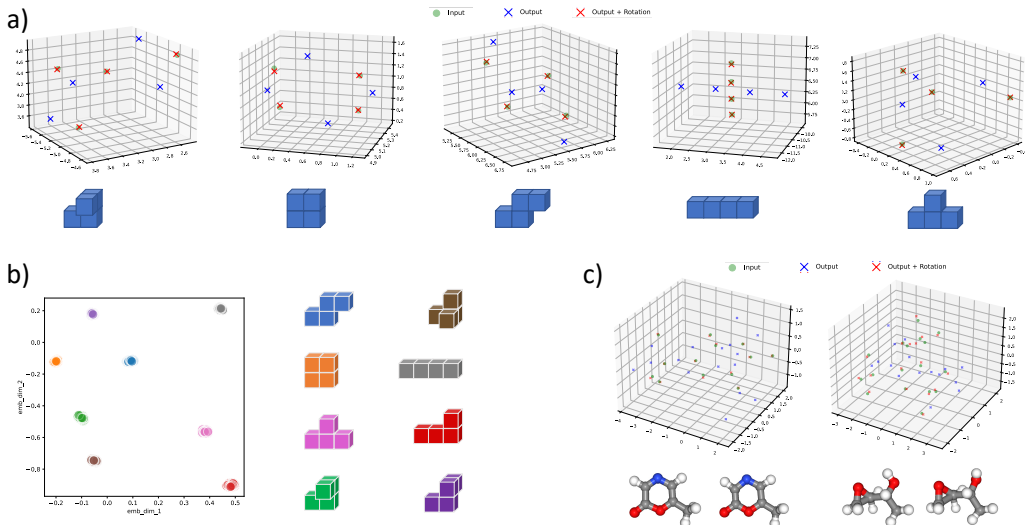


Figure 5: a) Five different Tetris shapes represented by points at the center of the four blocks respectively. Input points, output points and rotated (predicted group action) output points as reconstructed by our proposed SE(3)- and S(N)-invariant autoencoder are visualized. b) Two-dimensional latent space for all Tetris shapes augmented with Gaussian noise ($\sigma = 0.01$). Colors of points match colors of shapes on the right. c) Two molecular conformations and their reconstructions represented as point cloud and ball-and-stick model (left true, right predicted).

We only showcase the effect of aligning by applying the predicted rotation matrix, while the models are still trained in predicting the permutation matrix and translation vector to perform the alignment.

of the points. Notably, the SE(3)-invariant representations can distinguish the two chiral shapes (compare green and violet coloured shapes in the bottom right of Figure 5b). These two shapes are mirrored versions of themselves and should be distinguished in an SE(3) equivariant model. Models that achieve SE(3) invariant representations by restricting themselves to composition of symmetric functions only, such as working solely on distances (e.g. SchNet Schütt et al. (2018)) or angles (e.g. ANI-1 Smith et al. (2017)) between points fail to distinguish these two shapes Thomas et al. (2018).

Molecular Conformations. We showcase our learning framework on real-world data by autoencoding the atom types and geometries of small molecules from the QM9 database Ramakrishnan et al. (2014). We achieved a reconstruction RMSE of $0.15 \pm 0.07 \text{ \AA}$ for atom coordinates and perfect atom type accuracy on 5000 unseen test conformations (see Figure 5c for two examples and Appendix E for more reconstruction predictions). Given a point cloud of N nodes, notice that the G -inv. embedding z has to store information about the Cartesian coordinates $P \in \mathbb{R}^{3N}$ as well as the 5 distinct atom types $A \in \{0, 1\}^{5N}$ represented as one-hot encodings. Given the largest molecule in the QM9 database with $N_{\max} = 29$ atoms, the degrees of freedom of the data space³ X can be assumed to be $3 \cdot 29 \cdot 5 \cdot 29 = 12615$, making the input data high-dimensional, while we compress the molecular conformations into $z \in Z \subset \mathbb{R}^{256}$ dimensions.

6 Conclusion

In this work we proposed a novel unsupervised learning strategy to extract representations from data that are separated in a group invariant and equivariant part for any group G . We defined the sufficient conditions for the different parts of our proposed framework, namely the encoder, decoder and group function without further constraining the choice of a (G -) specific network architecture. In fact, we demonstrate the validity and flexibility of our proposed framework for diverse data types, groups and network architectures.

To the best of our knowledge, we propose the first general framework for unsupervised learning of separated invariant-equivariant representations valid for any group. Our learning strategy can

³Notice that the data space X can be described as the product space between \mathbb{R}^{3N} and \mathbb{N}^{5N} .

be applied to any AE framework, including variational AEs. It would be compelling to extend our approach to a fully probabilistic approach, where the group action function samples from a probability distribution. Such formalism would be relevant in scenarios where some elements of a group orbit occur with different frequencies, enabling this to be reflected in the generation process. For instance, predicting protein-ligand binding sites depends on the molecule's orientation with respect to the protein pocket or cavity. Thus, in a generative approach, it would be highly compelling to generate a group action reflecting a candidate molecule's orientation in addition to a candidate ligand. We plan to return to these generalization and apply our learning strategy to non-trivial real-world applications in future work.

References

- Rotated MNIST. https://sites.google.com/a/lisa.iro.umontreal.ca/public_static_twiki/variations-on-the-mnist-digits. [Online; accessed 05-January-2021].
- Anderson, B., Hy, T.-S., and Kondor, R. Cormorant: Covariant molecular neural networks. *arXiv preprint arXiv:1906.04015*, 2019.
- Averkiou, M., Kim, V. G., and Mitra, N. J. Autocorrelation descriptor for efficient co-alignment of 3d shape collections. *Computer Graphics Forum*, 35, 2016.
- Axelrod, S. and Gomez-Bombarelli, R. GEOM, 2021. URL <https://doi.org/10.7910/DVN/JNGTDF>.
- Bekkers, E. J., Lafarge, M. W., Veta, M., Eppenhof, K. A., Pluim, J. P., and Duits, R. Roto-translation covariant convolutional networks for medical image analysis. In *International conference on medical image computing and computer-assisted intervention*, pp. 440–448. Springer, 2018.
- Boyda, D., Kanwar, G., Racanière, S., Rezende, D. J., Albergo, M. S., Cranmer, K., Hackett, D. C., and Shanahan, P. E. Sampling using su(n) gauge equivariant flows. *Physical Review D*, 103(7): 074504, 2021.
- Bronstein, M. M., Bruna, J., Cohen, T., and Velicković, P. Geometric deep learning: Grids, groups, graphs, geodesics, and gauges. *arXiv preprint arXiv:2104.13478*, 2021.
- Bruna, J., Zaremba, W., Szlam, A., and LeCun, Y. Spectral networks and locally connected networks on graphs. *arXiv preprint arXiv:1312.6203*, 2013.
- Bunke, H. and Jiang, X. *Graph Matching and Similarity*, pp. 281–304. Springer US, Boston, MA, 2000. ISBN 978-1-4615-4401-2. doi: 10.1007/978-1-4615-4401-2_10. URL https://doi.org/10.1007/978-1-4615-4401-2_10.
- Chaouch, M. and Verroust-Blondet, A. A novel method for alignment of 3d models. *2008 IEEE International Conference on Shape Modeling and Applications*, pp. 187–195, 2008.
- Chaouch, M. and Verroust-Blondet, A. Alignment of 3d models. *Graph. Model.*, 71:63–76, 2009.
- Cohen, T. and Welling, M. Group equivariant convolutional networks. In *International conference on machine learning*, pp. 2990–2999. PMLR, 2016a.
- Cohen, T., Geiger, M., and Weiler, M. A general theory of equivariant cnns on homogeneous spaces. *arXiv preprint arXiv:1811.02017*, 2018.
- Cohen, T. S. and Welling, M. Steerable cnns. *arXiv preprint arXiv:1612.08498*, 2016b.
- Defferrard, M., Bresson, X., and Vandergheynst, P. Convolutional neural networks on graphs with fast localized spectral filtering. *Advances in neural information processing systems*, 29:3844–3852, 2016.
- Dieleman, S., De Fauw, J., and Kavukcuoglu, K. Exploiting cyclic symmetry in convolutional neural networks. In *International conference on machine learning*, pp. 1889–1898. PMLR, 2016.
- Ding, J., Ma, Z., Wu, Y., and Xu, J. Efficient random graph matching via degree profiles. *Probability Theory and Related Fields*, 179:29–115, 2020.
- Esteves, C., Makadia, A., and Daniilidis, K. Spin-weighted spherical cnns. *ArXiv*, abs/2006.10731, 2020.
- Feige, I. Invariant-equivariant representation learning for multi-class data, 2019.
- Finzi, M., Stanton, S., Izmailov, P., and Wilson, A. G. Generalizing convolutional neural networks for equivariance to lie groups on arbitrary continuous data. In III, H. D. and Singh, A. (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 3165–3176. PMLR, 13–18 Jul 2020. URL <https://proceedings.mlr.press/v119/finzi20a.html>.

