Leveraging AlphaFold 2 as a General-Purpose Architecture for Small Molecule Conformer Prediction

Candidate no. 1060676

Thesis word count: 10195

A thesis submitted for the degree of
Masters by Research in Statistics

Michaelmas 2023

Abstract

Small molecules naturally exist in several three-dimensional conformations, and these conformations dictate a small molecule's interactions and functions. Thus, learning the three-dimensional structure of small molecules is an important problem in research and drug discovery. Originally, the conformation of small molecules was determined through expensive experimental methods. The first computational approaches to small molecule conformer generation (MCG) relied on random or "knowledge-based" searches for low-energy conformations. More recently, deep learning approaches have become popular for MCG. The major advantage of deep learning methods is their relatively high speeds and low costs. Here, we build on AlphaFold2's architecture for the purpose of MCG. We explore how various architectures and loss functions modify performance for MCG. Our highest performing model, termed OpenMol, achieves a mean (median) COV-R and COV-P of 81.4% (100%) and 83.1% (100%), respectively on the GEOM-QM9 dataset. Additionally, we introduce triangle-aligned point error (TAPE) loss, a novel loss function for MCG tasks.

Leveraging AlphaFold 2 as a General-Purpose Architecture for Small Molecule Conformer Prediction



Candidate no. 1060676

Word count: 10195

A thesis submitted for the degree of

Masters by Research in Statistics

☐ I hereby certify that this is entirely my own work unless otherwise stated.

Michaelmas 2023

Acknowledgements

I would like to thank my advisors Professors Garrett Morris, Xiaowen Dong, and Michael Bronstein. I would also like to thank Dr. Nazim Bouatta for supervising this project and Wojtek Treyde for helping get the project off the ground. I would also like to thank the Department of Statistics at the University of Oxford for supplying the resources for me to complete this project. Finally, I would like to thank the Rhodes Trust for its support and funding.

Abstract

Small molecules naturally exist in several three-dimensional conformations, and these conformations dictate a small molecule's interactions and functions. Thus, learning the three-dimensional structure of small molecules is an important problem in research and drug discovery. Originally, the conformation of small molecules was determined through expensive experimental methods. The first computational approaches to small molecule conformer generation (MCG) relied on random or "knowledge-based" searches for low-energy conformations. More recently, deep learning approaches have become popular for MCG. The major advantage of deep learning methods is their relatively high speeds and low costs. Here, we build on AlphaFold2's architecture for the purpose of MCG. We explore how various architectures and loss functions modify performance for MCG. Our highest performing model, termed OpenMol, achieves a mean (median) COV-R and COV-P of 81.4% (100%) and 83.1% (100%), respectively on the GEOM-QM9 dataset. Additionally, we introduce triangle-aligned point error (TAPE) loss, a novel loss function for MCG tasks.

Contents

List of Figures **vi**

List of Abbreviations **viii**

1 Introduction **1**

2 Background **4**
2.1 Molecular Conformer Generation 4
2.1.1 Stochastic Methods 4
2.1.2 Systematic Methods 7
2.1.3 Bayesian Methods 8
2.1.4 Deep Learning Methods 8
2.2 Protein Folding 12

3 Methods **16**
3.1 Benchmark Datasets 16
3.2 Evaluation metrics 17
3.3 Base Model Architecture 19
3.3.1 Input filtering and featurization 19
3.3.2 Evoformer 19
3.3.3 Structure Module 20
3.3.4 Loss 20
3.3.5 Hyperparameters 22
3.4 Experimental Model Architectures 23
3.4.1 3D Evoformer 23
3.4.2 Modified Loss Functions 24

4 Results **29**
4.1 Baseline 29
4.2 Kabsch loss 31
4.3 3D Evoformer 38
4.4 TAPE loss 39

5 Discussion **42**

6 Conclusion **47**

References **48**

List of Figures

1 Example of four conformers for a small organic compound from the GEOM-QM9 dataset, demonstrating the task of molecular conformer generation. (a) The original molecular structure and (b) four of the possible 3D conformations the molecule can take on. 2

2 Associations between COV-R and COV-P and the number of rotatable bonds in each molecule for the baseline and Kabsch loss models. Associations between rotatable bonds and (a) COV-R or (b) COV-P in the baseline model. Association between rotatable bonds and (c) COV-R or (d) COV-P in the Kabsch loss model. 30

3 Associations between COV-R and COV-P in the baseline and Kabsch loss models and the number of reference conformers associated with each molecule in the testing set colored by the number of rotatable bonds. Associations between number of reference conformers and (a) COV-R or (b) COV-P in the baseline model. Association between the number of reference conformers and (c) COV-R or (d) COV-P in the Kabsch loss model. 31

4 Associations between AMR-R and AMR-P in the baseline and Kabsch loss models and the number of reference conformers associated with each molecule in the testing set colored by the number of rotatable bonds. Associations between number of reference conformers and (a) AMR-R or (b) AMR-P in the baseline model. Association between the number of reference conformers and (c) AMR-R or (d) AMR-P in the Kabsch loss model. 34

5 Associations between AMR-R and AMR-P in the baseline and Kabsch loss models and the number of reference conformers associated with each molecule in the testing set colored by the number of rotatable bonds. Associations between number of reference conformers and (a) AMR-R or (b) AMR-P in the baseline model. Association between the number of reference conformers and (c) AMR-R or (d) AMR-P in the Kabsch loss model. 35

6 Histogram of difference between each molecule's generated and ground truth (a) intra-molecuar average RMSD and (b) intra-molecular max RMSD. 36
7 Representative examples of small molecules with COV-P and COV-R values of 0. (a) Example of diastereomer and (b,c) examples of enantiomers. 37
8 Visualization of how the TAPE alignment steps look. (a) Image demonstrating how non-overlapping triangles are assigned to an arbitrary molecule. (b) Diagram of how alignment between two triangles takes place after they are both rotated onto the same plane. 39
9 Figures summarizing the series of tests performed to characterize the TAPE loss. (a) Average conversion of 10 highly varied structures from black hole initialization. (b) Average conversion of 10 highly varied structures from random initialization. (c) Error vs injected noise standard deviation plot for TAPE and Kabsch losses. (d) Example of one of the 10 learned small molecules at the 300th step of the conversion tests. 41

List of Abbreviations

2D, 3D . . . . Two- or three-dimensional, refers to n-dimensional Euclidean spaces.

MCG . . . . . Molecular conformer generation.

MMFF94 . . . Merck Molecular force field 94

RMSD . . . . . Root-mean-square distance.

FAPE . . . . . Frame-aligned point error.

TAPE . . . . . Triangle-aligned point error.

COV-R . . . . Coverage, recall.

COV-P . . . . Coverage, precision.

AMR-R . . . . Average minimum RMSD, recall.

AMR-P . . . . Average minimum RMSD, precision.

1 Introduction

Small molecules play a major role in drug discovery and scientific research. Central to the function of a small molecule is its 3D structure, which dictates how a small molecule interacts with its environment. Because the 3D structure of a small molecule is pivotal to its function, learning how small molecules arrange themselves in space is a crucial outstanding problem. A small molecule rarely adheres to a single, fixed 3D conformation; instead, it can exist in a range of energetically favorable 3D conformations, often referred to as conformers (Fig. 1). Deciphering the ensemble of potential conformers for a given small molecule remains an open scientific problem, complicated by variability in bond lengths, bond angles, torsion angles, and by the complexity of inter-atomic interactions.[1]

Determining the set of conformers for a given small molecule with complete certainty is currently only possible through an experimental method called small molecule crystallography.[2] In small molecule crystallography, molecules are crystallized and X-rays are shot at the crystallized structure. The pattern with which the X-rays diffract off the crystallized structure provides sufficient information to determine the small molecule's structure. Although very accurate, such experimental determination of molecular structure is time intensive and expensive. Thus, much work has been done to computationally predict small molecule conformations, with hopes of circumventing the need for such experiments. The process of

determining the conformers of a small molecule is referred to as molecular conformer generation (MCG).

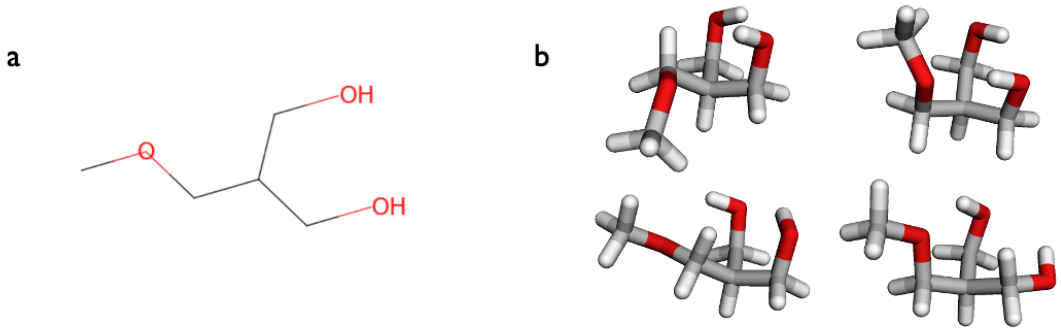


Figure 1: Example of four conformers for a small organic compound from the GEOM-QM9 dataset, demonstrating the task of molecular conformer generation. (a) The original molecular structure and (b) four of the possible 3D conformations the molecule can take on.

Initial methods for MCG were based on searching the space of potential conformers through either iterating over established patterns of small molecule geometry or randomly generated structures. Recently, deep learning methods have achieved state-of-the-art results in MCG tasks, enabling rapid and accurate generation of conformers. Here we introduce a new deep-learning model for MCG named OpenMol. OpenMol is heavily inspired by AlphaFold2, the first deep-learning architecture to be considered a near solution to the protein folding problem.[3] To build OpenMol, we build off of the OpenFold code base, an open-source version of AlphaFold2 released in 2022.[4]

We develop OpenMol based on the AlphaFold2 architecture for two reasons. The first is that AlphaFold2's primary task of protein structure prediction is very similar to the problem of MCG. Both aim to generate a physical 3D structure from a limited set of components (amino acids or atoms) where the primary source of variability comes from the interactions between the components and not the components themselves. Further, booth small molecules and proteins are easily represented in either one, two, or three dimensions. Thus, the majority of AlphaFold2's architecture is not specific to proteins and is readily transferable to the

task of MCG. Validation for this hypothesis - that AlphaFold2 could be a general-purpose architecture applicable to MCG - comes from UniMol, which uses a highly similar structure to AlphaFold2 to refine small molecule conformers.[5] Our method differs from UniMol in two key ways. First, OpenMol does not use pertaining; second, it takes 1D sequence representations and 2D graphs as input rather than using a secondary method to generate the initial structure (as in UniMol). By developing OpenMol, we are interested in exploring the potential of AlphaFold2 not just as a protein structure prediction architecture, but as a general-purpose model for solving structure prediction problems.

The secondary motivation for basing the architecture off of AlphaFold2 is that advancements made in the development of OpenMol can be translated to the case of proteins and improving the AlphaFold2 architecture. Because we are using such a similar model, we can use OpenMol to gain insights into the AlphaFold2 architecture and inspire modifications to the algorithm. Alongside the development of OpenMol, we also develop the triangle-aligned point error (TAPE) loss, a novel loss function for MCG.

2
Background

Contents

2.1 Molecular Conformer Generation 4
2.1.1 Stochastic Methods 4
2.1.2 Systematic Methods 7
2.1.3 Bayesian Methods 8
2.1.4 Deep Learning Methods 8
2.2 Protein Folding 12

2.1 Molecular Conformer Generation

MCG has been an active field of research for the past thirty years, and methods typically fall into one of four major categories: stochastic, systematic/rules-based, Bayesian, and deep learning.

2.1.1 Stochastic Methods

Stochastic MCG methods rely on random sampling and subsequent scoring of conformers. The four main types of stochastic methods are molecular dynamics, Monte Carlo-simulated annealing, distance geometry, and genetic algorithms. Molecular dynamics approximate the energy of a particular conformer by simulating the motions of the atoms in the molecule and the forces arising from interacting

atoms.[6] The conformers with the lowest energy generated are then selected and returned as the predicted conformers. Stochastic methods using molecular dynamics have largely fallen out of favor due to their high computational costs.

Monte Carlo

In Monte Carlo-simulated annealing, a conformer is initialized according to some heuristics or prior knowledge and then undergoes random structural modifications. If the modification results in a net decrease in energy, the structure is always kept; otherwise, the probability of keeping the modifications scales inversely with the positive energy change.[7] One example of a Monte Carlo simulated annealing method is MED-3DMC, which improved on previous methods by using Merck Molecular force field 94 (MMFF94) for its energy calculations. MMFF94 is an equation that approximates the molecule-mechanics force field around a molecule.[8] The MMF94 algorithm's main advantage over prior methods is that it approximates all physical interactions that contribute to a molecule's force field. The MMFF94 final energy term is a sum of the following energy terms: bond stretching, angle bending, stretch-bend interactions, out-of-plane bending tricoordinate centres, torsion interactions, Van der Waals interactions, and electrostatic interactions. Equation 1 shows the full MMFF94 formula.[8]

$$E_{MMFF} = \sum EB_{ij} + \sum EA_{ijk} + \sum EBA_{ijk} + \sum EOOP_{ijk;l} + \sum ET_{ijkl} + \sum EvdW_{ij} + \sum EQ_{ij} \quad (1)$$

Most of these calculations are derived from first-principles chemical and physics laws grounded in theoretical work. In contrast to preceding methods, experimental data is not used in the energy calculation because the authors wanted MMFF94 to readily apply to molecules not present in the available experimental data. MED-3DMC used the MMFF94 Van der Waals energy term specifically and showed similar performance to OMEGA (a popular systematic method that will be described in the following subsection) and a negative correlation between accuracy and the number

of rotatable bonds.[9] However, like all Monte Carlo-simulated annealing methods, MED-3DMC suffers from bias based on the choice of initial conformer.