- Fuchs, F. B., Worrall, D. E., Fischer, V., and Welling, M. Se (3)-transformers: 3d roto-translation equivariant attention networks. *arXiv preprint arXiv:2006.10503*, 2020.
- Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., and Dahl, G. E. Neural message passing for quantum chemistry. In *International conference on machine learning*, pp. 1263–1272. PMLR, 2017.
- Grover, A., Wang, E., Zweig, A., and Ermon, S. Stochastic optimization of sorting networks via continuous relaxations. *arXiv preprint arXiv:1903.08850*, 2019.
- Guo, X., Zhu, E., Liu, X., and Yin, J. Affine equivariant autoencoder. In *IJCAI*, pp. 2413–2419, 2019.
- Higgins, I., Amos, D., Pfau, D., Racanière, S., Matthey, L., Rezende, D. J., and Lerchner, A. Towards a definition of disentangled representations. *ArXiv*, abs/1812.02230, 2018.
- Hinton, G. E., Krizhevsky, A., and Wang, S. D. Transforming auto-encoders. In *International conference on artificial neural networks*, pp. 44–51. Springer, 2011.
- Hoogeboom, E., Peters, J. W. T., Cohen, T., and Welling, M. Hexaconv. *ArXiv*, abs/1803.02108, 2018.
- Hosoya, H. Group-based learning of disentangled representations with generalizability for novel contents. In *IJCAI*, 2019.
- Hutchinson, M. J., Le Lan, C., Zaidi, S., Dupont, E., Teh, Y. W., and Kim, H. Lietransformer: Equivariant self-attention for lie groups. In *International Conference on Machine Learning*, pp. 4533–4543. PMLR, 2021.
- Keurti, H., Pan, H.-R., Besserve, M., Grewe, B. F., and Scholkopf, B. Homomorphism autoencoder - learning group structured representations from observed transitions. *ArXiv*, abs/2207.12067, 2022.
- Kipf, T. N. and Welling, M. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.
- Klicpera, J., Groß, J., and Günnemann, S. Directional message passing for molecular graphs. *arXiv preprint arXiv:2003.03123*, 2020.
- Köhler, J., Klein, L., and Noé, F. Equivariant flows: sampling configurations for multi-body systems with symmetric energies. *arXiv preprint arXiv:1910.00753*, 2019.
- Köhler, J., Klein, L., and Noé, F. Equivariant flows: exact likelihood generative learning for symmetric densities. In *International Conference on Machine Learning*, pp. 5361–5370. PMLR, 2020.
- Kondor, R. and Trivedi, S. On the generalization of equivariance and convolution in neural networks to the action of compact groups. In *International Conference on Machine Learning*, pp. 2747–2755. PMLR, 2018.
- Koneripalli, K., Lohit, S., Anirudh, R., and Turaga, P. K. Rate-invariant autoencoding of time-series. *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 3732–3736, 2020.
- Kosiorsek, A. R., Sabour, S., Teh, Y. W., and Hinton, G. E. Stacked capsule autoencoders. *arXiv preprint arXiv:1906.06818*, 2019.
- LeCun, Y., Bengio, Y., et al. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 1995.
- Li, Y., Gu, C., Dullien, T., Vinyals, O., and Kohli, P. Graph matching networks for learning the similarity of graph structured objects. *ArXiv*, abs/1904.12787, 2019.
- Lyle, C., van der Wilk, M., Kwiatkowska, M. Z., Gal, Y., and Bloem-Reddy, B. On the benefits of invariance in neural networks. *ArXiv*, abs/2005.00178, 2020.
- Mehr, E., Lieutier, A., Bermudez, F. S., Guitteny, V., Thome, N., and Cord, M. Manifold learning in quotient spaces. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 9165–9174, 2018a.

- Mehr, , Lieutier, A., Bermudez, F. S., Guitteny, V., Thome, N., and Cord, M. Manifold learning in quotient spaces. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9165–9174, 2018b. doi: 10.1109/CVPR.2018.00955.
- Miller, B. K., Geiger, M., Smidt, T. E., and Noé, F. Relevance of rotationally equivariant convolutions for predicting molecular properties. *arXiv preprint arXiv:2008.08461*, 2020.
- Prillo, S. and Eisenschlos, J. Softsort: A continuous relaxation for the argsort operator. In *International Conference on Machine Learning*, pp. 7793–7802. PMLR, 2020.
- Puny, O., Atzmon, M., Ben-Hamu, H., Misra, I., Grover, A., Smith, E. J., and Lipman, Y. Frame averaging for invariant and equivariant network design, 2021. URL <https://arxiv.org/abs/2110.03336>.
- Ramakrishnan, R., Dral, P. O., Rupp, M., and von Lilienfeld, O. A. Quantum chemistry structures and properties of 134 kilo molecules. *Scientific Data*, 1, 2014.
- Rezende, D. and Mohamed, S. Variational inference with normalizing flows. In *International conference on machine learning*, pp. 1530–1538. PMLR, 2015.
- Romero, D. W. and Cordonnier, J.-B. Group equivariant stand-alone self-attention for vision. *arXiv preprint arXiv:2010.00977*, 2020.
- Romero, D. W. and Hoogendoorn, M. Co-attentive equivariant neural networks: Focusing equivariance on transformations co-occurring in data. *ArXiv*, abs/1911.07849, 2020.
- Romero, D. W., Bekkers, E. J., Tomczak, J. M., and Hoogendoorn, M. Attentive group equivariant convolutional networks. *ArXiv*, abs/2002.03830, 2020.
- Sabour, S., Frosst, N., and Hinton, G. E. Dynamic routing between capsules. *arXiv preprint arXiv:1710.09829*, 2017.
- Satorras, V. G., Hoogeboom, E., and Welling, M. E(n) equivariant graph neural networks, 2021.
- Schütt, K. T., Sauceda, H. E., Kindermans, P.-J., Tkatchenko, A., and Müller, K.-R. SchNet—a deep learning architecture for molecules and materials. *The Journal of Chemical Physics*, 148(24): 241722, 2018.
- Schütt, K. T., Unke, O. T., and Gastegger, M. Equivariant message passing for the prediction of tensorial properties and molecular spectra, 2021.
- Shu, Z., Sahasrabudhe, M., Güler, R. A., Samaras, D., Paragios, N., and Kokkinos, I. Deforming autoencoders: Unsupervised disentangling of shape and appearance. *ArXiv*, abs/1806.06503, 2018.
- Smidt, T. Euclidean symmetry and equivariance in machine learning. *ChemRxiv*, 2020. doi: 10.26434/chemrxiv.12935198.v1.
- Smith, J. S., Isayev, O., and Roitberg, A. E. Ani-1: an extensible neural network potential with dft accuracy at force field computational cost. *Chemical science*, 8(4):3192–3203, 2017.
- Thomas, N., Smidt, T., Kearnes, S., Yang, L., Li, L., Kohlhoff, K., and Riley, P. Tensor field networks: Rotation- and translation-equivariant neural networks for 3d point clouds, 2018.
- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. In *Advances in neural information processing systems*, pp. 5998–6008, 2017.
- Wang, Y., Asafi, S., van Kaick, O. M., Zhang, H., Cohen-Or, D., and Chen, B. Active co-analysis of a set of shapes. *ACM Transactions on Graphics (TOG)*, 31:1 – 10, 2012.
- Weiler, M. and Cesa, G. General e(2)-equivariant steerable cnns. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019a. URL <https://proceedings.neurips.cc/paper/2019/file/45d6637b718d0f24a237069fe41b0db4-Paper.pdf>.

- Weiler, M. and Cesa, G. General $e(2)$ -equivariant steerable cnns. *arXiv preprint arXiv:1911.08251*, 2019b.
- Weiler, M., Geiger, M., Welling, M., Boomsma, W., and Cohen, T. 3d steerable cnns: Learning rotationally equivariant features in volumetric data. In *NeurIPS*, 2018a.
- Weiler, M., Geiger, M., Welling, M., Boomsma, W., and Cohen, T. 3d steerable cnns: Learning rotationally equivariant features in volumetric data, 2018b.
- Weiler, M., Hamprecht, F. A., and Storath, M. Learning steerable filters for rotation equivariant cnns. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 849–858, 2018c.
- Winter, R., Noé, F., and Clevert, D.-A. Permutation-invariant variational autoencoder for graph-level representation learning. *arXiv preprint arXiv:2104.09856*, 2021.
- Worrall, D. E., Garbin, S. J., Turmukhambetov, D., and Brostow, G. J. Harmonic networks: Deep translation and rotation equivariance. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5028–5037, 2017.
- Xu, J., Kim, H., Rainforth, T., and Teh, Y. W. Group equivariant subsampling, 2021.
- Zaheer, M., Kottur, S., Ravanbakhsh, S., Poczos, B., Salakhutdinov, R., and Smola, A. Deep sets. *arXiv preprint arXiv:1703.06114*, 2017.

Checklist

1. For all authors...
 - (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
 - (b) Did you describe the limitations of your work? [Yes]
 - (c) Did you discuss any potential negative societal impacts of your work? [N/A]
 - (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
 - (a) Did you state the full set of assumptions of all theoretical results? [Yes]
 - (b) Did you include complete proofs of all theoretical results? [Yes]
3. If you ran experiments...
 - (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes]
 - (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]
 - (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]
 - (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
 - (a) If your work uses existing assets, did you cite the creators? [Yes]
 - (b) Did you mention the license of the assets? [N/A]
 - (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
 - (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
 - (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
 - (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
 - (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
 - (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

A Proofs

Proposition A.1. Any suitable group function $\psi : X \rightarrow G$ is G -equivariant at a point $x \in X$ up to the stabilizer G_x , i.e., $\psi(\rho_X(g)x) \subseteq g \cdot \psi(x)G_x$.

Proof: As the relation (see Property 2.2)

$$\rho_X(\psi(x))\delta(\eta(x)) = x \quad (7)$$

must hold for any $x \in X$, it must hold for any point $x' = \rho_X(g)x$ in the orbit of x , which then reads

$$\begin{aligned} x' &= \rho_X(\psi(x'))\delta(\eta(x')) \\ &= \rho_X(\psi(\rho_X(g)x))\delta(\eta(x)) , \end{aligned} \quad (8)$$

where we used the invariance of η . On the other hand, applying $\rho_X(g)$ to both sides of (7) we have

$$\begin{aligned} \rho_X(g)x &= \rho_X(g)\rho_X(\psi(x))\delta(\eta(x)) \\ &= \rho_X(g\psi(x))\delta(\eta(x)) , \end{aligned} \quad (9)$$

since $\eta(\rho_X(g')x) = \eta(x)$ and $\rho_X(g_1)\rho_X(g_2) = \rho_X(g_1g_2)$. Combining (8) and (9) it follows that

$$\rho_X(\psi(x)^{-1} \cdot g^{-1} \cdot \psi(\rho_X(g)x))\delta(\eta(x)) = \delta(\eta(x)) , \quad (10)$$

that is, $\psi(x)^{-1} \cdot g^{-1} \cdot \psi(\rho_X(g)x) \in G_{\delta(\eta(x))}$. Now, since x and $\delta(\eta(x))$ by assumption belong to the same orbit of G , it follows that they have isomorphic stabilizers, $G_{\delta(\eta(x))} \simeq G_x$. Thus, we have shown that $\psi(\rho_X(g)x) = g \cdot \psi(x) \cdot g'$, where $g' \in G_x$, which proves our claim. \square

Proposition A.2. The image of any suitable group function $\psi : X \rightarrow G$ is surjective into $\frac{G}{G_x}$, where G_x is the stabilizers of all the points of X .