FROG2 is another stochastic approach that uses a two-step Monte-Carlo sampling. FROG2 first segments an input 2D molecular graph into rings and acyclic components. Then, rings conformations are sampled from an online ring structure library. Acyclic components are subsequently initialized according to known bond angle patterns and optimized by MMFF94. The first Monte-Carlo step samples a known distribution of dihedral angles for the specific set of atoms involved in the bond. The second Monte-Carlo step uniformly samples small rotations around each conformation from step one, with preferential adjustment of conformers above a certain energy threshold.[10, 11] FROG2 is available online, and its main advantage is speed. When compared against other open-source MCG methods, FROG2 was the fastest by an order of magnitude.[12]

Distance Geometry

Distance geometry methods use physical constraints to identify potential conformers for a given molecule. An advantage of distance geometry is it is unbiased by the initial choice of conformer, an issue faced by both molecular dynamics and Monte Carlo simulated annealing. In newer differential geometry methods, a refinement step is included after the geometric constraints are applied. An early distance geometry approach is implemented in the widely used RDKit software package and consists of the following steps: initialization of bounds matrix representing the upper and lower bound of each interatomic distance, setting bounds for 1,2-, 1,3-, 1,4-, and 1,5- distances, smoothing bounds with the triangle inequality, generating a random conformer that fits the bounds, refining that conformer against an error function that measures physical constraint violations, and finally removing all conformations within some distance of each other to remove any redundancy (a typical threshold would be root mean squared distance of 0.5Å).[13–15] Experimental Torsion Distance Geometry generation with additional basic knowledge (ETKDG) is a popular method that builds off of the above-described distance geometry

framework. The first improvement is that ETKDG searches the input molecule for torsion patterns with experimentally determined distributions. These torsion patterns are then optimized against the known torsion angle distributions. Finally, the "basic knowledge" in ETKDG refers to a set of rules used in optimizing the conformers that incorporates information about aromatic rings, sp^2 atoms, and triple bonds. Because of ETKDG's performance and incorporation into the RDKit software package, it is an often-used baseline measure for newly developed methods.[15]

2.1.2 Systematic Methods

Systematic (also referred to as rules-based) methods rely on sets of predefined rules that are based on patterns observed in large datasets, such as the Protein Databank or the Cambridge Structural Database. Examples of rules could be a pre-defined distribution of torsion angles or bond lengths. Confab and OMEGA are two of the most popular systematic methods.

Confab is a straightforward approach that works by assigning a set of torsion angles to each rotatable bond in the molecule and then generating by default 10^6 conformers. If there are more than 10^6 possibilities, a random subset is generated. The lowest energy conformers as determined by MMFF94 are kept and redundant conformers are removed (as in the RDKit distance geometry method described in the previous section). Although Confab's performance is competitive with other methods, it is unable to work on molecules that contain rings or no rotatable bonds.[16] The OMEGA algorithm first establishes a fragment database and torsion sampling dictionary, neither of which are molecule specific. The fragment database consists of fragments of commercially available compounds. Each fragment is associated with 3D coordinates calculated by physical constraints and MMFF94-based energy minimization. The torsion sampling dictionary is constructed by associating known torsion patterns with torsion angles based on data from the Protein Data Bank and MMFF94-based energy estimations. The first molecule-specific steps of OMEGA are then matching a molecule's fragments and torsion patterns to the fragment database and torsion sampling dictionary, respectively.

The resulting generated conformers are then filtered to keep the conformers without clashes and the lowest energy is determined by a modified MMFF94.[17] Evaluation of OMEGA's performance showed that though the method struggles with flexible rings, it is successful in generating conformers, even for those with large numbers of rotatable bonds.[18] An evaluation of commercially available MCG software showed that OMEGA is both the fastest and most accurate.[19]

2.1.3 Bayesian Methods

In addition to stochastic and systematic methods, there have been recent advances in MCG from Bayesian optimization approaches. One such example is Bayesian optimization approach (BOA).[20] BOA works by first sampling conformations based on torsion angles with a prior bias toward sampling more commonly occurring torsion angles. Evaluation of conformers during the search was done using the MMFF94. The authors showed that BOA found the lowest energy conformation of a molecular more rapidly that other algorithms, especially for molecules with fewer rotatable bonds. Notably, BOA is set around determining only the lowest energy conformation and not a diverse set of small molecule conformations, which is often desired. A later method, Bayesian Optimization with Knowledge-based Expected Improvement (BOKEI) built on the success of BOA.[21] BOKEI builds upon BOA by incorporating a set of constraints based on previously described rotatable bond patterns. BOKEI was more successful than BOA, finding lower energy conformers for more than 60% of test cases. However, BOKEI suffers the same drawback of BOA by only finding the lowest energy conformer as opposed to a set of multiple conformers.

2.1.4 Deep Learning Methods

Deep learning-based methods have the advantage of fast prediction time and can quickly generate variably sized ensembles of conformers for a given small molecule. Deep learning methods for MCG generally fall into three categories: graph neural networks, traditional deep learning, and diffusion models.

Graph neural networks

All small molecules can be represented as undirected graphs, with nodes as atoms and bonds as edges. Thus, they naturally lend themselves to graph neural networks. The core idea of graph neural networks is to propagate information between nodes along edges. Thus, information is embedded in a way that preserves information about the graph's structure. The most common type of graph neural network, and the kind that has been used for MCG, is referred to as a message passing neural network. In a message passing network, nodes update their neighbors' representations by "passing" information along edges.

The first method to use such graph neural networks for MCG was GraphDG, a variational autoencoder that uses a message passing network during encoding.[22] GraphDG first creates an additional edge between all atoms that are two bonds apart and random edges between atoms three bonds apart. Euclidean distances between atoms are then added to edges as additional features. Euclidean distances are used in place of coordinates because they are invariant to rotation and translation. This resulting graph is embedded using a message passing neural network followed by a multilayer perceptron to obtain a Gaussian distribution in latent space z. Alongside the graph, the set of distances between all atoms in the molecule are encoded. A decoder then transforms z into a distribution of distances. This resulting distribution is subsequently used to generate an ensemble of conformers. GraphDG achieved state-of-the-art results but tended to create too much inter-conformer variation.[22]

A new method named Conditional Graph Continuous Flow (CGCF) bested the results of GraphDG using a network similarly integrating probabilistic generative models and message passing neural networks.[23] In CGCF, the molecule embedding process is similar to the one in GraphDG. However, rather than incorporating the molecule's inter-atomic distances, CGCF embeds distances sampled from a Gaussian distribution. The conditional distribution of pair-wise inter-atomic distances for a given molecular graph is then represented as a conditional graph continuous flow. Unlike GraphDG, which converts values from inter-atomic distance distribution to coordinates, CGCF learns a distribution of conformers conditioned on the distance

distribution. This enables CGCF to downstream directly optimize the distribution of conformers conditioned on the input molecule. CGCF achieves superior results to GraphDG on both the GEOM-DRUGS and GEOM-QM9 datasets (these datasets are described in detail in Chapter 3:Methods).[23]

The current state-of-the-art message-passing method is GeoMol.[24] Unlike GraphDG and CGCF, GeoMol is not a probabilistic model. Molecular graphs are input alongside random noise. The random noise enables the generation of an ensemble of conformers. GeoMol takes an iterative approach to generating conformers. First, the position of each non-terminal atom's bonded atoms are independently predicted. These predictions are then stitched together through a neural network that predicts the torsion angles between non-terminal atoms. GeoMol also introduces a new optimal transport loss. The optimal transport loss enables the comparison of an ensemble of conformers against variably sized ensembles of ground truth conformers.[24] The optimal transport loss is discussed further in Chapter 3:Methods.

Diffusion Models

Diffusion neural networks are a category of probabilistic generative models that operate in two steps. The first step is the noising process, where noise is injected until the input data loses all structure. The second step is the denoising process, where the network learns to reverse the noising process, thus recovering a distribution for the input data from which new samples can be generated. Diffusion models are the current state-of-the-art in image generation and are being applied to numerous other problems, including MCG.[25] The first method that applied diffusion to MCG was GeoDiff.[26]. GeoDiff begins with 3D structures and progressively adds noise to the atom coordinates until a Gaussian distribution is reached. Geodiff then learns a process that will recover the input conformers from noise given the target molecule's 2D graph. GeoDiff ensures that the initial noise is an isotropic Gaussian centered at 0, making it invariant to rotations and translations (the SE(3) transformations). By then ensuring each denoising step is SE(3)-equivariant, GeoDiff recovers an

SE(3)-invariant representation of the input. They prove this by showing that a series of SE(3)-equivariant Markov transitions on an initial SE(3)-invariant distribution results in an SE(3)-invariant distribution. GeoDiff achieved state-of-the-art results using their model alone and found even further improved results with force field optimization using MMFF94.[26]. Torsional Diffusion was released shortly after GeoDiff. Rather than adding noise to 3D atom coordinates as in GeoDiff, the authors of Torsional Diffusion add noise to the input conformer's torsion angles. Torsion angle coordinates lie in [0, 2pi) and define an n-dimensional hypertorus, where n is the number of torsion angles in the conformer. Since the space of torsion angles has significantly fewer degrees of freedom than Euclidean space, Torsional Diffusion is far more efficient than GeoDiff. While GeoDiff requires 5000 denoising steps, Torsional Diffusion reaches optimal performance at 20 denoising steps and surpasses GeoDiff's performance with just 5 denoising steps. These reductions in denoising steps translate to run times that are 173 and 60 times faster for 5 and 20 denoising steps, respectively. Torsional Diffusion is the current state-of-the-art method, with similar performance for smaller molecules (GEOM-QM9) and significantly improved results for larger molecules (GEOM-DRUGS).[27]

Other Deep Learning Approaches

A recent method named UniMol achieved results comparable to Torsional Diffusion using a more conventional deep learning architecture, relying primarily on self-attention.[5] Unlike previously described methods, UniMol uses pretraining before training the model on the MCG task. Pretraining is inspired by large language models, where models are trained to fill in the text with removed words. In UniMol, rather than removing atom coordinates, random atoms' positions are "corrupted" and then the pretraining is done by predicting the correct position. The atom type is additionally masked and predicted during UniMol's pretraining. For training the MCG task, UniMol takes as input 3D conformations, rather than molecular graphs, as in the previously described methods. These input conformers are generated by using ETKDG to generate up to 2,000 conformers and then clustering them

into 10 representative conformers. Thus, the UniMol framework can be thought of as a fine-tuning model on top of ETKDG.

UniMol's architecture is heavily inspired by AlphaFold2, which will be covered in depth in the following subsection.[3, 5] In brief, UniMol simultaneously represents each atom during training using a linear "atom representation" and a 2D "pair representation." The atom representation consists of all the atoms in the molecule written in a sequence and the 2D representation is a matrix where each entry represents the relationship between every pair of atoms in the molecule. In addition to this multi-track network structure, UniMol also uses weight "recycling" from AlphaFold2. UniMol's results narrowly improve over GeoDiff, making it the second most accurate MCG algorithm following Torsional Diffusion.

While deep learning approaches appear to be quickly outperforming traditional methods, a recent paper questioned these results.[28] Gengmo Zhou et. al. showed that a simple method that generated roughly 2000 conformers per molecule using a stochastic approach followed by kmeans clustering outperformed nearly all existing methods. This clustering approach inspired the clustering method used in UniMol, demonstrating that UniMol may be exploiting issues in the datasets to achieve its high performance. The results from this clustering approach fall short only of Torsional Diffusion, which they did not directly compare against. This paper demonstrates that the common benchmarks the field uses are not necessarily signs of valuable results.

2.2 Protein Folding

Proteins are the molecules responsible for carrying out nearly all functions in living organisms. Similar to how words are constructed from an alphabet, all proteins are constructed as linear strings where each element comes from the same set of 20 amino acids. This set of amino acids is consistent across all living organisms. Each amino acid contains a central carbon atom to which three groups are attached: a carboxylic acid, an amine, and a side chain. The side chain is the only variable section of an amino acid. Proteins are constructed through the iterative, linear

addition of amino acids. Each new amino is incorporated by binding its amine with the carboxylic acid of the final amino acid in the current amino acid chain. The constructed chain of amino acids then takes on a complex three-dimensional structure based on a variety of interactions between all the amino acids in the protein. The process by which the protein takes on its final 3D structure is known as protein folding. Importantly, a given sequence of amino acids will, by and large, take on the same three-dimensional structure each time it folds.

Proteins function through their interactions with other proteins and molecules, and these interactions are dictated by a protein's structure. Determining the three-dimensional structure of a protein experimentally is expensive, requiring the use of X-ray crystallography. Thus, the task of learning some function that takes as input an amino acid sequence and outputs a three-dimensional protein structure has been one of the grand outstanding challenges in biology research. This task is known as protein structure prediction.

While there are decades of research exploring computational methods for protein structure prediction, the first attempt that has been regarded as a solution to the problem was AlphaFold2, a neural network architecture published in 2021.[3] AlphaFold2 was trained using structures from a publicly available database of protein crystal structures. AlphaFold2 takes as input an amino acid sequence of a protein alongside the amino acid sequence of the same protein across organisms. Many proteins are shared across organisms, as they are necessary for life-enabling cellular function. However, the same protein will vary in its amino acid sequence across organisms. Thus, for any given protein, a set of amino acid sequences that are highly similar can be obtained and aligned, resulting in a multiple-sequence alignment (MSA). The MSA provides information about how an amino acid has evolved over time, which also provides information about the 3D structure of the protein. For example, if amino acids in positions 2 and 50 in the amino acid sequence often vary together, it is more likely that they are also close in 3D space. AlphaFold2 relies on the MSA for protein structure prediction, with drastic decreases in performance observed if only a single amino acid sequence is provided as input.[29]

AlphaFold2 is split into two components: the Evoformer and the structure module. The Evoformer takes as input the MSA and from the MSA creates a one-dimensional sequence representation using self-attention.[30] This one-dimensional sequence representation is then used to create a pair representation, where each entry in the pair-representation matrix represents the relationship between two amino acids in the protein. Within the Evoformer, the pair representation matrix is updated in a pattern of triangles. This means that the entry (i, j) is updated by summing (k, i) and (k, j). Performing operations in this order enables the algorithm to ensure the triangle inequality is satisfied during learning without requiring any loss function that explicitly checks for inconsistencies.

In the structure module, each amino acid is assigned a "frame" at its central carbon. The frame defines a rotation and translation from the origin. Each frame is initialized identically at the origin in what is referred to as the "black hole initialization." As a result, AlphaFold2 makes no assumptions about the proximity of two amino acids in 3D space based on their proximity in sequence space. To update these frames, the structure module uses the "invariant point attention" algorithm to first update the sequence representation using both the frames and pair representation. This updated sequence representation is then used to predict quaternarions that define the rotational and translational updates to each amino acid's frames. These quaternarions are then applied to each frame. AlphaFold2's primary loss function is frame-aligned point error (FAPE). FAPE is calculated by individually aligning each amino acid's predicted and true frames and then calculating the pair-wise distance between all other frames. The error is then averaged over the total number of amino acids/alignments.[3]

Both the Evoformer and structure module are repeated several times, and a loss is calculated for each intermediate output of the structure module. These intermediate losses make up the auxiliary loss. Additionally, AlphaFold2 introduces recycling. Recycling is where the results from a layer are fed back into the model without backpropagation. This enables the algorithm to efficiently iterate several times through the model without the expensive backpropagation step.

AlphaFold2 was the first time an algorithm succeeded in predicting protein structures with an accuracy comparable to experimental structure determination. Since then, others have developed methods heavily inspired by AlphaFold2, but none have surpassed AlphaFold2 in prediction accuracy.

One of these architectures is ESMFold. ESMFold uses the same architecture as AlphaFold2 for prediction but generates a modified input.[29] Because AlphaFold2 relies on MSAs for protein structure prediction, it is unable to predict protein structure from sequence alone. This means AlphaFold2 fails in several important tasks, such as predicting the structure of amino acids, synthetic proteins, proteins without homologs across species, and proteins following some alteration to the amino acid sequence. To address this limitation, ESMFold sought to bypass the reliance on the MSA by first training a protein language model, similar to natural language models, except here the model learns to predict masked amino acids within proteins. Once trained, the protein language model can be used to develop high-dimensional embeddings of protein sequences that are then fed into the AlphaFold2 architecture. ESMFold s overall slightly underperforms AlphaFold2.[29]

A recently developed alternative to AlphaFold2 is RosettaFold2.[31] RosettaFold2 borrowed many ideas from AlphaFold2, notably FAPE and recycling. However, a major difference between RosettaFold2 and AlphaFold2 is that RosettaFold2's architecture represents proteins using three tracks. Two of these tracks are the same sequence and pair representation as in AlphaFold2. The third track, which AlphaFold2 does not have, is the 3D track. In the 3D track, the 3D structure is calculated earlier and is used to update information of the sequence and pair representations. RosettaFold2 slightly outperforms AlphaFold2 for prediction of some protein structures with a compute time that scales linearly with amino acid length as opposed to exponentially as in AlphaFold2.[31]

3 Methods

Contents

3.1 Benchmark Datasets 16
3.2 Evaluation metrics 17
3.3 Base Model Architecture 19
3.3.1 Input filtering and featurization 19
3.3.2 Evoformer 19
3.3.3 Structure Module 20
3.3.4 Loss 20
3.3.5 Hyperparameters 22
3.4 Experimental Model Architectures 23
3.4.1 3D Evoformer 23
3.4.2 Modified Loss Functions 24

3.1 Benchmark Datasets

Most of the recent work on improving MCG methods is done using two benchmark datasets: GEOM-QM9 and GEOM-DRUGS.[32]

The original QM9 dataset was released in 2014 and consists of 133,885 organic molecules composed of hydrogen and up to 9 of the following four atoms: carbon, oxygen, nitrogen, and fluorine. Cations and anions are removed as well, but zwitterions are kept because they are present in biologically-relevant small molecules such as amino acids. Structures were computed using quantum chemistry calculations

performed at the B3LYP/6-41G(2df,p) level, which is comparable to experimental accuracy. This method of calculation is often referred to as density functional theory.[32] To create GEOM-QM9, 133,258 molecules from QM9 were re-evaluated with the GFnN2-xTB level of theory.[33, 34] They were then input into CREST, an algorithm that generates ensembles of conformers for small molecules given a conformer optimized at the GFN2-xTB level of theory.[35] The size of molecules in the GEOM-QM9 dataset is summarized in table 1

The GEOM-DRUGS dataset consists of 304,466 drug-like molecules that were originally part of a machine learning drug discovery challenge for COVID-19.[34] Similar to GEOM-QM9, molecule structures were first estimated at the GFN2-xTB level of theory and the ensembles of conformers generated with CREST.[33, 35] GEOM-DRUGS contains larger molecules, making it a more complex baseline for MCG. Small molecules in GEOM-DRUGS have on average nearly triple the number of non-hydrogen atoms and rotatable bonds than molecules in GEOM-QM9, as shown in Table 1. GEOM-DRUGS is not used in the results in this paper due to GPU memory constraints caused by the handling of larger molecules.

Dataset	Atoms	Non-hydrogen atoms	Rotatable bonds
GEOM-QM9	18.0 ± 3.0	8.8 ± 0.51	2.2 ± 1.6
GEOM-DRUGS	44.4 ± 11.3	24.9 ± 5.7	6.5 ± 3.0

Table 1: Summary of GEOM datasets. Values reported as mean ± standard deviation.

3.2 Evaluation metrics

The similarity between two molecules is measured using root-mean-square distance (RMSD) (2) defined as:

$$RMSD := \sqrt{\frac{1}{n}\sum_{i=1}^{n} ||v_i - w_i||^2} \tag{2}$$

where n is the number of atoms in the molecules and v_i and w_i are the 3D positions of the predicted and ground truth atom i, respectively. RMSD is reported in angstroms (Å).

In this study, we use four evaluation metrics commonly used in measuring the performance of MCG algorithms: coverage recall (COV-R), coverage precision (COV-P), average minimum RMSD recall (AMR-R), and average minimum RMSD precision (AMR-P).

COV-R is the proportion of ground truth conformers that are represented in the generated ensemble. COV-P is the proportion of generated conformers that are represented in the ground truth ensemble. Values for COV-R (3) and COV-P (4) can be between 0% and 100% and higher values indicate better performance. COV-R is defined as:

$$COV - R := \frac{1}{|S_t|}|\{C \in S_t : RMSD(C, \hat{C}) < \delta, \exists \hat{C} \in S_g\}| \tag{3}$$

and COV-P is defined as:

$$COV - P := \frac{1}{|S_g|}|\{\hat{C} \in S_g : RMSD(\hat{C}, C) < \delta, \exists C \in S_t\}| \tag{4}$$

where S_t and S_g are the sets of true and generated conformers, respectively. C and $\hat{C}$ are individual examples of true and generated conformers, respectively, and δ is a threshold (Å) that defines the maximum RMSD value at which two conformers can be considered a match (or "covered"). Typically, $\delta = 0.5$Å for GEOM-QM9 and $\delta = 1.25$Å for GEOM-DRUGS.[24] A higher threshold is used for GEOM-DRUGS because of the larger size of the molecules.

In addition to COV, AMR provides a more granular evaluation of how similar the generated and ground truth conformers are. Values for AMR-R and AMR-P (in Å) can be any positive real number or zero. Lower values for AMR indicate better performance. AMR-R averages the RMSD between each ground truth conformer and its corresponding most similar conformer in the set of generated conformers. AMR-P switches the order, averaging the RMSD between each generated conformer and its most similar ground truth conformer. AMR-R is defined as:

$$AMR-R := \frac{1}{|S_t|} \sum_{\hat{C} \in S_t} \min_{C \in S_g} RMSD(C, \hat{C}) \tag{5}$$

and AMR-P is defined as:

$$AMR-P := \frac{1}{|S_g|} \sum_{C \in S_g} \min_{\hat{C} \in S_t} RMSD(\hat{C}, C) \tag{6}$$

where S_g, S_t, C, and $\hat{C}$ are as previously defined for COV-R and COV-P.

3.3 Base Model Architecture

3.3.1 Input filtering and featurization

OpenMol takes input molecules from the GEOM-QM9 or GEOM-DRUGS datasets. For each molecule input during training, the molecule's order of conformers is randomly shuffled. Conformers that are fragmented, have no dihedral angles, or that are the product of a reaction are removed. The following atom properties are used to initialize features vectors: chirality, atomic number, aromaticity, atom degree (one-hot encoding 0-6), hybridization (one-hot encoding sp, sp2, sp3, sp3d, sp3d2), implicit hydrogens (one-hot encoding 0-6), formal charge (one-hot encoding -1-1), number of rings involved in (one-hot encoding 0-3), ring size (3-8). For each of the one-hot encoded properties, an additional category is included for values that fall outside of the predefined ranges. Bond features are one-hot encodings of the type of bond (single, double, triple, aromatic). These properties are obtained using the corresponding RDKit functions. The code and choice of properties for initialization are the same as in GeoMol.[24]

3.3.2 Evoformer

The pair representation is initialized with an indicator for whether or not two atoms are directly bonded. Input molecule sequence representations are first updated using self-attention with bias coming from the pair representation. This is the same as "MSAAttentionWithPairBias" from Alphafold2.[3] MSA column attention is not used as there is no equivalent to MSA for small molecules. The remaining steps in

the Evoformer are identical to those in AlphaFold2. In brief, the pair representation is a square matrix where each entry represents the relationship between two amino acids in the protein. The pair representation is initialized by taking the outer product of the sequence representation with itself. The pair representation is then updated using triangular multiplicative updates and triangular attention updates. Each of these updates is performed twice, once for outgoing and incoming edges, thus making the matrix asymmetric. For further details on the algorithm, see the original AlphaFold2 paper.[3]

3.3.3 Structure Module

In the OpenMol structure module, atom coordinates are initialized at the origin, as in AlphaFold2.[3] However, unlike in AlphaFold2, each atom is represented only by a translation away from the origin (or simply a 3D coordinate). The lack of a rotation or orientation component to an atom's position is because there are no frames assigned to atoms. Despite this, the OpenMol structure module still uses invariant point attention, as it is in AlphaFold2, to update atom coordinates.[3] We chose to keep invariant point attention because its property of invariance still holds in the translation-only setting of OpenMol. We can show this by fixing the rotation matrix to the identity matrix.

3.3.4 Loss

The loss function in OpenMol has 5 components: the positional loss ($\mathcal{L}_{\text{pos}}$), the optimal transport loss ($\mathcal{L}_{\text{OT}}$), the distogram loss ($\mathcal{L}_{\text{dist}}$), the auxiliary positional loss ($\mathcal{L}_{\text{pos}_{\text{aux}}}$), and the auxiliairy optimal transport loss ($\mathcal{L}_{\text{OT}_{\text{aux}}}$). Equation 7 shows the full loss function. Weights are additionally assigned to each term in the loss function.

$$\mathcal{L}_{\text{final}} = \mathcal{L}_{\text{pos}} + \mathcal{L}_{\text{OT}} + \mathcal{L}_{\text{dist}} + \mathcal{L}_{\text{pos}_{\text{aux}}} + \mathcal{L}_{\text{OT}_{\text{aux}}} \tag{7}$$

Positional Loss

The positional loss is a modified RMSD taken from AlphaFold2 (Alg 1).[3] The loss contains an ϵ term to ensure there are no 0-value errors. The clamp value d_{clamp} helps prevent exploding gradients. The loss is additionally scaled by Z, so the result will always have a value between 0 and 1.

Algorithm 1 PositionalLoss

def PointwiseLoss($\{C\}, \{\hat{C}\}, \epsilon, d_{\text{clamp}}, Z$) :

1: $error = \sqrt{\sum_{i=1}^{n} ||C_i - \hat{C}_i||^2 + \epsilon}$

2: $error_{\text{clamped}} = \frac{1}{Z}\min(error, d_{\text{clamp}})$

3: **return** $error_{\text{clamped}}$

Distogram Loss

The distogram loss $\mathcal{L}_{dist}$ (Alg 2) calculates the average difference between the predicted distance between each pair of atoms and the true distance between each pair of atoms in a molecule. Since it is a loss over distances, the histogram loss is SE(3)-invariant. The distances used for the distogram loss are calculated by finding the inter-atomic distances in the predicted 3D structure. The distogram loss is not a loss over the pair representation, as it is in AlphaFold2.

Algorithm 2 Distogram Loss

def DistogramLoss($\{C\}, \{\hat{C}\}$) :

Compute distance between all atoms

1: $z_{i,j} = dist(C)$

2: $\hat{z}_{i,j} = dist(\hat{C})$

3: $error = \frac{1}{n}\sum_{i,j}(z_{i,j} - \hat{z}_{i,j})^2$

4: **return** $error$

Optimal Transport Loss

The optimal transport loss ($\mathcal{L}_{\text{OT}}$) is the same as in GeoMol.[24] The optimal transport loss enables a comparison between a mismatched number of predicted

and ground truth conformers by finding the optimal correspondence between each of the generated and ground truth conformers.