Proof: Let $x \in X$ be such that $x = \delta(\eta(x))$, that is, $\psi(x) = G_x$, the stabilizer of x . Note that each orbit contains at least one such element. For any element $g \in G$ we have that, using Proposition A.1, $\psi(\rho_X(g)x) = g \cdot \psi(x) \cdot \tilde{g}$, where $\tilde{g} \in G_x$. Since $\psi(x) \cdot \tilde{g} \in G_x$ as well, it then follows that the image of ψ is G up to an action by an element of the stabilizer G_x . Applying the above reasoning to every points $x \in X$, we have that $\text{Im}(\psi) = \cup_{x \in X} \frac{G}{G_x} = \frac{G}{\cap_{x \in X} G_x}$, where $\cap_{x \in X} G_x = G_X = \{g \in G \mid \rho_X(g)x = x, \forall x \in X\}$, proving our claim. \square

Lemma A.3. Any suitable group function ψ is an isomorphism $O_x \simeq G/G_x$ for any $x \in X$, where $O_x \subset X$ is the orbit of x with respect to G in X .

Proof: Surjectivity follows directly from Proposition A.2. To show injectivity, consider $x, x' \in O_x$ such that $\psi(x') = \psi(x) \cdot \tilde{g}$, where $\tilde{g} \in G_x$. From Proposition 2.3 it follows that $x' = x$, which proves the claim. \square

Proposition A.4. Let $\psi = \xi \circ \mu$ be a suitable group function and let $\mu : X \rightarrow Y$ be G -equivariant. Then, $G_x = G_{\mu(x)}$ for all $x \in X$.

Proof: Let $g \in G_x$, that is, $\rho_X(g)x = x$. Applying μ to both sides of this equation we obtain $\mu(x) = \mu(\rho_X(g)x) = \rho_Y(g)\mu(x)$, where we used the G -equivariance of μ . Hence, $G_x \subseteq G_{\mu(x)}$. To prove the opposite inclusion, let $g \in G_{\mu(x)}$ but $g \notin G_x$, and let $x' = \rho_X(g)x$. Now, $\mu(x') = \rho_Y(g)\mu(x) = \mu(x)$, thus μ , and therefore $\psi = \xi \circ \mu$, maps the distinct element x, x' to the same group element $\psi(x) = \psi(x')$, in contradiction with Proposition 2.3. \square

Proposition A.5. Given y, y_0 , the element g such that $y \equiv \rho_Y(g)y_0$ is unique up to the stabilizer G_{y_0} .

Proof: Suppose that there exist $g_1, g_2 \in G$ such that $\rho_Y(g_1)y_0 = \rho_Y(g_2)y_0$, then $\rho_Y(g_2^{-1}g_1)y_0 = y_0$, which implies $g_2^{-1}g_1 \in G_{y_0}$. \square

Proposition A.6. Let $\psi = \xi \circ \mu$ where μ and ξ are as described above. Then, ψ is a suitable group function.

Proof: We show that our construction describes an isomorphism $O_x \simeq G/G_x$ for all $x \in X$. Given $x \in X$ and $g \in G$, Propositions A.4 and A.5 imply

$$\xi(\mu(\rho_X(g)x)) = \xi(\rho_Y(g)\mu(x)) \subseteq g \cdot \xi(\mu(x))G_x , \quad (11)$$

that is, ψ possesses the G -equivariant property as required in Proposition 2.3, which in turns imply injectivity, as in Lemma A.3. Surjectivity follows from the same argument as in Proposition A.2, since the proof only relies on the equivariant properties of ψ , which we showed in (11). \square

B Model architecture Rotated MNIST

We follow Weiler & Cesa (2019b) and use steerable CNNs to parameterize functions η and μ . In contrast to classical CNNs, CNNs with $O(2)$ -steerable kernels transform feature fields respecting the transformation law under actions of $O(2)$. We can define scalar fields $s : \mathbb{R}^2 \rightarrow \mathbb{R}$ and vector fields $v : \mathbb{R}^2 \rightarrow \mathbb{R}^2$ that transform under group actions (rotations) the following:

$$s(x) \mapsto s(g^{-1}x) \quad v(x) \mapsto g \cdot v(g^{-1}x) \quad \forall g \in O(2). \quad (12)$$

Thus, scalar values are moved from one point on the plane \mathbb{R}^2 to another but are not changed, while vectors are moved and changed (rotated) equivalently. Hence, we can utilize steerable CNNs to encode samples in \mathbb{R}^2 in $O(2)$ -invariant scalar features and $O(2)$ -equivariant vector features. We can use the scalar features s as \mathcal{G} -invariant representation $z \in Z$ and following Section 3 (Orthogonal group $SO(2)$) utilizing a single vector features v to construct the rotation matrix R as:

$$R = \begin{bmatrix} \bar{v}_x & -\bar{v}_y \\ \bar{v}_y & \bar{v}_x \end{bmatrix}, \quad \bar{v} = \frac{v}{\|v\|}. \quad (13)$$

In our experiments we used seven layers of steerable CNNs as implemented by Weiler & Cesa (2019b). We did not use pooling layers, as we found them to break rotation equivariance and only averaged over the two spatial dimensions after the final layer to extract the final invariant embedding and equivariant vector. In each layer we used 32 hidden scalar and 32 hidden vector fields. In the final layer we used 32 scalar fields (32 dimensional invariant embedding) and one vector feature field.

The Decoding function $\delta : Z \rightarrow \mathbb{R}^2$ can be parameterized by a regular CNN. In our experiments we used six layers of regular CNNs with 32 hidden channels, interleaved with bilinear upsampling layers starting from the embedding expanded to a $2 \times 2 \times 32$ tensor.

Training was done on one NVIDIA Tesla V100 GPU in approximately 6 hours.

We also implemented and trained the quotient autoencoder (QAE) approach proposed by Mehr et al. (2018a) on the MNIST dataset for the group $SO(2)$, discretized in 36 rotations with the loss

$$\min_{\theta \in \{10i, i=0, \dots, 35\}} \{\text{MSE}(x - \rho_X(g(\theta))y)\}, \quad (14)$$

where x is a MNIST sample and y is the reconstructed sample. We evaluated the resulting embeddings on the rotated MNIST test set (in such a way that the evaluation is the same as for our model). In Figure 6 we plot TSNE embeddings for this approach, and we can observe that the embedding space shows a clearer structure, in comparison with the classical model. However, in comparison, our approach results in a better clustering of the different digits classes. That shows that the discretization step, while it helps in structuring the embedding space in “signal clusters”, still does not capture the full continuous nature of the group. To further quantitatively compare the three methods (ours, QAE (a) and classical AE), we evaluated the reconstruction loss as well as the (digit class) classification accuracy of a KNN classifier trained on 1000 embeddings of each method. We present in the table below the results for the reconstruction loss and for the classification accuracy of a KNN classifier trained on the AE embeddings. To obtain a fair comparison, we kept the architecture and the training hyperparameters exactly identical for all the strategies. We note that our strategy outperforms both the classical AE as well as the strategy of (a) in both tasks.

We also performed an additional experiment, we trained a fully equivariant AE (that is, the embedding itself is fully equivariant, i.e. multiple 2-dimensional vectors) on MNIST with $G = SO(2)$, and we perform an invariant pooling afterwards to extract the invariant part. Specifically, we have trained KNN classifiers on (a) the invariant embedding corresponding to the norm of the 2-dimensional vectors forming the bottleneck representation, (b) the angles between the first and all other vectors and on (c) the full invariant embedding we obtained by combining the the norms and angles. We choose the number of vectors in the bottleneck in such a way that the dimensionality of the full invariant representation coincides with the one of our model. We visualized the resulting TSNE embeddings in Figure 6 and show the downstream performance of the KNN classifiers in Table 1. From the results we can see that, in comparison to the approximate invariant (QAE) and our invariant trained model, the invariant projected equivariant representations perform inferior. Although we extract a complete invariant representation (which performs better than a subset of this representation like the norm or angle part), the resulting representation is apparently not as expressive and e.g. useful in a downstream classification task. This aligns well with our hypothesis, that our proposed framework poses a sensible supervisory signal to extract expressive invariant representations that are superior to invariant projections of equivariant features.

Table 1: Comparison of our approach vs classical and quotient autoencoder (QAE) as well as an fully equivariant AE with invariant pooling after training.

Model	Rec. Loss	KNN Acc.
classical	0.0170	0.68
QAE	0.0227	0.82
invariant (ours)	0.0162	0.90
equiv AE (norm)	0.0189	0.56
equiv AE (angle)	0.0189	0.53
equiv AE (complete)	0.0189	0.67

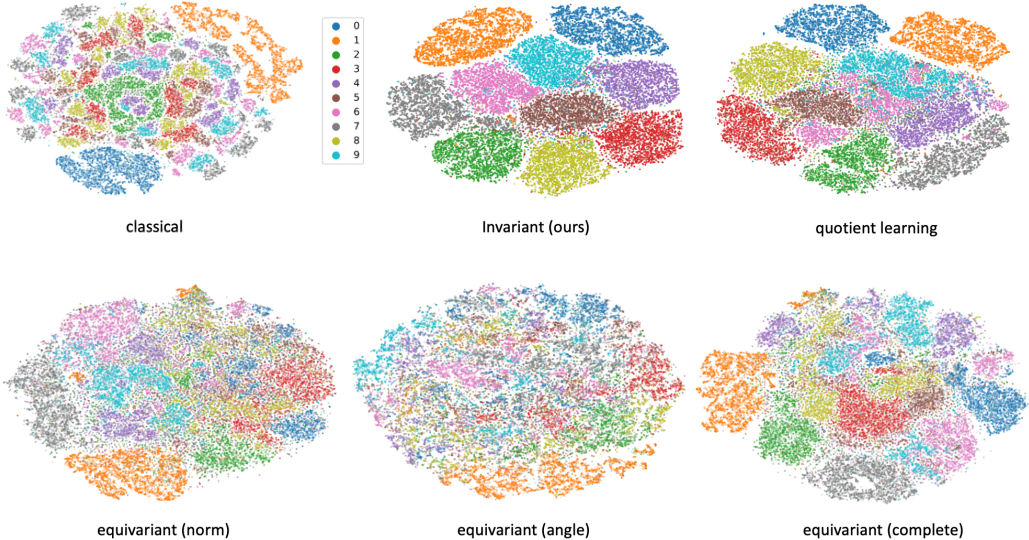


Figure 6: Top row: TSNE embedding of the encoded test dataset for a classical autoencoder, our proposed $SO(2)$ invariant autoencoder, and for the quotient autoencoder of Mehr et al. (2018a). Bottom row: Fully equivariant trained autoencoder with invariant projection after training, either by taking the norm, angles between vectors or the combination (complete).

C Model architecture Set of Digits

We can rewrite the equation $P_\sigma(1, 2, \dots, n) = (\sigma(1), \sigma(2), \dots, \sigma(n))$, in vector form by representing set elements by standard $n \times 1$ column vectors \mathbf{e}_i (one-hot encoding) and σ by a permutation matrix P_σ whose (i, j) entry is 1 if $i = \sigma(j)$ and 0 otherwise, then:

$$P_\sigma \mathbf{e}_i = \mathbf{e}_{\sigma(i)} \tag{15}$$

Hence, encoding function η should encode a set of elements in a permutation invariant way and ψ should map a set M to a permutation matrix P_σ :

$$\psi : M \rightarrow P_\sigma \tag{16}$$

We follow Zaheer et al. (2017) and parameterize η by a neural network γ that is applied element-wise on the set followed by an invariant aggregation function Σ (e.g. sum or average) and a second neural network β :

$$\eta(X) = \beta(\Sigma_{x \in X} \gamma(x)) . \tag{17}$$

In our experiments we parameterized γ and β with regular feed-forward neural networks with three layers respectively, also using ReLU activations and Batchnorm.

The output of function γ is equivariant and can also be used to construct ψ . We follow Winter et al. (2021) and define a function $s : \mathbb{R}^d \rightarrow \mathbb{R}$ mapping the output of γ for every set element to a scalar

value. By sorting the resulting scalars, we construct the permutation matrix P_σ with entries p_{ij} that would sort the set of elements with respect to the output of s :

$$p_{ij} = \begin{cases} 1, & \text{if } j = \text{argsort}(s)_i \\ 0, & \text{else} \end{cases} \quad (18)$$

As the argsort operation is not differentiable, we utilize a continuous relaxation of the argsort operator proposed in (Prillo & Eisenschlos, 2020; Grover et al., 2019):

$$\mathbf{P} \approx \hat{\mathbf{P}} = \text{softmax}\left(\frac{-d(\text{sort}(s)\mathbf{1}^\top, \mathbf{1}s^\top)}{\tau}\right), \quad (19)$$

where the softmax operator is applied row-wise, $d(x, y)$ is the L_1 -norm and $\tau \in \mathbb{R}_+$ a temperature-parameter.

Decoding function δ can be parameterized by a neural network that maps the permutation-invariant set representation back to either the whole set or single set elements. In the latter case, where the same function is used to map the same set representation to the different elements, additional fixed position embeddings can be fed into the function to decode individual elements for each position/index. For the reported results we choose this approach, using one-hot vectors as position embeddings and a 4-layer feed-forward neural network.

Training was done on one NVIDIA Tesla V100 GPU in approximately 1 hours.

D Model architecture Point Cloud - Tetris 3D

We implement a graph neural network (GNN) that transform equivariantly under rotations and translations in 3D space, respecting the invariance and equivariance constraints mentioned in Eq. (5) and (6) for $n = 3$.

Assume we have a point cloud of N particles each located at a certain position $x_i \in \mathbb{R}^3$ in Cartesian space. Now given some arbitrary ordering $\sigma(\cdot)$ for the points, we can store the positional coordinates in the matrix $P = [x_1, \dots, x_N] \in \mathbb{R}^{N \times 3}$. Standard Graph Neural Networks (GNNs) perform message passing Gilmer et al. (2017) on a local neighbourhood for each node. Since we deal with a point cloud, common choice is to construct neighbourhoods through a distance cutoff $c > 0$. The edges of our graph are specified by *relative* positions

$$x_{ij} = x_j - x_i \in \mathbb{R}^3,$$

and the neighbourhood of node i is defined as $\mathcal{N}(i) = \{j : d_{ij} := \|x_{ij}\| \leq c\}$.

Now, our data (i.e., the point cloud) lives on a vector space X , where we want to learn an SE(3) invariant and equivariant embedding wrt. arbitrary rotations and translations in 3D space. Let the feature for node i consist of an invariant (type-0) embedding $h_i \in \mathbb{R}^{F_s}$, an equivariant (type-1) embedding $w_i \in \mathbb{R}^{3 \times F_v}$ that transforms equivariantly wrt. arbitrary rotation **but** is invariant to translation. Such a property can be easily obtained, when operating with relative positions.

Optionally, we can model another equivariant (type-1) embedding $t_i \in \mathbb{R}^3$ which transforms equivariantly wrt. translation **and** rotation. As our model needs to learn to predict group actions in the SE(3) symmetry, we require to predict an equivariant translation vector ($b \in T_3$), as well as a rotation matrix ($A \in \text{SO}(3)$), where we will dedicate the t vector to the translation and the w vector(s) to the rotation matrix.

As point clouds might not have initial features, we initialize the SE(3)-invariant embeddings as one-hot encoding $h_i = e_i$ for each node $i = 1, \dots, N$. The (vector) embedding dedicated for predicting the rotation matrix is initialized as zero-tensor for each particle, i.e., $w_i = \mathbf{0}$ and the translation vector is initialized as the absolute positional coordinate, i.e. to, $t_i = x_i$.

We implement following edge function $\phi_e : \mathbb{R}^{2F_s+1} \mapsto \mathbb{R}^{F_s+2F_v+k}$ with

$$m_{ij} = \phi_e(h_i, h_j, d_{ij}) = W_e[h_i, h_j, d_{ij}] + b_e, \quad (20)$$

and set $k = 1$ if the GNN should model the translation and $k = 0$ else. Notice that the message m_{ij} in Eq. (20) only depends on SE(3) invariant embeddings. Now, (assuming $k = 1$) we further split the message tensor into 4 tensors,

$$m_{ij} = [m_{h,ij}, m_{w_0,ij}, m_{w_1,ij}, m_{t,ij}],$$

which we require to compute the aggregated messages for the SE(3) invariant and equivariant node embeddings.

We include a row-wise transform $\phi_s : \mathbb{R}^{F_s} \mapsto \mathbb{R}^{F_s}$ for the invariant embeddings using a linear layer:

$$\tilde{h}_i = W_s h_i + b_s, \quad (21)$$

The aggregated messages for invariant (type-0) embedding h_i are calculated using:

$$m_{i,h} = \sum_{j \in \mathcal{N}(i)} m_{h,ij} \odot \tilde{h}_i \in \mathbb{R}^{F_s}. \quad (22)$$

where \odot is the (componentwise) scalar-product.

The aggregated equivariant features are computed using the tensor-product \otimes and scalar-product \odot from (invariant) type-0 representations with (equivariant) type-1 representations:

$$m_{i,w} = \sum_{j \in \mathcal{N}(i)} (x_{ij} \otimes m_{w_0,ij} + (w_i \times w_j) \odot (\mathbf{1} \otimes m_{w_1,ij})) \in \mathbb{R}^{3 \times F_v}, \quad (23)$$

where $\mathbf{1} \in \mathbb{R}^3$ is the vector with 1's as components and $(a \times b)$ denotes the cross product between two vectors $a, b \in \mathbb{R}^3$.

The tensor in Eq. (23) is equivariant to arbitrary rotations and invariant to translations. It is easy to prove the translation invariance, as any translation $t^* \in T_3$ acting on points x_i, x_j does not change the relative position $x_{ij} = (x_j + t^*) - (x_i + t^*) = x_j - x_i$.

To prove the rotation equivariance, we first observe that given any rotation matrix $A \in \text{SO}(3)$ acting on the provided data, as a consequence relative positions rotate accordingly, since

$$Ax_j - Ax_i = A(x_j - x_i) = Ax_{ij} \in \mathbb{R}^3.$$

The tensor product \otimes between two vectors $u \in \mathbb{R}^3$ and $v \in \mathbb{R}^{F_s}$, commonly also referred to as *outer product* is defined as

$$u \otimes v = uv^\top \in \mathbb{R}^{3 \times F_s},$$

and returns a matrix given two vectors. For the case that a group representation of $\text{SO}(3)$, i.e. a rotation matrix R , acts on u , it is obvious to see with the associativity property

$$(Au) \otimes v = (Au)v^\top = Auv^\top = A(uv^\top) = A(u \otimes v) = Au \otimes v.$$

The cross product $(w_i \times w_j) \in \mathbb{R}^{3 \times F_v}$ used in equation (23) between type-1 features w_i and w_j is applied separately on the last axis. The cross product has the algebraic property of rotation invariance, i.e. given a rotation matrix A acting on two 3-dimensional vectors $a, b \in \mathbb{R}^3$ the following holds:

$$(Aa) \times (Ab) = A(a \times b). \quad (24)$$

Now, notice that the quantities that "transform as a vector" which we call type-1 embeddings are in $S = \{x_{ij}, w_i, t_i\}_{i,j=1}^N$.