Let G and T be the number of generated and ground truth conformers, respectively. The optimal transport loss is defined as:

$$\mathcal{L}_{OT} := \min_{P \in Q_{G,T}} \sum_{g,t} P_{g,t} \mathcal{L}(C, \hat{C}) \tag{7}$$

where $\mathcal{L}$ can be any loss function comparing two conformers and the transport plan P satisfies:

$$Q_{G,T} := \{P \in \mathbb{R}^{G \times T} : P 1_T = \frac{1}{G} 1_G, P^P 1_G = \frac{1}{T} 1_T\}$$

where G and T are the number of generated and true conformers, respectively.[24]

Auxiliary Losses

An additional auxiliary loss is calculated for both the positional and optimal transport loss. The auxiliary loss is calculated for each structure generated in the structure module stack. Rather than backpropagating after each of the auxiliary losses is calculated, they are instead stored and contribute to the final backpropagation step after the training batch is pushed through the entire model architecture.[3]

3.3.5 Hyperparameters

A cyclic learning rate scheduler was used, with an initial learning rate of $4e-5$ and a maximum learning rate of 0.001. The training was done over 500 epochs, ending up with a total run time of 74 hours. The training set consisted of 10,000 small molecules randomly selected from GEOM-QM9. The validation set and testing set consisted of 1,000 small molecules randomly chosen from GEOM-QM9. The model weights with the lowest validation loss were used on the testing set. Models were run using a batch size of 12 across 2 GPUs (total batch size of 24). All code was written using the PyTorch package.[36]

3.4 Experimental Model Architectures

The following sections describe the various modifications made to the baseline model.

3.4.1 3D Evoformer

The 3D Evoformer described here is intended to replace the triangular multiplicative and attention updates used in the AlphaFold2 Evoformer.[3] The 3D Evoformer is inspired by a similar module in RosettaFold2.[31] In the 3D Evoformer, after updating the sequence representation, we use BackBoneUpdate from AlphaFold2 to project the linear representation into three dimensions. This provides an updated 3D molecular structure. Then, a radial basis function is applied to the 3D representation to obtain a pair representation (this replaces the previously used outer product calculation). Next, the pair representation is updated using axial (column- and row-wise) self-attention with bias from the pair representation. Since the pair representation is initialized from the 3D structure, we know it must satisfy the triangle inequality. Thus, the bias term enables a soft constraint similar to what the triangular updates it is replacing but without the added computational cost. The full algorithm is provided below (Alg 3).

Algorithm 3 3D Evoformer

function 3DEvoformer($\{s_i\}, \{z_{ij}\}, \{T_i\}$)
    s_i+ = SequenceAttentionWithPairBias($s_i, z_{ij},$)
    s_i+ = SequenceTransition(s_i)
    $T_i \leftarrow T_i \circ$ BackboneUpdate(s_i)
    z_{ij} = rbf(T_i)
    z_{ij}+ = RowWiseAttentionWithPairBias(z_{ij}, z_{ij})
    z_{ij}+ = ColumnWiseAttentionWithPairBias(z_{ij}, z_{ij}))
    z_{ij}+ = PairTransition(z_{ij}))
    return $\tilde{s}_i$
end function

3.4.2 Modified Loss Functions

Kabsch Algorithm

The Kabach algorithm, introduced in 1976 by Wolfgang Kabsch, finds the optimal alignment between two sets of coordinates in 3D space.[37] The algorithm finds the optimal rotation vector through the singular value decomposition of the covariance matrix between the two sets of coordinates to be aligned (Alg 4). By aligning the generated and true conformers, the loss function becomes invariant to global rotations and translations. After calculating the optimal alignment, the positional loss from the baseline model is used (Alg 1). The Kabsch algorithm is also used in baseline model during testing to calculate COV and AMR.

Algorithm 4 Kabsch Algorithm for Conformer Adjustment

def KabschRealignment($\{C\}, \{\hat{C}\}$) :

Center points

1: $\overline{C} = \frac{1}{n}\sum_{i=1}^{n} C_i$, $\overline{\hat{C}} = \frac{1}{n}\sum_{i=1}^{n} \hat{C}_i$

2: $\tilde{C}_i = C_i - \overline{C}$, $\tilde{\hat{C}}_i = \hat{C}_i - \overline{\hat{C}}$, for $i = 1, 2, \ldots, n$

Compute then decompose covariance matrix

3: $H = \sum_{i=1}^{n} \tilde{C}_i \tilde{\hat{C}}_i^T$

4: $H = USV^T$

Compute and apply rotation matrix

5: $R = VU^T$

6: $\hat{C}_{rot} = R \cdot \hat{C}$

7: **return** $\hat{C}_{rot}$

Triangle-aligned point error Loss

The triangle-aligned point error (TAPE) loss is a novel loss function that provides a local alignment between two conformers. Rather than finding the optimal rotation vector between two 3D points clouds, TAPE loss iteratively aligns three-atom triangles throughout the molecule (Alg 5).

The first step of TAPE is identifying a set of non-overlapping triangles in the molecule. This is done by first constructing a list of all non-overlapping triangles

in a molecule (triangles are defined as 3-atom sets where all atoms are directly connected to at least one other atom in the set). From this set of all triangles, a non-overlapping set of triangles (a set of triangles where no atom is present in more than one triangle) is constructed. We then iterate over the non-overlapping set of triangles, aligning each triangle in both the ground truth and generated conformer on the XY plane. The alignment is done in three steps (Alg 6). In the first step, both triangles are centered at the origin and a vertex is projected along the X-axis. Then, the rotation matrix applied implicitly during the projection is determined and applied to all other atoms in the molecule (Alg 7 and Alg 9). To complete the alignment, a rotation matrix is calculated to flatten the triangle on the XY plane. This is done by determining the rotation needed to rotate the side opposite the projected vertex such that it lies on the XY plane (Alg 8). As written, the method does not guarantee that the vertices opposite the projected vertex fall on the right sides of the y-axis. We correct this by reflecting the generated conformer over the XY plane if the vertices opposite the projected vertex are not properly aligned. Once aligned, the average distance between all points not in the aligned triangle in the generated and true conformers is calculated.

This three-step procedure is repeated for each of the triangle's vertices so the result is not biased toward any point on the triangle. The loss for each triangle is then clamped and divided by the clamp value so each triangle contributes to the loss at a value between 0 and 1. The final TAPE loss is the sum of the loss for each triangle divided by the total number of triangles. Due to the process of alignment, the TAPE loss is invariant to global translations and rotations.

Algorithm 5 TAPE Loss

def TAPE($\{C\}, \{\hat{C}\}, \epsilon, d_{\text{clamp}}, Z, \{triangles\}$) :

Iterate through non-overlapping one-hop triangles in the molecule

for $t \in triangles$ **do**

Shift all points by center of the triangle

$m_t = \frac{1}{3}\sum_{i \in t} C_i$

$\hat{m}_t = \frac{1}{3}\sum_{i \in t} \hat{C}_i$

$C_m = C_i - m_t$, for $i = 1, 2, ..., n$

$\hat{C}_m = \hat{C}_i - \hat{m}_t$, for $i = 1, 2, ..., n$

for $v \in t$ **do**

Align triangles at center and one vertex

$C_R = \text{AlignTriangle}(C, v)$

$\hat{C}_R = \text{AlignTriangle}(\hat{C}, v)$

$error_{\text{tri}}$ += $\text{RMSD}(C_R, \hat{C}_R, \epsilon)$

end for

$error_{\text{summed}}$ += $\frac{1}{Z}\text{clamp}(error_{\text{tri}}, d_{\text{clamp}})$

end for

$error_{\text{final}} = error_{\text{summed}}/n_{\text{triangles}}$

return $error_{\text{final}}$

Algorithm 6 Align Triangle

def AlignTriangle(C, t) :

Iterate through each vertex of triangle

for $v \in t$ **do**

Project vertex onto x-axis

$p_v = ||v||^2 * (1, 0, 0)$

Rotate molecule so triangle lies on xy-plane

$R_1 = \text{RotateToX}(v, p_v)$

$R_2 = \text{RotateToXY}(t, v, R_1)$

$C_R = R_2 R_1 C$

end for

return C_R

Algorithm 7 Get First Rotation

def GetFirstRotation(C, t) :

1: $\vec{v} = \vec{v_1} \times \vec{v_2}$

2: $\theta = arccos(\vec{v_1} \cdot \vec{v_2})$

3: **return** $GetRotationMatrix(\vec{v}, \theta)$

Algorithm 8 Get Second Rotation

def GetSecondRotation(v, p) :

1: $\vec{v} = \vec{v_2} - \vec{v_3}$

2: $A = v \cdot e_3$

3: $B = (e_1 \times \vec{v}) \cdot e_3$

4: $C = (\vec{v} \cdot e_1)(e_3 \cdot e_1)$

5: $x = \frac{C(C-A)-\sqrt{B^2[(A-C)^2+B^2-C^2]}}{(A-C)^2+B^2}$

6: $y = \frac{(C-A)x}{B}$

7: $\theta = arctan(\frac{y}{x})$

8: **return** $GetRotationMatrix(\vec{v}, \theta)$

Algorithm 9 Get Rotation Matrix

def GetRotationMatrix(v, θ) :

1: $R = \begin{bmatrix} cos\theta + x^2(1-cos\theta) & xy-(1-cos\theta)-zsin\theta & xz(1-cos\theta)+ysin\theta \\ yx(1-cos\theta)+zsin\theta & cos\theta + y^2(1-cos\theta) & yz(1-cos\theta)-xsin\theta \\ zx(1-cos\theta)-ysin\theta & zy(1-cos\theta)+xsin\theta & cos\theta + z^2(1-cos\theta) \end{bmatrix}$

2: **return** R

Testing TAPE loss convergence

To determine if the TAPE loss properly converges, coordinates were initialized either in a black hole (normal distribution with mean 0 and standard deviation 10^{-5}) or at random (normal distribution with mean 0 and standard deviation 1). The loss between the initialized coordinataes and the true structure is then calculated and backpropagated to find the gradient. Then, the adam optimizer is used to update the coordinates.[38] This is repeated 300 times for each molecule. The Kabsch loss is simultaneously calculated at each step. For this experiment, the following 10

molecules were used to test TAPE's convergence, selected for their diversity in size and structure:
O-C-C-C-C-N,
COCC(CO)CO,
CCCCCC[n+]1cc[nH]c1,
CN1C(=O)CN=C(C2=C1C=CC(=C2)Cl)C3=CC=CC=C3,
CC(C)CN1C(=O)COC2=C1C=CC(=C2)NC(=O)/C=C/C3=CC=CC=C3,
C1CC2=C3C(=CC=C2)C(=CN3C1)[C@H]4[C@@H]
(C(=O)NC4=O)C5=CNC6=CC=CC=C65,
O=C1C=CN(C(=O)N1)C5OC(CO)C(OP(=O)(O)OP(=O)
(O)OCC4OC(n3cnc2c(ncnc23)N)C(O)C4O)C5,
Cc1onc(-c2ccccc2)c1C(=O)N[C@@H]1C(=O)N2[C@@H](C(=O)O)C(C)
(C)S[C@H]12, CC1(C)SC2C(NC(=O)Cc3ccccc3)C(=O)N2C1C(=O)O.[Na],
C/C=C/CC(C)C(O)C1C(=O)NC(CC)C(=O)N(C)CC(=O)N(C)C(CC(C)C)
C(=O)NC(C(C)C)C(=O)N(C)C(CC(C)C)C(=O)NC(C)C(=O)NC(C)
C(=O)N(C)C(CC(C)C)C(=O)N(C)C(CC(C)C)C(=O)N(C)C(C(C)C)C(=O)N1C.

4
Results

Contents

4.1 Baseline **29**
4.2 Kabsch loss **31**
4.3 3D Evoformer **38**
4.4 TAPE loss **39**

4.1 Baseline

The baseline OpenMol model (outline in detail in Chapter 3: Methods) achieved a mean (median) COV-R of 51.58% (50.0%), COV-P of 79.13% (100.0%), AMR-R of 0.51Å (0.48Å), and AMR-P of 0.33Å (0.26Å). To better understand these results, we explored the association between the four evaluation metrics and either the number of rotatable bonds or the total number of conformers being predicted (Fig. 2AB, Fig. 3AB, Fig. 4AB, Fig. 5AB). To assess the strength of the association between variables, we performed Spearman rank correlation testing, testing the hypothesis that performance monotonically weakened with increasing molecule complexity. We found a significant association (Spearman rank correlation coefficients, pval) between the number of rotatable bonds and COV-R (-0.470, <0.001) (Fig. 2A), AMR-R (0.501, <0.001) (Fig. 4A), and AMR-P (0.293, <0.001) (Fig. 4B).