Given a rotation matrix A acting on elements of S , we can see that the result in (23)

$$\begin{aligned} & \sum_{j \in \mathcal{N}(i)} (Ax_{ij} \otimes m_{w_0,ij} + (Aw_i) \times (Aw_j) \odot (\mathbf{1} \otimes m_{w_1,ij})) \\ &= \sum_{j \in \mathcal{N}(i)} (Ax_{ij} \otimes m_{w_0,ij} + A(w_i \times w_j) \odot (\mathbf{1} \otimes m_{w_1,ij})) \\ &= A \sum_{j \in \mathcal{N}(i)} (x_{ij} \otimes m_{w_0,ij} + (w_i \times w_j) \odot (\mathbf{1} \otimes m_{w_1,ij})) \\ &= Am_{i,w} \end{aligned}$$

is rotationally equivariant.

We update the hidden embedding with a residual connection

$$\begin{aligned} h_i &\leftarrow h_i + m_{i,h}, \\ w_i &\leftarrow w_i + m_{i,w}, \end{aligned} \quad (25)$$

and use a Gated-Equivariant layer with equivariant non-linearities as proposed in the PaiNN architecture Schütt et al. (2021) to enable an information flow between type-0 and type-1 embeddings. The type-1 embedding for the translation vector is updated in a residual fashion

$$\begin{aligned} t_i &\leftarrow t_i + \sum_{j \in \mathcal{N}(i)} x_{ij} \otimes m_{t,ij} \\ &= t_i + \sum_{j \in \mathcal{N}(i)} m_{t,ij} \odot x_{ij}, \end{aligned} \quad (26)$$

where we can replace the tensor-product with a scalar-product, as $m_{t,ij} \in \mathbb{R}$. The result in Eq. (26) is translation and rotation equivariant as the first summand t_i is rotation and translation equivariant, while the second summand is only rotation equivariant since we utilize relative positions.

For the SE(3) Tetris experiment, the encoding function $\eta : X \mapsto Z$ is a 5-layer GNN encoder with $F = F_s = F_v = 32$ scalar- and vector channels and implements the translation vector, i.e. $k = 1$. The encoding network η outputs four quantities: two SE(3) invariant node embedding matrices $\tilde{H}, M \in \mathbb{R}^{N \times F}$, one SO(3) equivariant order-3 tensor $\tilde{W} \in \mathbb{R}^{N \times 3 \times F}$ as well as another SE(3) equivariant matrix $T \in \mathbb{R}^{N \times 3}$.

We use two linear layers⁴ to obtain the SE(3) invariant embedding matrix $H \in \mathbb{R}^{N \times 2}$ as well as the SO(3) equivariant embedding tensor $W \in \mathbb{R}^{N \times 3 \times 2}$. Notice that the linear layer returning the W tensor can be regarded as the function ψ_{rot} that aims to predict the group action in the SO(3) symmetry, while we use the identity map for the translation vector, i.e. $\psi_{\text{transl}} = T$.

As point clouds can be regarded as sets, we obtain an permutation invariant embedding by averaging over the first dimension of the $\{H, W, T\}$ tensors,

$$h = \frac{1}{N} \sum_{i=1}^N H_i \in \mathbb{R}^2, \quad (27)$$

$$t = \frac{1}{N} \sum_{i=1}^N T_i \in \mathbb{R}^3, \quad (28)$$

$$w = \frac{1}{N} \sum_{i=1}^N W_i \in \mathbb{R}^{3 \times 2}, \quad (29)$$

while we use the M matrix to predict the permutation matrix P_σ with the ψ_{perm} function, in similar fashion as described in Eq. (16). To construct the rotation matrix R out of 2 vectors in \mathbb{R}^3 as described in Section 3, we utilize the SO(3) equivariant embedding w .

The decoding network $\delta : Z \mapsto X$ is similar to the encoder a 5-layer SE(3)-equivariant GNN but does not model the translation vector, i.e. $k = 0$. The decoder δ maps the SE(3) as well as S(N)-invariant embedding h back to a reconstructed point cloud $\hat{P} \in \mathbb{R}^{N \times 3}$. At the start of decoding, we utilize a linear layer to map the G -invariant embedding $h \in \mathbb{R}^2$ to a higher-dimension, i.e.

$$\tilde{h} = W_0 h + b_0 \in \mathbb{R}^{F_s}, \quad (30)$$

Next, to “break” the symmetry and provide the nodes with initial type-0 features, we utilize fixed (deterministic) positional encodings as suggested by Winter et al. (2021) for each node $i = 1, \dots, N$ to be summed with \tilde{h} . Notice that this addition enables us to obtain distinct initial type-0 embeddings $\{\hat{h}_i\}_{i=1}^N$.

For the start positions, we implement a trainable parameter matrix P_θ of shape $(N \times 3)$ for the decoder.

Now, given an initial node embedding $\hat{H} \in \mathbb{R}^{N \times F_s}$, we apply the S(N) group action, by multiplying the predicted permutation matrix P_σ with \hat{H} from the left to obtain the canonical ordering as

$$\hat{H}_\sigma = P_\sigma \hat{H}. \quad (31)$$

To retrieve the correct orientation required for the pairwise-reconstruction loss, we multiply the constructed rotation matrix R with the initial start position matrix P_θ

$$\hat{P}_r = P_\theta R^\top. \quad (32)$$

⁴The transformation is always applied on the last (feature) axis.

Target	Fraction	Pretrained	From Scratch
H	0.05	0.7529	0.0970
	0.25	0.9908	0.9093
G	0.05	0.7703	0.4758
	0.25	0.9856	0.9751
U	0.05	0.6083	0.2574
	0.25	0.9962	0.9808
$\langle R^2 \rangle$	0.05	0.7806	0.1468
	0.25	0.9918	0.8546
μ	0.05	0.8698	0.8443
	0.25	0.9718	0.9718
α	0.05	0.9455	0.9237
	0.25	0.9937	0.9764

Table 2: Generalization performance in terms of the coefficient of determination R^2 of models on a held-out test set of 1000 samples. Higher R^2 indicates better performance.

With such construction, we can feed the two tensors to the decoder network δ to obtain the reconstructed point cloud as

$$\hat{P}_{\text{recon}} = \delta(\hat{H}_\sigma, \hat{P}_r) + t, \quad (33)$$

where $t \in \mathbb{R}^3$ is the predicted translation vector from the encoder network, added row-wise for each node position.

Architecture QM9. For the QM9 dataset, we use the same model components as described in the Tetris experiment, with the difference of including atom species as $SE(3)$ -invariant features and setting $F_s = 256$, $F_v = 32$ and increasing the dimensionality of the latent space to 256.

Finetuning. We performed additional experiments on the pretrained group-invariant AE on the extended GEOM-QM9 dataset Axelrod & Gomez-Bombarelli (2021) which, as opposed to the standard QM9 dataset ($\approx 130k$ samples), contains multiple conformations of small molecules. We trained the autoencoder on a reduced set of GEOM-QM9 ($\approx 641k$), containing up to 10 conformations per molecule and utilized this pretrained encoder network to regress (invariant) energy targets, such as internal energy U or enthalpy H on the original QM9 dataset.

We observed that the pretrained encoder network learns faster and achieves better generalization performance than the architectural identical network trained from scratch. In Figure 7 we illustrates the learning curves for the two networks on different fraction on 5% and 25% labelled samples from original QM9 dataset to analyze the benefit of finetuning a pre-trained encoder network on a low-data regime, when regressing on the enthalpy H . On a held-out test dataset of 1000 samples, the pretrained encoder network achieves superior generalization performance in terms of R^2 with 0.7529 vs. 0.0970 in the 5% data regime, and 0.9908 vs. 0.9093 in the 25% data regime compared to the encoder that was trained from scratch. In Table 2 we show additional comparisons of the pretrained network against a network that was trained from scratch for 50 epochs on the restricted dataset. As shown in Table 2, the pretrained encoder achieves improved generalization performance on the test dataset compared to its architectural identical model that was trained from scratch. We believe that training the group-invariant autoencoder on a larger diverse dataset of (high-quality) molecular conformations facilitates new opportunities in robust finetuning on different data-scarce datasets for molecular property prediction.

E Molecular Conformations: Further Examples

We show additional reconstructions of 12 randomly selected small molecules from the QM9 test dataset. Noticeably, our trained autoencoder is able to reconstruct molecular conformations with complex geometries as depicted in the third column (from the left). We notice that the AE is not able to perfectly reconstruct the conformation shown in the 4th column of the 2nd row. Although this molecule does not exhibit a complicated geometrical structure, its atomistic composition (of only containing nitrogen and carbon as heavy atoms) could be the reason why the encoding of the

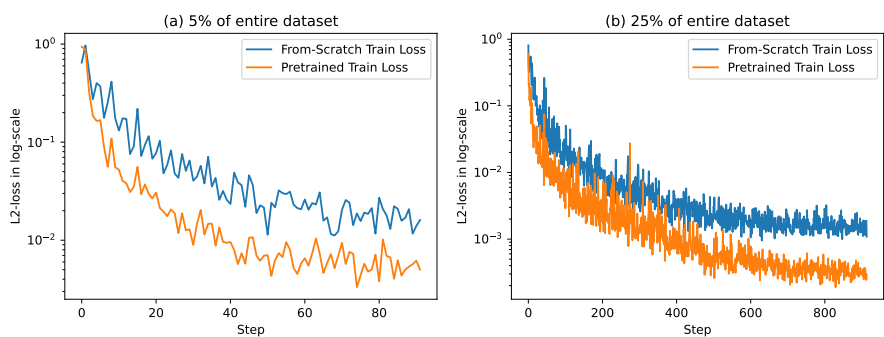


Figure 7: Learning curves of models trained on limited enthalpy H targets.

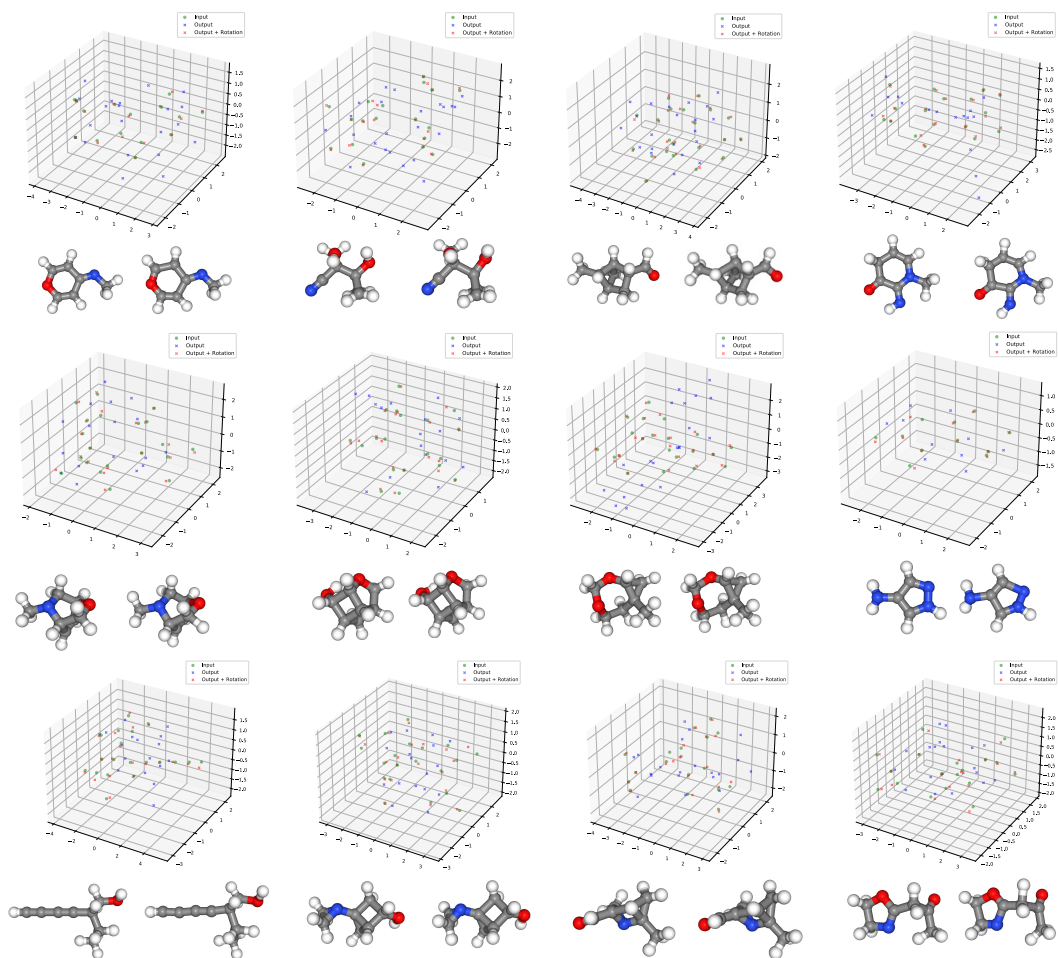


Figure 8: 12 molecular conformations and their reconstructions represented as point cloud and ball-and-stick model (left true, right predicted).

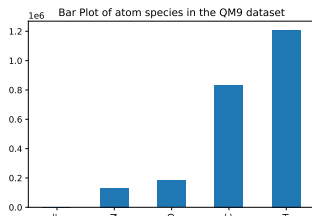


Figure 9: Atomistic species count on the QM9 dataset.



Figure 10: TSNE embedding of the encoded dataset for a classical autoencoder and our proposed SO(3) invariant autoencoder.

conformation is pointing into a non-densely populated region in the latent space, as nitrogen does not have a large count in the total QM9 database, see Figure 9.

Training was done on one NVIDIA Tesla V100 GPU in approximately 1 day.

F ShapeNet

We also run experiments on the ShapeNet dataset. As the dataset comes in an aligned form (e.g. cars are always aligned in the same orientation), we additionally applied random 90 degree rotations to remove this bias. We utilized 3D Steerable CNNs proposed by Weiler et al. (2018b) as equivariant encoder for the 3d voxel input space. We utilized the scalar outputs as rotation-invariant embedding (z) and predict (analogously to our experiments on 3d point clouds) 2 rotation-equivariant vectors to construct a rotation matrix (g). Similar to our MNIST experiment, we compared the resulting embedding space to the embeddings produced by a non-invariant autoencoder model. In Figure 10 we visualize a TSNE projection of the embeddings of both models. We can see a well structured embedding space for our model with distinct clusters for the different shape classes. On the other hand, the embeddings produced by the non-invariant autoencoder is less structured and one can make out different clusters for the same shape label but in different orientations. Moreover, we compared the downstream performance and generalizability of a KNN classifier on shape classification, trained on 1000 embeddings and tested on the rest. The classifier based on our rotation-invariant embeddings achieved an accuracy of 0.81 while the classifier based on the non-invariant embeddings achieved an accuracy of only 0.63.

G Additional Related Work Section

The field of unsupervised invariant representation learning can be roughly divided into two categories. The first consists in learning an approximate group action in order to match the input and the reconstructed data. For instance, Mehr et al. (2018b) propose (like us) to encode the input in quotient space, and train the model with a loss that is defined by taking the infimum over the group G . While this is feasible for (small) finite groups, for continuous groups they either have to approximately discretize them or perform a separate optimization of h at every back propagation step to find the best match. Other work Shu et al. (2018); Koneripalli et al. (2020) proposes to disentangle the

embedding in a shape-like and a deformation-like component. While this is in spirit with our work, their transformations are local (we focus on global transformations) and are approximative, that is, the components are not explicitly invariant and equivariant with respect to the transformation, respectively.

In the case of 2D/3D data, co-alignment of shapes can be used to match the input and the reconstructed shapes. Some approaches are unfeasible Wang et al. (2012) as they are not compatible with a purely unsupervised approach, while other Averkiou et al. (2016); Chaouch & Verroust-Blondet (2008, 2009) leverage symmetry properties of the data and PCA decomposition, exhibiting however limitation regarding scalability. For graphs, the problem of graph matching Bunke & Jiang (2000) has been tackled in several works and with different approaches, for instance algorithmically, e.g., Ding et al. (2020), or by means of a GNN Li et al. (2019).

On the topic of representation theory-based embedding disentanglement, the works Hosoya (2019); Keurti et al. (2022), based on the definition of Higgins et al. (2018) of a disentangled representations, design unsupervised generative VAEs approaches for learning representation corresponding to orthogonal symmetry actions on the data space. In our work, on the other hand, we disentangled different representations, which all can act on our data space.

Chapter 3

Conclusion

This thesis mainly concerned itself with the development of novel methods to describe and represent molecules in a meaningful way and utilize such representations for molecular generation and optimization in pharmaceutical drug design. An important aspect of representation learning for molecules is the necessity of unsupervised learning approaches to extract such representations, as labeled data is usually scarce. In the published works, we propose different methods, handling the different challenging data types molecules are usually represented by, such as line notations (e.g. SMILES), graphs and point clouds, defining different unsupervised learning objectives to extract expressive descriptors.

In the first publication, we proposed a Deep Learning method that learns from a large unlabeled dataset to extract meaningful molecular representations from the SMILES line notation. We demonstrated its competitive if not superior performance in various molecular property prediction and virtual screening benchmarks.

In the second work, we utilized this novel representation for molecular de-novo design and lead optimization. We showed how the proposed method can be used to efficiently optimize a molecule with respect to a multi-objective value function with potential structural constraints.

In the third publication, we combined the first two works to design an interactive web application to steer a molecular optimization procedure from start to end.

In the fourth work, we moved past representing just the molecular topology and proposed a method that can learn expressive representations from the molecular conformation. We demonstrated how the internal coordinates of different molecules can be compressed in the same fixed-size latent space and how such representation can be used to efficiently sample energetically reasonable conformations for a given molecular topology.

In the fifth publication, we proposed a method to represent any graph (including

molecular graphs) in a fixed-size permutation invariant way. We discussed the benefits and necessity of such a permutation invariant graph-level representation for a variety of task and demonstrated its superiority over other methods lacking this property.

In the final publication, we generalized the concept of learning permutation-invariant graph representations with an autoencoder framework to any other data type and group. We derived the necessary conditions for the different parts of the model independently of a specific data type, group or network architecture. We demonstrated how our proposed method is able to represent images in a rotation-invariant way, sets of digits in permutation-invariant way and point clouds in a rotation, translation and permutation-invariant way. The later approach can be used in future work to autoencode molecular conformations represented in Cartesian coordinates.

In essence, this thesis investigated how unsupervised learning can be utilized to extract powerful representations of molecular compounds that can be used to improve or enable downstream applications relevant for pharmaceutical drug development such as efficient molecular property prediction or molecular de novo design.

We discussed how naive applications of unsupervised learning method might lead to inferior representations if crucial symmetries of the data space are not accounted for. We proposed novel methods that account for such symmetries, either indirectly, by employing a self-supervised learning objective (translation between different line notations of molecules) or directly, by modeling the symmetry groups of interest (permutation group or Euclidean group).

While this thesis is mainly concerned with the representation of drug candidate molecules, another interesting angle to investigate in future work is the use of unsupervised learning methods to represent the target of a potential drug. The methods developed, especially in publications five and six, could for example be used to represent a protein pocket or cavity in a rotation-invariant and fixed-sized descriptor. Such a descriptor could be combined with a ligand descriptor to build powerful *proteochemometric* models [Van Westen et al., 2011], leveraging not only the structural similarity of ligands but also of their targets.

The present work utilized many different *raw* representations of molecules (such as line-notations, internal coordinates and Cartesian coordinates) as input for the proposed unsupervised learning algorithms. Another interesting input not discussed in this work could be the electrostatic potential or electron density of a molecule. As the electrostatic interaction of a ligand with its target is the main driver for binding, it could be beneficial to extract a molecular descriptor directly from the electron density. In future work we would like to utilize the method developed in publication six to develop such a model.

As the development of drugs gets more challenging [Wouters et al., 2020], the importance and necessity of computational methods to support this process gets increasingly substantial. However, in order to apply any computational method to the field of chemistry, the first step usually involves representing chemical structures in a computer-interpretable way. If this representation fails to reflect important properties of molecules, subsequent methods will inevitably fail to model those properties. Hence, we argue that finding the most suitable and meaningful molecular representations is a crucial part in enabling computational chemistry and ultimately the discovery of novel cures for diseases. We hope that the work presented in this thesis helped in pushing the field forward to the development of such more desirable molecular representations.