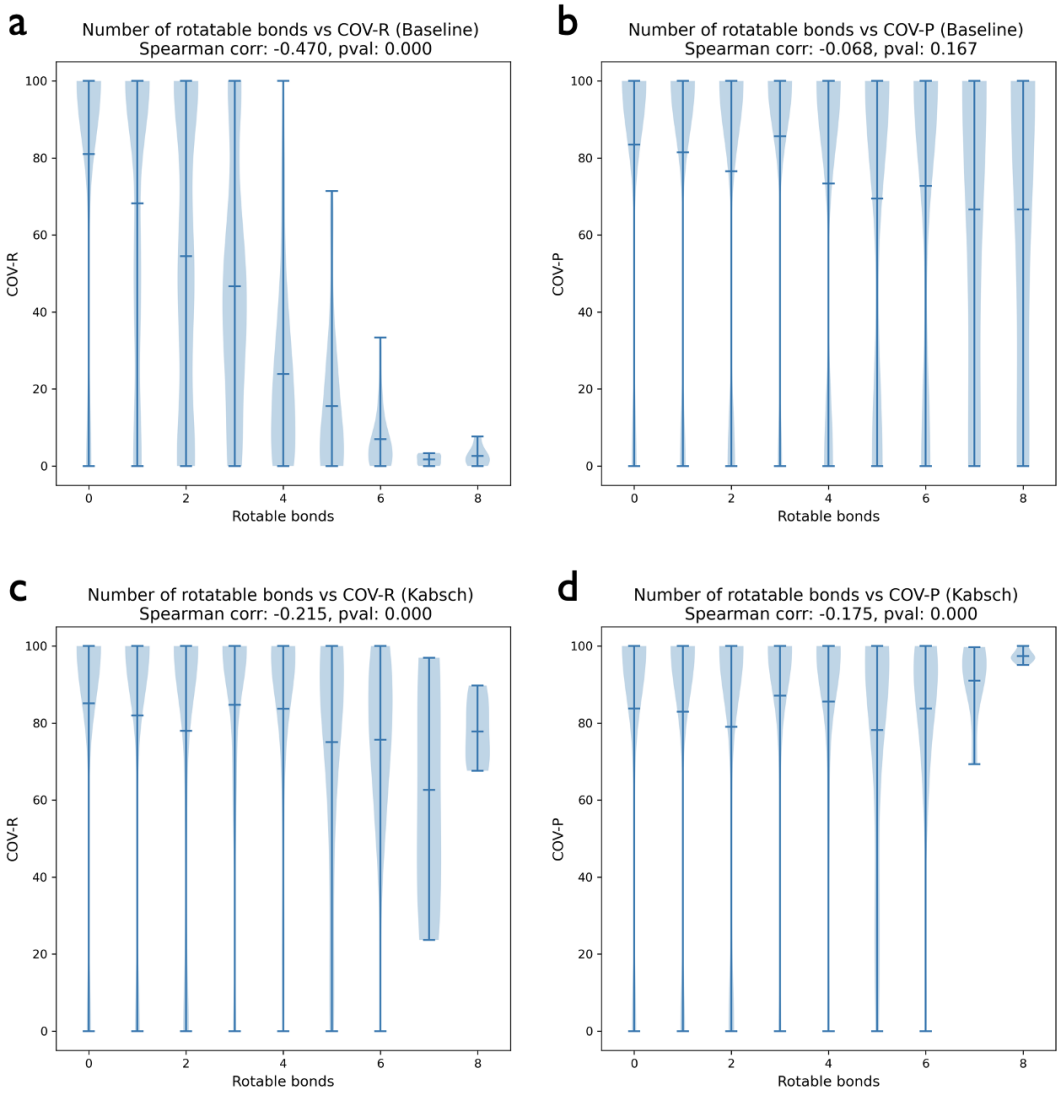


Figure 2: Associations between COV-R and COV-P and the number of rotatable bonds in each molecule for the baseline and Kabsch loss models. Associations between rotatable bonds and (a) COV-R or (b) COV-P in the baseline model. Association between rotatable bonds and (c) COV-R or (d) COV-P in the Kabsch loss model.

Associations between the number of ground truth conformations for each small molecule showed similar associations, with significant associations (Spearman rank correlation coefficients, pval) between the number of conformers and COV-R (-0.557, <0.001) (Fig. 3A), COV-P (-0.087, 0.009) (Fig. 3B), AMR-R (0.607, <0.001) (Fig. 5A), and AMR-P (0.370, <0.001) (Fig. 5B). Notably, we observe COV-P values exclusively of 0% and 100%, indicating that there may be a lack of structural diversity among the generated conformers.

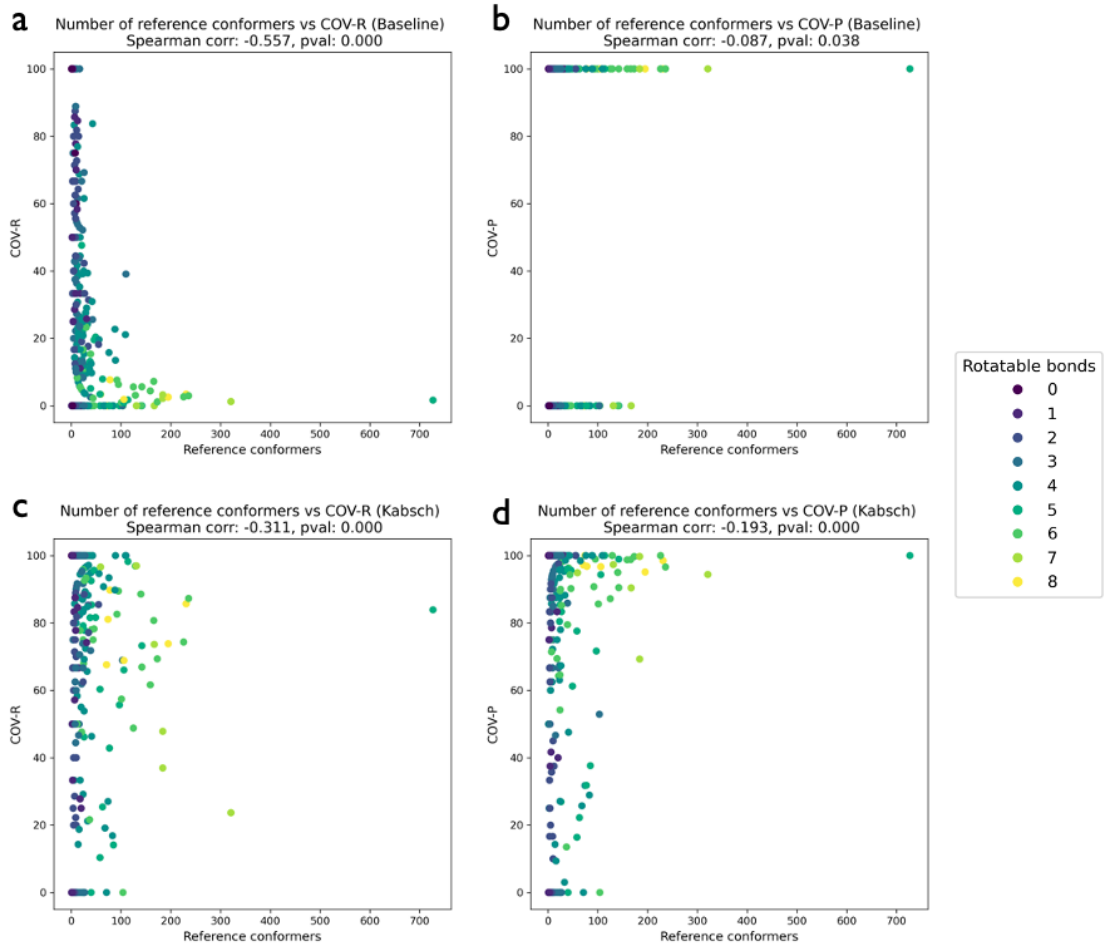


Figure 3: Associations between COV-R and COV-P in the baseline and Kabsch loss models and the number of reference conformers associated with each molecule in the testing set colored by the number of rotatable bonds. Associations between number of reference conformers and (a) COV-R or (b) COV-P in the baseline model. Association between the number of reference conformers and (c) COV-R or (d) COV-P in the Kabsch loss model.

4.2 Kabsch loss

We next hypothesized that using a loss function that is invariant to translation and rotation would lead to improved performance. We chose to achieve this by using the Kabsch alignment algorithm (Chapter 3: Methods) to align generated and ground truth conformers before calculating the loss. We implemented the Kabsch alignment in all four components of the loss function where atom coordinates are compared ($\mathcal{L}_{\text{pos}}$, $\mathcal{L}_{\text{OT}}$, $\mathcal{L}_{\text{pos}_{\text{aux}}}$, $\mathcal{L}_{\text{OT}_{\text{aux}}}$). Using the Kabsch loss resulted in a mean (median) COV-R of 81.42% (100.0%) COV-P of 83.15% (100%), AMR-R of 0.302Å

(0.23Å), AMR-P of 0.294Å (0.209Å). Despite the significant improvement above the baseline model, these results significantly fall behind those of nearly all other models except in terms of Mean COV-P for OMEGA, median AMR-P for GeoMol, and outperforms GeoDiff on all metrics except mean and median AMR-R. Otherwise, the median COV-R and COV-P is matched by OpenMol across all the models (Table 2).

Models	Metrics							
	COV-R [%] ↑		COV-P [%] ↑		AMR-R [Å] ↓		AMR-P [Å] ↓	
	Mean	Med	Mean	Med	Mean	Med	Mean	Med
RDKit	85.1	**100**	86.8	**100**	0.235	0.199	0.232	0.205
OMEGA	85.5	**100**	82.9	**100**	**0.177**	**0.126**	0.224	**0.186**
GeoMol	91.5	**100**	86.7	**100**	0.225	0.193	0.270	0.241
GeoDiff	76.5	**100**	50.0	**100**	0.297	0.229	0.524	0.510
Torsional diffusion	**92.8**	**100**	**92.7**	**100**	0.178	0.147	**0.221**	0.195
OpenMol	81.4	**100**	83.1	**100**	0.302	0.230	0.294	0.209

Table 2: Comparison of results from OpenMol with other recent models. The best results for each metric are bolded. Results for other models are from the Torsional Diffusion paper, which used the same testing set. [27]

We again assessed how these evaluation metrics were associated with the number of rotatable bonds in each small molecule. We found significant associations (Spearman rank correlation coefficients, pval) between the number of rotatable bonds and COV-R (-0.215, <0.001) (Fig. 2C), COV-P (-0.175, <0.001) (Fig. 2D), AMR-R (0.363, <0.001) (Fig. 5C), and AMR-P (0.349, <0.001) (Fig. 5D).

We also measured the associations between the four evaluation metrics and the number of ground truth conformers for each molecule. We found significant correlations (Spearman rank correlation coefficients, pval) between the number of reference conformers and COV-R (-0.311, <0.001) (Fig. 2C), COV-P (-0.193, <0.001) (Fig. 2D), AMR-R (0.492, <0.001) (Fig. 4C), AMR-P (0.431, <0.001) (Fig. 4D).

We hypothesized that the clustering of COV-P at 0% and 100% for both the baseline and Kabsch model implied a lack of diversity among the generated conformers. To test this hypothesis, we generated two metrics of conformer diversity. The first is the average RMSD between all conformers for a given molecule (intra-molecular average RMSD). Using a paired t-test we find a significant ($p<0.001$) difference between the mean intra-molecular average RMSD. Taking the difference between the intra-molecular average RMSD between all ground truth conformers and generated conformers, we find that the generated conformers have a mean (median) intra-molecular average RMSD that is 0.193Å (0.189Å) lower (Fig. 6A). Since intra-molecular RMSD could be biased by the total number of conformers generated during testing, we developed a second metric termed intra-molecular max RMSD. Intra-molecular max RMSD finds the maximum RMSD among all pairwise RMSD comparisons within a set of conformers for a given molecule. Again, we find that the generated conformers are significantly less diverse than the ground truth conformers, with a mean (median) intra-molecular max RMSD that is 0.149Å (0.123Å) lower. Again, a paired t-test comparing the distribution between true and generated intra-molecular max RMSD values showed a significant decrease in the generated conformer diversity ($p<0.001$) (Fig. 6B).

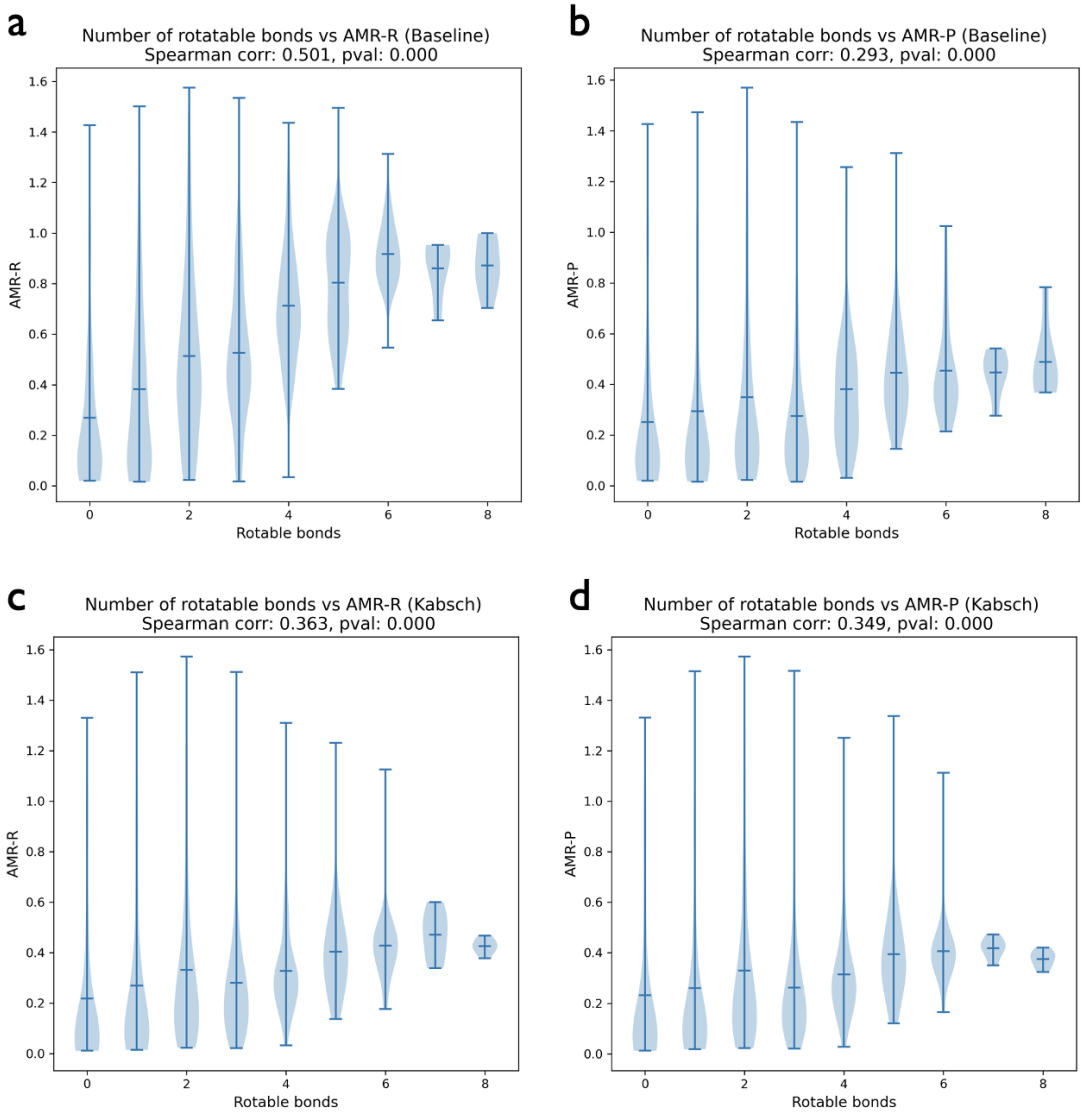


Figure 4: Associations between AMR-R and AMR-P in the baseline and Kabsch loss models and the number of reference conformers associated with each molecule in the testing set colored by the number of rotatable bonds. Associations between number of reference conformers and (a) AMR-R or (b) AMR-P in the baseline model. Association between the number of reference conformers and (c) AMR-R or (d) AMR-P in the Kabsch loss model.