Bibliography

- [Bai et al., 2019] Bai, Y., Ding, H., Qiao, Y., Marinovic, A., Gu, K., Chen, T., Sun, Y., and Wang, W. (2019). Unsupervised inductive graph-level representation learning via graph-graph proximity. *arXiv preprint arXiv:1904.01098*.
- [Bajorath, 2004] Bajorath, J. (2004). Chemoinformatics. *Concepts, Methods, and Tools for Drug Discovery*.
- [Bank et al., 2020] Bank, D., Koenigstein, N., and Giryes, R. (2020). Autoencoders. *arXiv preprint arXiv:2003.05991*.
- [Bender and Brown, 2018] Bender, A. and Brown, N. (2018). Cheminformatics in drug discovery.
- [Bender and Glen, 2004] Bender, A. and Glen, R. C. (2004). Molecular similarity: a key technique in molecular informatics. *Organic & biomolecular chemistry*, 2(22):3204–3218.
- [Bengio et al., 2012] Bengio, Y., Courville, A. C., and Vincent, P. (2012). Unsupervised feature learning and deep learning: A review and new perspectives. *CoRR*, abs/1206.5538, 1:2012.
- [Bjerrum, 2017] Bjerrum, E. J. (2017). Smiles enumeration as data augmentation for neural network modeling of molecules. *arXiv preprint arXiv:1703.07076*.
- [Bronstein et al., 2021] Bronstein, M. M., Bruna, J., Cohen, T., and Veličković, P. (2021). Geometric deep learning: Grids, groups, graphs, geodesics, and gauges. *arXiv preprint arXiv:2104.13478*.
- [Brown et al., 2019] Brown, N., Fiscato, M., Segler, M. H., and Vaucher, A. C. (2019). Guacamol: benchmarking models for de novo molecular design. *Journal of chemical information and modeling*, 59(3):1096–1108.
- [Bruna et al., 2013] Bruna, J., Zaremba, W., Szlam, A., and LeCun, Y. (2013). Spectral networks and locally connected networks on graphs. *arXiv preprint arXiv:1312.6203*.

- [Cao et al., 2016] Cao, S., Lu, W., and Xu, Q. (2016). Deep neural networks for learning graph representations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 30.
- [Cereto-Massagué et al., 2015] Cereto-Massagué, A., Ojeda, M. J., Valls, C., Mulero, M., Garcia-Vallvé, S., and Pujadas, G. (2015). Molecular fingerprint similarity search in virtual screening. *Methods*, 71:58–63.
- [Cherkasov et al., 2014] Cherkasov, A., Muratov, E. N., Fourches, D., Varnek, A., Baskin, I. I., Cronin, M., Dearden, J., Gramatica, P., Martin, Y. C., Todeschini, R., et al. (2014). Qsar modeling: where have you been? where are you going to? *Journal of medicinal chemistry*, 57(12):4977–5010.
- [Chithrananda et al., 2020] Chithrananda, S., Grand, G., and Ramsundar, B. (2020). Chemberta: Large-scale self-supervised pretraining for molecular property prediction. *arXiv preprint arXiv:2010.09885*.
- [Clevert et al., 2021] Clevert, D.-A., Le, T., Winter, R., and Montanari, F. (2021). Img2mol—accurate smiles recognition from molecular graphical depictions. *Chemical science*, 12(42):14174–14181.
- [Cohen and Welling, 2016] Cohen, T. and Welling, M. (2016). Group equivariant convolutional networks. In *International conference on machine learning*, pages 2990–2999. PMLR.
- [Corso et al., 2020] Corso, G., Cavalleri, L., Beaini, D., Liò, P., and Veličković, P. (2020). Principal neighbourhood aggregation for graph nets. *Advances in Neural Information Processing Systems*, 33:13260–13271.
- [Cramer et al., 2008] Cramer, R. D., Cruz, P., Stahl, G., Curtiss, W. C., Campbell, B., Masek, B. B., and Soltanshahi, F. (2008). Virtual screening for r-groups, including predicted pic50 contributions, within large structural databases, using topomer comfa. *Journal of chemical information and modeling*, 48(11):2180–2195.
- [Danziger and Dean, 1989] Danziger, D. and Dean, P. (1989). Automated site-directed drug design: a general algorithm for knowledge acquisition about hydrogen-bonding regions at protein surfaces. *Proceedings of the Royal Society of London. B. Biological Sciences*, 236(1283):101–113.
- [Defferrard et al., 2016] Defferrard, M., Bresson, X., and Vandergheynst, P. (2016). Convolutional neural networks on graphs with fast localized spectral filtering. *Advances in neural information processing systems*, 29.
- [Doweyko, 2004] Doweyko, A. M. (2004). 3d-qsar illusions. *Journal of computer-aided molecular design*, 18(7):587–596.
- [Duvenaud et al., 2015] Duvenaud, D. K., Maclaurin, D., Iparraguirre, J., Bombarell, R., Hirzel, T., Aspuru-Guzik, A., and Adams, R. P. (2015). Convolutional networks on graphs for learning molecular fingerprints. *Advances in neural information processing systems*, 28.

- [Ertl, 2014] Ertl, P. (2014). Cheminformatics and its role in the modern drug discovery process. *University of Strasbourg. Novartis, nd Web*, 18.
- [Fan et al., 2019] Fan, W., Ma, Y., Li, Q., He, Y., Zhao, E., Tang, J., and Yin, D. (2019). Graph neural networks for social recommendation. In *The world wide web conference*, pages 417–426.
- [Gilmer et al., 2017] Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., and Dahl, G. E. (2017). Neural message passing for quantum chemistry. In *International conference on machine learning*, pages 1263–1272. PMLR.
- [Glen et al., 2006] Glen, R. C., Bender, A., Arnby, C. H., Carlsson, L., Boyer, S., and Smith, J. (2006). Circular fingerprints: flexible molecular descriptors with applications from physical chemistry to adme. *IDrugs*, 9(3):199.
- [Göller et al., 2020] Göller, A. H., Kuhnke, L., Montanari, F., Bonin, A., Schneckener, S., Ter Laak, A., Wichard, J., Lobell, M., and Hillisch, A. (2020). Bayer’s in silico admet platform: A journey of machine learning over the past two decades. *Drug Discovery Today*, 25(9):1702–1709.
- [Gómez-Bombarelli et al., 2018] Gómez-Bombarelli, R., Wei, J. N., Duvenaud, D., Hernández-Lobato, J. M., Sánchez-Lengeling, B., Sheberla, D., Aguilera-Iparraguirre, J., Hirzel, T. D., Adams, R. P., and Aspuru-Guzik, A. (2018). Automatic chemical design using a data-driven continuous representation of molecules. *ACS central science*, 4(2):268–276.
- [Goodfellow et al., 2014] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial nets. *Advances in neural information processing systems*, 27.
- [Guimaraes et al., 2017] Guimaraes, G. L., Sanchez-Lengeling, B., Outeiral, C., Farias, P. L. C., and Aspuru-Guzik, A. (2017). Objective-reinforced generative adversarial networks (organ) for sequence generation models. *arXiv preprint arXiv:1705.10843*.
- [Hartenfeller and Schneider, 2010] Hartenfeller, M. and Schneider, G. (2010). De novo drug design. *Chemoinformatics and computational chemical biology*, pages 299–323.
- [Henaff et al., 2015] Henaff, M., Bruna, J., and LeCun, Y. (2015). Deep convolutional networks on graph-structured data. *arXiv preprint arXiv:1506.05163*.
- [Hjelm et al., 2018] Hjelm, R. D., Fedorov, A., Lavoie-Marchildon, S., Grewal, K., Bachman, P., Trischler, A., and Bengio, Y. (2018). Learning deep representations by mutual information estimation and maximization. *arXiv preprint arXiv:1808.06670*.
- [Hochreiter and Schmidhuber, 1997] Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.

- [Hoffman et al., 2022] Hoffman, S. C., Chenthamarakshan, V., Wadhawan, K., Chen, P.-Y., and Das, P. (2022). Optimizing molecules using efficient queries from property evaluations. *Nature Machine Intelligence*, 4(1):21–31.
- [Hu et al., 2012] Hu, G., Kuang, G., Xiao, W., Li, W., Liu, G., and Tang, Y. (2012). Performance evaluation of 2d fingerprint and 3d shape similarity methods in virtual screening. *Journal of chemical information and modeling*, 52(5):1103–1113.
- [Jaiswal et al., 2020] Jaiswal, A., Babu, A. R., Zadeh, M. Z., Banerjee, D., and Makedon, F. (2020). A survey on contrastive self-supervised learning. *Technologies*, 9(1):2.
- [Kang and Zhao, 2022] Kang, D. and Zhao, Y.-P. (2022). Predicting the molecular models, types, and maturity of kerogen in shale using machine learning and multi-nmr spectra. *Energy & Fuels*.
- [Kennedy and Eberhart, 1995] Kennedy, J. and Eberhart, R. (1995). Particle swarm optimization. In *Proceedings of ICNN’95-international conference on neural networks*, volume 4, pages 1942–1948. IEEE.
- [Kim et al., 2020] Kim, P., Winter, R., and Clevert, D.-A. (2020). Deep protein-ligand binding prediction using unsupervised learned representations.
- [Kipf and Welling, 2016a] Kipf, T. and Welling, M. (2016a). Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308v*.
- [Kipf and Welling, 2016b] Kipf, T. N. and Welling, M. (2016b). Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*.
- [Kirillov Jr, 2008] Kirillov Jr, A. (2008). *An introduction to Lie groups and Lie algebras*. Number 113. Cambridge University Press.
- [Krizhevsky et al., 2012] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25.
- [Le et al., 2020] Le, T., Winter, R., Noé, F., and Clevert, D.-A. (2020). Neuraldecipher–reverse-engineering extended-connectivity fingerprints (ecfps) to their molecular structures. *Chemical science*, 11(38):10378–10389.
- [LeCun et al., 1995] LeCun, Y., Bengio, Y., et al. (1995). Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10):1995.
- [LeCun et al., 2015] LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *nature*, 521(7553):436–444.
- [LeCun et al., 1998] LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