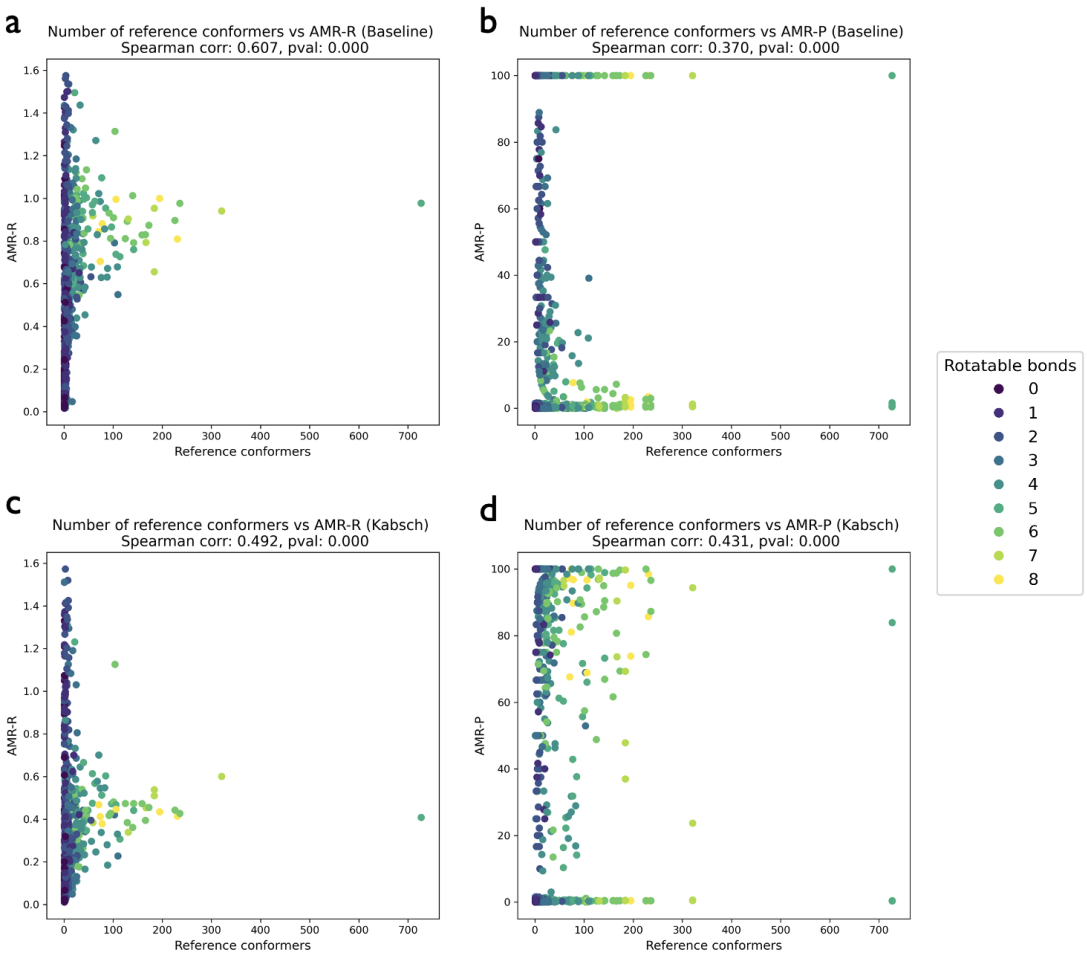


Figure 5: Associations between AMR-R and AMR-P in the baseline and Kabsch loss models and the number of reference conformers associated with each molecule in the testing set colored by the number of rotatable bonds. Associations between number of reference conformers and (a) AMR-R or (b) AMR-P in the baseline model. Association between the number of reference conformers and (c) AMR-R or (d) AMR-P in the Kabsch loss model.

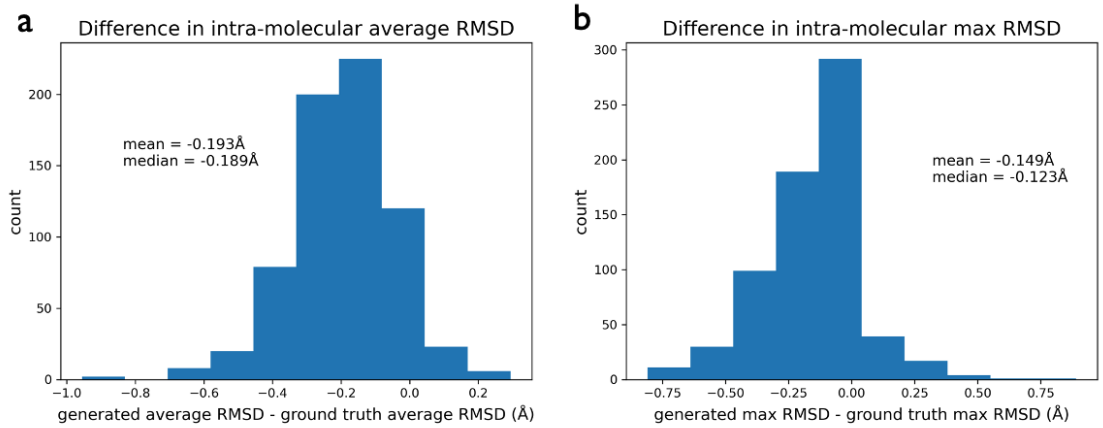


Figure 6: Histogram of difference between each molecule's generated and ground truth (a) intra-molecuar average RMSD and (b) intra-molecular max RMSD.

To better understand the weaknesses of the model, we explored the cluster of molecules that had a COV-R and COV-P of 0 (Fig 3CD). Manual inspection of molecules in this category showed that nearly all instances were a result of improper chirality assignments. Chirality assignment issues were either diastereomers (non-superimposable, non-mirror images, Fig. 7A) or enantiomers (non-superimposable, mirror images, Fig. 7BC). Small molecule chirality is specified via the ordering of the atoms in the input, meaning that diastereomers and enantiomers should be considered failures of the model.

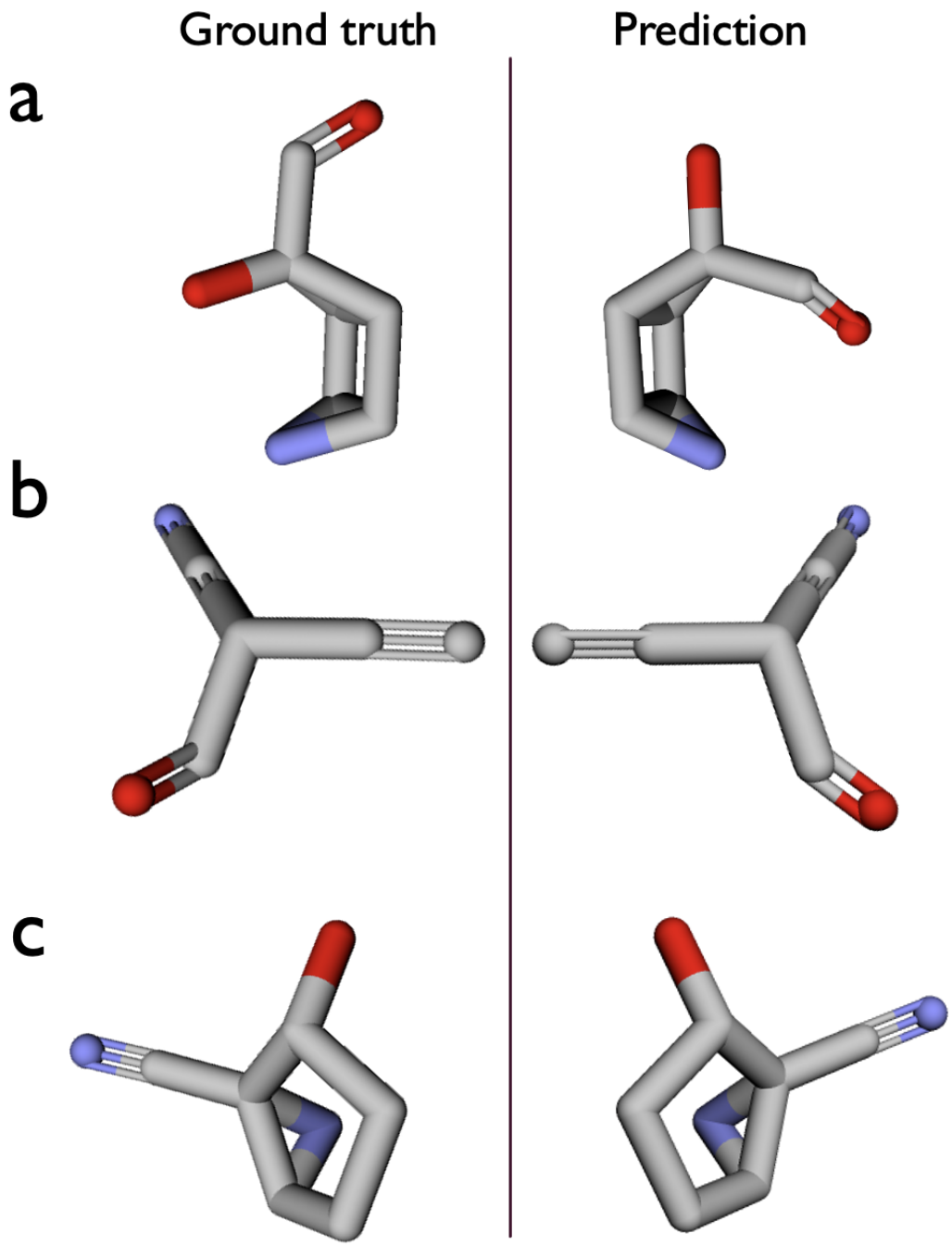


Figure 7: Representative examples of small molecules with COV-P and COV-R values of 0. (a) Example of diastereomer and (b,c) examples of enantiomers.

To more systematically evaluate the outputs of OpenMol, we analyzed all the 24487 generated conformers for violations of physical constraints using PoseBusters, which checks an input list of conformers against a set of physical constraints.[39] For each conducted test, PoseBusters determines whether a molecules counts as a "pass" or "fail" depending on whether or not the specified physical constraint was satisfied. Using PoseBusters for the output of the Kabsch model we obtained the following results: atoms connected (24487 pass, 0 fail), bond lengths within bounds (14359 pass, 10128 fail), bond angles within bounds (24098 pass, 389 fail), no internal clashes (24469 pass, 18 fail), aromatic ring flatness (24487 pass, 0 fail), double bond flatness (24480 pass, 7 fail), energy ratio (21745 pass, 2742 fail).

4.3 3D Evoformer

Modeled after AlphaFold2, the baseline model OpenMol model's Evoformer utilizes two representations of the input molecule: a 1D sequence representation and a 2D pair representation. Within the Evoformer, the sequence and pair representations are used to update each other, and triangular operations (Chapter 3: Methods) are used to help ensure the triangle inequality is satisfied. We hypothesized that using a 3D representation of the molecule in the Evoformer to generate the pair representation could improve the model's runtime (a detailed overview of the 3D Evoformer is provided in Chapter 3: Methods). This idea is similar to what was implemented in RosettaFold2, a protein folding model heavily inspired by AlphaFold2 which utilizes 1D, 2D, and 3D representations in the Evoformer [31].

The 3D Evoformer, using the Kabsch loss, resulted in diminished performance performance relative to the baseline model but with significant improvements in memory usage and runtime. The 3D Evoformer obtained a mean (median) COV-R of 79.33% (100.0%), COV-P of 80.98% (100.0%), AMR-R of 0.33Å (0.26Å), and AMR-P of 0.33Å (0.25Å). We found that average GPU memory utilization was reduced by 19.13%, peak GPU memory utilization was reduced by 21.35%, and total runtime was reduced by 14.56%. Ablation experiments showed performance

substantially deteriorated when no bias term from the pair representation was included in the attention scores.

4.4 TAPE loss

Upon observing significant gains in performance when implementing the Kabsch loss, we hypothesized that a loss utilizing local alignments could lead to further improvements in performance. Inspired by AlphaFold2's FAPE loss, we designed the triangle-aligned point error (TAPE) loss (detailed outline and algorithm provided in Chapter 3: Methods). AlphaFold2's loss function, FAPE, relies on frames, which are translation and rotation vectors relative to the origin assigned to each amino acid. This can be done systematically because each non-terminal amino acid always has two bonds to the neighboring amino acid, making assigning a plane trivial. Because in a small molecule the number of bonds a single molecule shares with surrounding molecules ranges from 1 to 4, there is no fixed method that can be used to assign frames to each atom. Thus, TAPE works by assigning non-overlapping sets of triangles to a small molecule (Fig. 8A) which are then iteratively aligned (Fig. 8B).

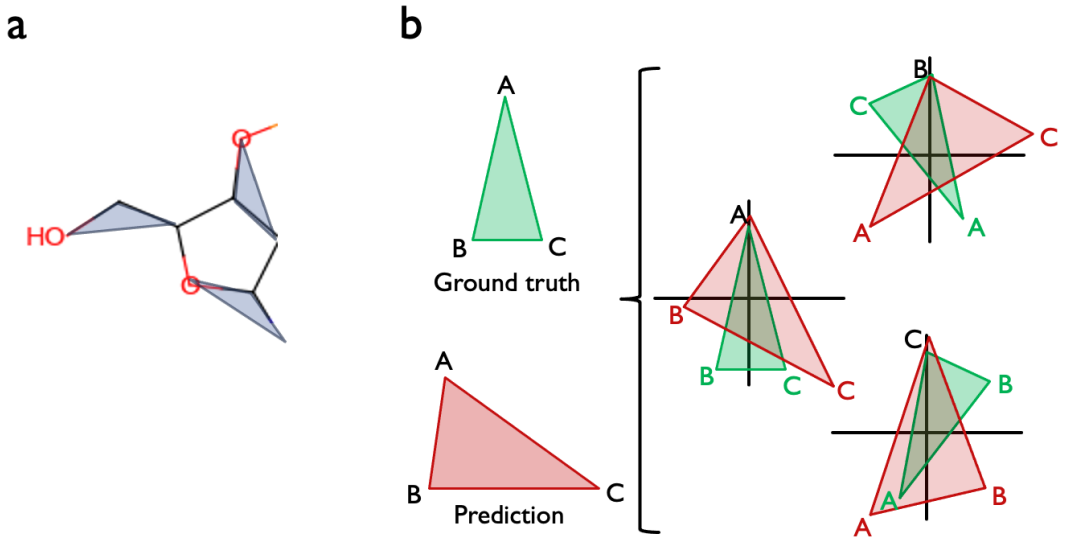


Figure 8: Visualization of how the TAPE alignment steps look. (a) Image demonstrating how non-overlapping triangles are assigned to an arbitrary molecule. (b) Diagram of how alignment between two triangles takes place after they are both rotated onto the same plane.

To better understand the properties of TAPE, we constructed a series of synthetic tests. First, we explored the ability of the TAPE to converge from a black hole initialization to a given structure by following a TAPE loss. We conducted this test for a set of 10 highly diverse molecules, ranging in size from 6 to 85 non-hydrogen atoms. TAPE loss converged for each of these structures, and the minimum TAPE loss corresponded with the minimum Kabsch RMSD (Fig. 9A). We then repeated the same test, but with random initialization around the origin (Fig. 9B). From a random initialization, we saw a similar pattern of behavior, although TAPE appeared to converge more quickly from the random initialization.

We next looked at how TAPE and Kabsch responded to noise injections. We took a molecule and iteratively added random noise to each with a mean of 1/100Å. This was done a total of 100 times (Fig. 9C). We see a deviation in behaviour between TAPE and Kabsch, with Kabsch have a purely linear relationship between injected noise and error. On the other hand, TAPE begins to plateau. An example of one of the conformers that was converged from noise to a proper structure following the TAPE loss is shown (Fig. 9D).

We then tested TAPE in a scaled-down experiment where only 1,000 molecules were used in the training set and only one conformer was predicted for each molecule. On the testing set, the Kabsch loss obtained a mean RMSD of 0.514Å, a median of 0.438Å, and a minimum of 0.037Å. TAPE loss obtained a mean RMSD of 0.559Å, a median of 0.469Å, and a minimum of 0.071Å. TAPE's run time was 9.12 times slower than that of Kabsch, owing to the large number of additional calculations that need to be made to calculate the loss.

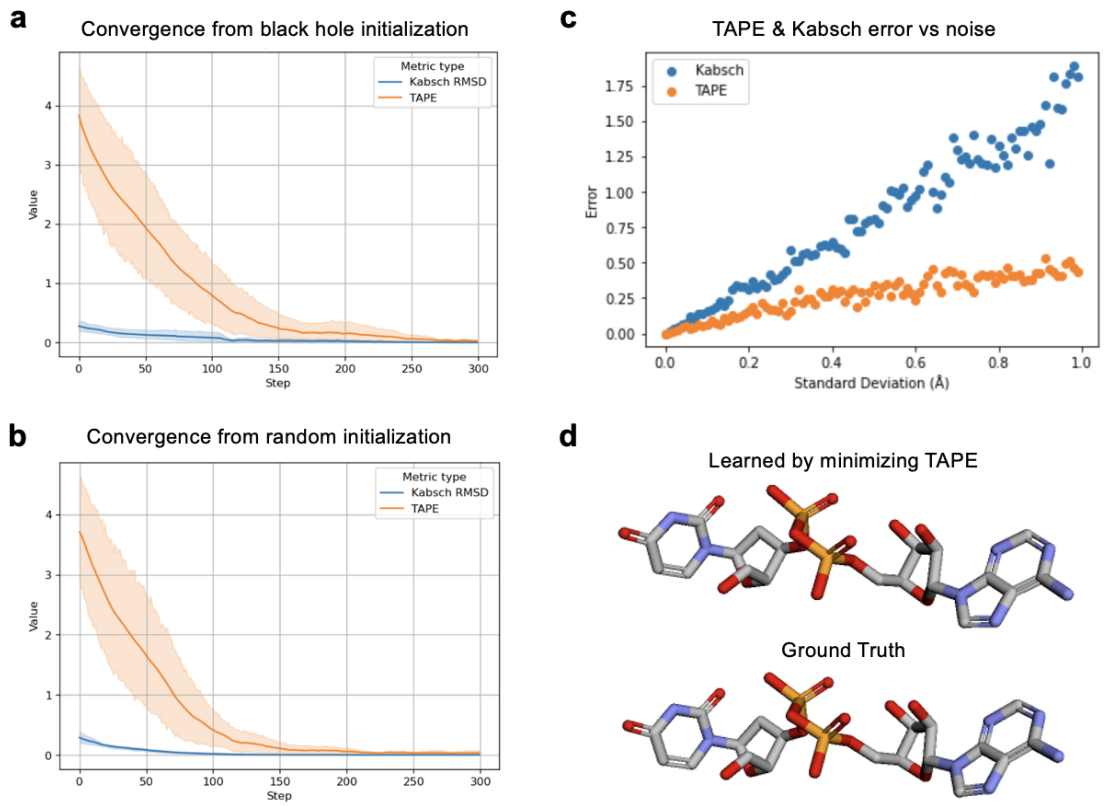


Figure 9: Figures summarizing the series of tests performed to characterize the TAPE loss. (a) Average conversion of 10 highly varied structures from black hole initialization. (b) Average conversion of 10 highly varied structures from random initialization. (c) Error vs injected noise standard deviation plot for TAPE and Kabsch losses. (d) Example of one of the 10 learned small molecules at the 300th step of the conversion tests.

5 Discussion

Here we developed OpenMol, a deep learning model for MCG heavily inspired by AlphaFold2. By doing so, we showed that the AlphaFold2 architecture can be repurposed to the task of MCG. Our final model obtained a mean COV-R and COV-P of 81.4% and 83.1%, respectively. Although falling behind the state-of-the-art MCG models, OpenMol outperformed GeoDiff on both COV-R and COV-P and outperformed OMEGA on COV-P (Table 2). Notably, OpenMol's mean (median) AMR-R of 0.302Å (0.230Å) was lower than all other models used for comparison, demonstrating a minor discrepancy between AMR and COV. However, because the threshold used for COV is strict (0.5Å RMSD), we believe COV is a more important measure of model performance on the MCG task. This is because further reductions in RMSD that already fall below the threshold are unlikely to be as valuable as predicting a greater variety of potential conformers. The importance of COV is supported by the discrepancy between COV and AMR in Torsional Diffusion, which obtained a mean COV-R 7.3% higher than OMEGA despite having a lower mean AMR-R. Such variations in performance are likely explained by Torsional Diffusion being able to generate more diverse sets of conformers than OMEGA. Analysis of the diversity of structures generated by OpenMol showed that OpenMol generated sets of conformers with average intra-molecular average RMSD and max RMSD values of 0.193Å and 0.149Å below the ground truth sets

of conformers, respectively. Thus, we concluded that OpenMol does not generate as diverse of conformer sets as is observed in nature. However, because these intra-molecular average RMSD and max RMSD metrics fall below less than half of the COV threshold value, we believe conformer diversity is not the major limitation of the network as these values are relatively small.

Instead, we believe OpenMol's major limitation was in its inability to accurately generate conformers with proper chirality. Upon manual inspection, we observed that nearly all examples of molecules where COV-R and P had values of 0, which was the case for 11.3% of all molecules in the testing set, had proper structures but incorrect chirality. To demonstrate the magnitude of the effect these instances have on the model performance, we removed these improperly predicted molecules and obtained COV-R and COV-P values of 91.77% and 93.72%, respectively. Comparing against current state-of-the-art methods, these results would place OpenMol's mean COV-R as the second highest, falling 1.03% behind Torsional Diffusion. Meanwhile, a COV-P of 93.72% would be the new state-of-the-art. While the removal of molecules with improperly predicted chirality is an oversimplification of the problem, it highlights the importance of improving OpenMol's ability to accurately predict chirality. GeoMol avoided this problem by simply reflecting generated molecules in the testing set that had improperly generated chirality.[24]. We chose not to pursue this solution to the problem because our aim was to develop a model that retains the key properties of AlphaFold2 but in the MCG setting. AlphaFold2 automatically handles chirality and so we wanted to develop a model that does the same.[3]. Chirality is also handled automatically in Torsional Diffusion, the current state-of-the-art method for MCG.[27]

In developing OpenMol, we found that replacing a simple point-wise loss with an invariant, alignment-based loss (the Kabsch loss) resulted in a 29.8% gain in mean COV-R and 3.97% gain in mean COV-P. The magnitude of this improvement led us to develop a novel loss function specifically designed for the task of MCG. Inspired by AlphaFold2's FAPE, we developed TAPE. The introduction of the FAPE loss in AlphaFold2 was a key component that enable the significant advance in protein

structure prediction accuracy observed in AlphaFold2.[3, 31] In AlphaFold2, FAPE relies on the use of rotations and translations associated with "frames" throughout a polypeptide sequence. However, this is not possible in a small molecule setting because of the more varied patterns of atom connectivity. In TAPE, we solved this problem by defining a molecule through a set of triangles, each of which then defines a plane with an associated rotation relative to the origin. The main advantage of the TAPE loss is that it enables local alignment between conformers, so structures can be learned iterative during training. Additionally, TAPE retains Kabsch's properties of invariance to global rotations and translations. We also found that TAPE has a less linear relationship with noise than Kabsch, a potentially advantageous property for training. However, in our testing of the TAPE loss for MCG, we found it to be 9 times slower and substantially less stable than Kabsch. Additionally, we observed slightly diminished performance on the single conformer generation task, with TAPE obtaining an average RMSD of 0.045Å lower than Kabsch.

Despite these results, the TAPE loss may still be a promising loss function for MCG. Using TAPE we still obtained a very similar final testing accuracy to Kabsch, showing that TAPE can be used to train a model for MCG tasks. An improved understanding of the potential benefits of TAPE will require testing on larger molecules as well as on MCG tasks involving multiple conformers per molecule so that COV-R and COV-P can be determined. Further, improvements to the TAPE algorithm itself can be made. For example, we could align one molecule onto the other, rather than aligning both molecules onto the XY plane. This would reduce the computational time by roughly 2-fold. Another set of future directions is to test the simultaneous use of both the TAPE and Kabsch losses, balancing the sensitivity of TAPE with the stability and speed of Kabsch.

Finally, our testing of a modified 3D Evoformer resulted in significant improvements in speed and memory usage at the cost of performance. These results were surprising given the lack of harm observed by similar changes in RosettaFold2.[31] However, our diminished performance may be a result of insufficient hyperparameter tuning. We found these initial results promising because the training and validation

loss patterns appeared to reach earlier plateaus in manner that was previously solved by hyperparameter tuning of the baseline architecture. Future directions will focus on performing more exhaustive testing of the 3D Evoformer architecture hyperparameters.

OpenMol's failure to achieve state-of-the-art results is in part due to resource limitations. Due to the cost and time associated with running experiments, extensive hyperparameter tuning was not performed. Additionally, we used a limited training set size of 10,000 molecules, while all current state-of-the-art methods used for comparison used the entire GEOM-QM-9 training set of 106,586 molecules. Furthermore, we observed that increasing the training set size from 1,000 molecules to 10,000 molecules resulted in a roughly 3% gain in COV-R and COV-P. Thus, we anticipate an increase in training data set size will help improve OpenMol's results toward reaching state-of-the-art performance.

An additional limitation of OpenMol is that it did not use recycling. This is because experiments with recycling resulted in substantially diminished performance. Notably, we observed that recycling benefited single conformer generation and diminished performance on MCG. Thus, we believe that the failure of recycling is specific to the multi-conformer setting and future steps will work to address this limitation.