- [Lounkine et al., 2012] Lounkine, E., Keiser, M. J., Whitebread, S., Mikhailov, D., Hamon, J., Jenkins, J. L., Lavan, P., Weber, E., Doak, A. K., Côté, S., et al. (2012). Large-scale prediction and testing of drug activity on side-effect targets. *Nature*, 486(7403):361–367.
- [Mauser and Guba, 2008] Mauser, H. and Guba, W. (2008). Recent developments in de novo design and scaffold hopping. *Current opinion in drug discovery & development*, 11(3):365–374.
- [Mayr et al., 2016] Mayr, A., Klambauer, G., Unterthiner, T., and Hochreiter, S. (2016). Deeptox: toxicity prediction using deep learning. *Frontiers in Environmental Science*, 3:80.
- [McNaught, 2006] McNaught, A. (2006). The iupac international chemical identifier. *Chemistry international*, pages 12–14.
- [Miller et al., 2020] Miller, B. K., Geiger, M., Smidt, T. E., and Noé, F. (2020). Relevance of rotationally equivariant convolutions for predicting molecular properties. *arXiv preprint arXiv:2008.08461*.
- [Morgan, 1965] Morgan, H. L. (1965). The generation of a unique machine description for chemical structures—a technique developed at chemical abstracts service. *Journal of chemical documentation*, 5(2):107–113.
- [Narayanan et al., 2017] Narayanan, A., Chandramohan, M., Venkatesan, R., Chen, L., Liu, Y., and Jaiswal, S. (2017). graph2vec: Learning distributed representations of graphs. *arXiv preprint arXiv:1707.05005*.
- [Olivecrona et al., 2017] Olivecrona, M., Blaschke, T., Engkvist, O., and Chen, H. (2017). Molecular de-novo design through deep reinforcement learning. *Journal of cheminformatics*, 9(1):1–14.
- [Pan et al., 2018] Pan, S., Hu, R., Long, G., Jiang, J., Yao, L., and Zhang, C. (2018). Adversarially regularized graph autoencoder for graph embedding. *arXiv preprint arXiv:1802.04407*.
- [Polishchuk et al., 2013] Polishchuk, P. G., Madzhidov, T. I., and Varnek, A. (2013). Estimation of the size of drug-like chemical space based on gdb-17 data. *Journal of computer-aided molecular design*, 27(8):675–679.
- [Putin et al., 2018] Putin, E., Asadulaev, A., Vanhaelen, Q., Ivanenkov, Y., Aladinskaya, A. V., Aliper, A., and Zhavoronkov, A. (2018). Adversarial threshold neural computer for molecular de novo design. *Molecular pharmaceutics*, 15(10):4386–4397.
- [Qiu et al., 2018] Qiu, J., Dong, Y., Ma, H., Li, J., Wang, K., and Tang, J. (2018). Network embedding as matrix factorization: Unifying deepwalk, line, pte, and node2vec. In *Proceedings of the eleventh ACM international conference on web search and data mining*, pages 459–467.

- [Rogers and Hahn, 2010] Rogers, D. and Hahn, M. (2010). Extended-connectivity fingerprints. *Journal of chemical information and modeling*, 50(5):742–754.
- [Ruddigkeit et al., 2012] Ruddigkeit, L., Van Deursen, R., Blum, L. C., and Reymond, J.-L. (2012). Enumeration of 166 billion organic small molecules in the chemical universe database gdb-17. *Journal of chemical information and modeling*, 52(11):2864–2875.
- [Schmidhuber, 2015] Schmidhuber, J. (2015). Deep learning in neural networks: An overview. *Neural networks*, 61:85–117.
- [Schneider and Fechner, 2005] Schneider, G. and Fechner, U. (2005). Computer-based de novo design of drug-like molecules. *Nature Reviews Drug Discovery*, 4(8):649–663.
- [Schwalbe-Koda and Gómez-Bombarelli, 2020] Schwalbe-Koda, D. and Gómez-Bombarelli, R. (2020). Generative models for automatic chemical design. In *Machine Learning Meets Quantum Physics*, pages 445–467. Springer.
- [Segler et al., 2018] Segler, M. H., Kogej, T., Tyrchan, C., and Waller, M. P. (2018). Generating focused molecule libraries for drug discovery with recurrent neural networks. *ACS central science*, 4(1):120–131.
- [Stewart et al., 2015] Stewart, D., Stewart, A., Wheatley-Price, P., and et al (2015). Impact of time to drug approval on potential years of life lost: The compelling need for improved trial and regulatory efficiency. In *16th World Conference on Lung Cancer*.
- [Sun et al., 2019] Sun, F.-Y., Hoffmann, J., Verma, V., and Tang, J. (2019). Info-graph: Unsupervised and semi-supervised graph-level representation learning via mutual information maximization. *arXiv preprint arXiv:1908.01000*.
- [Sutskever et al., 2014] Sutskever, I., Vinyals, O., and Le, Q. V. (2014). Sequence to sequence learning with neural networks. *Advances in neural information processing systems*, 27.
- [Sutton and Barto, 1998] Sutton, R. S. and Barto, A. G. (1998). Reinforcement learning: an introduction mit press. *Cambridge, MA*, 22447.
- [Todeschini and Consonni, 2008] Todeschini, R. and Consonni, V. (2008). *Handbook of molecular descriptors*. John Wiley & Sons.
- [Tschannen et al., 2018] Tschannen, M., Bachem, O., and Lucic, M. (2018). Recent advances in autoencoder-based representation learning. *arXiv preprint arXiv:1812.05069*.
- [Van Westen et al., 2011] Van Westen, G. J., Wegner, J. K., IJzerman, A. P., Van Vlijmen, H. W., and Bender, A. (2011). Proteochemometric modeling as a tool to design selective compounds and for extrapolating to novel targets. *MedChemComm*, 2(1):16–30.

- [Vaswani et al., 2017] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 30.
- [Wale and Karypis, 2009] Wale, N. and Karypis, G. (2009). Target fishing for chemical compounds using target-ligand activity data and ranking based methods. *Journal of chemical information and modeling*, 49(10):2190–2201.
- [Wang et al., 2016] Wang, D., Cui, P., and Zhu, W. (2016). Structural deep network embedding. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1225–1234.
- [Wang et al., 2019] Wang, S., Guo, Y., Wang, Y., Sun, H., and Huang, J. (2019). Smiles-bert: large scale unsupervised pre-training for molecular property prediction. In *Proceedings of the 10th ACM international conference on bioinformatics, computational biology and health informatics*, pages 429–436.
- [Wang et al., 2012] Wang, S., Li, Y., Wang, J., Chen, L., Zhang, L., Yu, H., and Hou, T. (2012). Admet evaluation in drug discovery. 12. development of binary classification models for prediction of hERG potassium channel blockage. *Molecular pharmaceutics*, 9(4):996–1010.
- [Weininger, 1988] Weininger, D. (1988). Smiles, a chemical language and information system. 1. introduction to methodology and encoding rules. *Journal of chemical information and computer sciences*, 28(1):31–36.
- [Winter et al., 2022] Winter, R., Bertolini, M., Le, T., Noé, F., and Clevert, D.-A. (2022). Unsupervised learning of group invariant and equivariant representations. *arXiv preprint arXiv:2202.07559*.
- [Winter et al., 2019a] Winter, R., Montanari, F., Noé, F., and Clevert, D.-A. (2019a). Learning continuous and data-driven molecular descriptors by translating equivalent chemical representations. *Chemical science*, 10(6):1692–1701.
- [Winter et al., 2019b] Winter, R., Montanari, F., Steffen, A., Briem, H., Noé, F., and Clevert, D.-A. (2019b). Efficient multi-objective molecular optimization in a continuous latent space. *Chemical science*, 10(34):8016–8024.
- [Winter et al., 2021a] Winter, R., Noé, F., and Clevert, D.-A. (2021a). Auto-encoding molecular conformations. *arXiv preprint arXiv:2101.01618*.
- [Winter et al., 2021b] Winter, R., Noé, F., and Clevert, D.-A. (2021b). Permutation-invariant variational autoencoder for graph-level representation learning. *Advances in Neural Information Processing Systems*, 34.
- [Winter et al., 2020] Winter, R., Retel, J., Noé, F., Clevert, D.-A., and Steffen, A. (2020). grūnifai: interactive multiparameter optimization of molecules in a continuous vector space. *Bioinformatics*, 36(13):4093–4094.

- [Wouters et al., 2020] Wouters, O. J., McKee, M., and Luyten, J. (2020). Estimated research and development investment needed to bring a new medicine to market, 2009-2018. *Jama*, 323(9):844–853.
- [Xiong et al., 2019] Xiong, Z., Wang, D., Liu, X., Zhong, F., Wan, X., Li, X., Li, Z., Luo, X., Chen, K., Jiang, H., et al. (2019). Pushing the boundaries of molecular representation for drug discovery with the graph attention mechanism. *Journal of medicinal chemistry*, 63(16):8749–8760.
- [Xu et al., 2018] Xu, K., Hu, W., Leskovec, J., and Jegelka, S. (2018). How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*.
- [Yang et al., 2019] Yang, K., Swanson, K., Jin, W., Coley, C., Eiden, P., Gao, H., Guzman-Perez, A., Hopper, T., Kelley, B., Mathea, M., et al. (2019). Analyzing learned molecular representations for property prediction. *Journal of chemical information and modeling*, 59(8):3370–3388.
- [Ying et al., 2018] Ying, R., He, R., Chen, K., Eksombatchai, P., Hamilton, W. L., and Leskovec, J. (2018). Graph convolutional neural networks for web-scale recommender systems. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 974–983.
- [Zang et al., 2017] Zang, Q., Mansouri, K., Williams, A. J., Judson, R. S., Allen, D. G., Casey, W. M., and Kleinstreuer, N. C. (2017). In silico prediction of physicochemical properties of environmental chemicals using molecular fingerprints and machine learning. *Journal of chemical information and modeling*, 57(1):36–49.