Apart from hyperparameter tuning, scaling up the network, optimizing TAPE, and introducing recycling, a critical next step is testing OpenMols on the GEOM-DRUGS dataset. The GEOM-DRUGS dataset is a much more realistic task setting for MCG in science and research, as the primary goal of MCG is for drug discovery and basic science. We did not test OpenMol on GEOM-DRUGS here because of resource constraints (GPU memory and availability).

Through OpenMol, we showed that AlphaFold2's architecture is generalizable to structure prediction tasks outside of proteins. Simultaneously, our results in this study provide future directions for exploring more applications of AlphaFold2. Here, we use a simple noise injection procedure during initialization to provide a diverse set of conformers. A similar approach could be taken to predict protein structure

ensembles. Additionally, using the AlphaFold2 architecture for MCG enables the possibility of co-inputting protein sequence information alongside a small molecule to generate structures of proteins bounds to small molecules.

6
Conclusion

Deep learning approaches to MCG have grown in popularity over experimental methods due to their substantial improvements in speed and cost. Here we built on recent deep learning advances by introducing OpenMol, an AlphaFold2-inspired architecture for MCG. We demonstrated that by modifying the input and loss function of AlphaFold2, we can build a model that achieves near-state-of-the-art performance on the GEOM-QM9 dataset. Future work will center on training OpenMol using the entire GEOM-QM9 dataset, testing OpenMol on larger molecules using the GEOM-DRUGS dataset, and addressing OpenMol's failure to consistently generate structures with proper chirality. We also found that replacing the triangular updates in OpenMol's baseline Evoformer with a 3D Evoformer utilizing axial attention led to significant improvements in memory usage and speed, but at the cost of diminished performance. Finally, we introduced a novel local alignment-based loss function for small molecules, which we termed the TAPE loss. Like the frames in AlphaFold2's FAPE, the assigned triangles in TAPE enable systematic ways to align local regions on small molecules. TAPE loss performed similarly to Kabsch on a single conformer generation task but was substantially slower and suffered from instability. More generally, we provide evidence that AlphaFold2 can be thought of as a general-purpose architecture for structure production tasks outside of the original protein structure prediction case for which it was designed.

References

[1] Paul C. D. Hawkins. "Conformation Generation: The State of the Art". In: *Journal of Chemical Information and Modeling* 57.8 (Aug. 28, 2017), pp. 1747–1756. URL: https://pubs.acs.org/doi/10.1021/acs.jcim.7b00221 (visited on 04/26/2023).

[2] Colin R. Groom and Jason C. Cole. "The Use of Small-Molecule Structures to Complement Protein–Ligand Crystal Structures in Drug Discovery". In: *Acta Crystallographica. Section D, Structural Biology* 73 (Pt 3 Feb. 22, 2017), pp. 240–245. pmid: 28291759. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5349436/ (visited on 04/26/2023).

[3] John Jumper et al. "Highly Accurate Protein Structure Prediction with AlphaFold". In: *Nature* 596.7873 (7873 Aug. 2021), pp. 583–589. URL: https://www.nature.com/articles/s41586-021-03819-2 (visited on 08/17/2023).

[4] Gustaf Ahdritz et al. *OpenFold: Retraining AlphaFold2 Yields New Insights into Its Learning Mechanisms and Capacity for Generalization*. Aug. 12, 2023. URL: https://www.biorxiv.org/content/10.1101/2022.11.20.517210v3 (visited on 10/02/2023). preprint.

[5] Gengmo Zhou et al. *Uni-Mol: A Universal 3D Molecular Representation Learning Framework*. Mar. 6, 2023. URL: https://chemrxiv.org/engage/chemrxiv/article-details/6402990d37e01856dc1d1581 (visited on 08/17/2023). preprint.

[6] Jacob D. Durrant and J. Andrew McCammon. "Molecular Dynamics Simulations and Drug Discovery". In: *BMC Biology* 9.1 (Oct. 28, 2011), p. 71. URL: https://doi.org/10.1186/1741-7007-9-71 (visited on 04/28/2023).

[7] Stephen R. Wilson et al. "Applications of Simulated Annealing to the Conformational Analysis of Flexible Molecules". In: *Journal of Computational Chemistry* 12.3 (1991), pp. 342–349. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/jcc.540120307 (visited on 05/09/2023).

[8] Thomas A. Halgren. "Merck Molecular Force Field. I. Basis, Form, Scope, Parameterization, and Performance of MMFF94". In: *Journal of Computational Chemistry* 17.5-6 (1996), pp. 490–519. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/%28SICI%291096-987X%28199604%2917%3A5/6%3C490%3A%3AAID-JCC1%3E3.0.CO%3B2-P (visited on 05/13/2023).

[9] Olivier Sperandio et al. “MED-3DMC: A New Tool to Generate 3D Conformation Ensembles of Small Molecules with a Monte Carlo Sampling of the Conformational Space”. In: *European Journal of Medicinal Chemistry* 44.4 (Apr. 1, 2009), pp. 1405–1409. URL: https://www.sciencedirect.com/science/article/pii/S0223523408004558 (visited on 05/09/2023).

[10] T. Bohme Leite et al. “Frog: A FRee Online druG 3D Conformation Generator”. In: *Nucleic Acids Research* 35 (suppl_2 July 1, 2007), W568–W572. URL: https://doi.org/10.1093/nar/gkm289 (visited on 05/16/2023).

[11] Maria A. Miteva, Frederic Guyon, and Pierre Tuffi¿½ry. “Frog2: Efficient 3D Conformation Ensemble Generator for Small Compounds”. In: *Nucleic Acids Research* 38 (suppl_2 July 1, 2010), W622–W627. URL: https://doi.org/10.1093/nar/gkq325 (visited on 05/16/2023).

[12] Jean-Paul Ebejer, Garrett M. Morris, and Charlotte M. Deane. “Freely Available Conformer Generation Methods: How Good Are They?” In: *Journal of Chemical Information and Modeling* 52.5 (May 25, 2012), pp. 1146–1158. URL: https://doi.org/10.1021/ci2004658 (visited on 04/27/2023).

[13] Timothy F. Havel. “Distance Geometry: Theory, Algorithms, and Chemical Applications”. In: *Encyclopedia of Computational Chemistry*. Ed. by Paul Von Ragu Schleyer et al. Chichester, UK: John Wiley & Sons, Ltd, Apr. 15, 2002, cda018. URL: https://onlinelibrary.wiley.com/doi/10.1002/0470845015.cda018 (visited on 05/16/2023).

[14] David C. Spellmeyer et al. “Conformational Analysis Using Distance Geometry Methods”. In: *Journal of Molecular Graphics and Modelling* 15.1 (Feb. 1997), pp. 18–36. URL: https://linkinghub.elsevier.com/retrieve/pii/S1093326397000144 (visited on 05/16/2023).

[15] Sereina Riniker and Gregory A. Landrum. “Better Informed Distance Geometry: Using What We Know To Improve Conformation Generation”. In: *Journal of Chemical Information and Modeling* 55.12 (Dec. 28, 2015), pp. 2562–2574. URL: https://pubs.acs.org/doi/10.1021/acs.jcim.5b00654 (visited on 05/09/2023).

[16] Noel M. O’Boyle et al. “Confab - Systematic Generation of Diverse Low-Energy Conformers”. In: *Journal of Cheminformatics* 3.1 (Mar. 16, 2011), p. 8. URL: https://doi.org/10.1186/1758-2946-3-8 (visited on 05/16/2023).

[17] Paul C. D. Hawkins et al. “Conformer Generation with OMEGA: Algorithm and Validation Using High Quality Structures from the Protein Databank and Cambridge Structural Database”. In: *Journal of Chemical Information and Modeling* 50.4 (Apr. 26, 2010), pp. 572–584. URL: https://pubs.acs.org/doi/10.1021/ci100031x (visited on 05/13/2023).

[18] Paul C. D. Hawkins and Anthony Nicholls. “Conformer Generation with OMEGA: Learning from the Data Set and the Analysis of Failures”. In: *Journal of Chemical Information and Modeling* 52.11 (Nov. 26, 2012), pp. 2919–2936. URL: https://pubs.acs.org/doi/10.1021/ci300314k (visited on 05/16/2023).

[19] Nils-Ole Friedrich et al. "Benchmarking Commercial Conformer Ensemble Generators". In: *Journal of Chemical Information and Modeling* 57.11 (Nov. 27, 2017), pp. 2719–2728. URL: https://doi.org/10.1021/acs.jcim.7b00505 (visited on 05/16/2023).

[20] Lucian Chan, Geoffrey R. Hutchison, and Garrett M. Morris. "Bayesian Optimization for Conformer Generation". In: *Journal of Cheminformatics* 11.1 (May 21, 2019), p. 32. URL: https://doi.org/10.1186/s13321-019-0354-7 (visited on 04/27/2023).

[21] Lucian Chan, Geoffrey R. Hutchison, and Garrett M. Morris. "BOKEI: Bayesian Optimization Using Knowledge of Correlated Torsions and Expected Improvement for Conformer Generation". In: *Physical Chemistry Chemical Physics* 22.9 (2020), pp. 5211–5219. URL: https://pubs.rsc.org/en/content/articlelanding/2020/cp/c9cp06688h (visited on 04/27/2023).

[22] Gregor Simm and Jose Miguel Hernandez-Lobato. "A Generative Model for Molecular Distance Geometry". In: *Proceedings of the 37th International Conference on Machine Learning*. International Conference on Machine Learning. PMLR, Nov. 21, 2020, pp. 8949–8958. URL: https://proceedings.mlr.press/v119/simm20a.html (visited on 05/19/2023).

[23] Minkai Xu et al. *Learning Neural Generative Dynamics for Molecular Conformation Generation*. arXiv.org. Feb. 20, 2021. URL: https://arxiv.org/abs/2102.10240v3 (visited on 06/07/2023).

[24] Octavian-Eugen Ganea et al. *GeoMol: Torsional Geometric Generation of Molecular 3D Conformer Ensembles*. June 8, 2021. arXiv: 2106.07802 [physics]. URL: http://arxiv.org/abs/2106.07802 (visited on 11/29/2022). preprint.

[25] Ling Yang et al. *Diffusion Models: A Comprehensive Survey of Methods and Applications*. Mar. 23, 2023. arXiv: 2209.00796 [cs]. URL: http://arxiv.org/abs/2209.00796 (visited on 07/06/2023). preprint.

[26] Minkai Xu et al. *GeoDiff: A Geometric Diffusion Model for Molecular Conformation Generation*. Mar. 6, 2022. arXiv: 2203.02923 [cs, q-bio]. URL: http://arxiv.org/abs/2203.02923 (visited on 05/15/2023). preprint.

[27] Bowen Jing et al. *Torsional Diffusion for Molecular Conformer Generation*. Feb. 28, 2023. arXiv: 2206.01729 [physics, q-bio]. URL: http://arxiv.org/abs/2206.01729 (visited on 05/15/2023). preprint.

[28] Gengmo Zhou et al. *Do Deep Learning Methods Really Perform Better in Molecular Conformation Generation?* Mar. 27, 2023. arXiv: 2302.07061 [cs, q-bio]. URL: http://arxiv.org/abs/2302.07061 (visited on 05/11/2023). preprint.

[29] Zeming Lin et al. "Evolutionary-Scale Prediction of Atomic-Level Protein Structure with a Language Model". In: *Science* 379.6637 (Mar. 17, 2023), pp. 1123–1130. URL: https://www.science.org/doi/10.1126/science.ade2574 (visited on 08/18/2023).

[30] Ashish Vaswani et al. *Attention Is All You Need*. arXiv.org. June 12, 2017. URL: https://arxiv.org/abs/1706.03762v7 (visited on 08/18/2023).

[31] Minkyung Baek et al. *Efficient and Accurate Prediction of Protein Structure Using RoseTTAFold2*. May 25, 2023. URL: https://www.biorxiv.org/content/10.1101/2023.05.24.542179v1 (visited on 08/17/2023). preprint.

[32] Raghunathan Ramakrishnan et al. "Quantum Chemistry Structures and Properties of 134 Kilo Molecules". In: *Scientific Data* 1.1 (1 Aug. 5, 2014), p. 140022. URL: https://www.nature.com/articles/sdata201422 (visited on 05/18/2023).

[33] Christoph Bannwarth, Sebastian Ehlert, and Stefan Grimme. "GFN2-xTB—An Accurate and Broadly Parametrized Self-Consistent Tight-Binding Quantum Chemical Method with Multipole Electrostatics and Density-Dependent Dispersion Contributions". In: *Journal of Chemical Theory and Computation* 15.3 (Mar. 12, 2019), pp. 1652–1671. URL: https://doi.org/10.1021/acs.jctc.8b01176 (visited on 05/19/2023).

[34] Simon Axelrod and Rafael Gómez-Bombarelli. "GEOM, Energy-Annotated Molecular Conformations for Property Prediction and Molecular Generation". In: *Scientific Data* 9.1 (1 Apr. 21, 2022), p. 185. URL: https://www.nature.com/articles/s41597-022-01288-4 (visited on 05/19/2023).

[35] Philipp Pracht, Fabian Bohle, and Stefan Grimme. "Automated Exploration of the Low-Energy Chemical Space with Fast Quantum Chemical Methods". In: *Physical Chemistry Chemical Physics* 22.14 (2020), pp. 7169–7192. URL: https://pubs.rsc.org/en/content/articlelanding/2020/cp/c9cp06869d (visited on 05/19/2023).

[36] Adam Paszke et al. "PyTorch: An Imperative Style, High-Performance Deep Learning Library". In: *Advances in Neural Information Processing Systems*. Vol. 32. Curran Associates, Inc., 2019. URL: https://papers.nips.cc/paper_files/paper/2019/hash/bdbca288fee7f92f2bfa9f7012727740-Abstract.html (visited on 10/05/2023).

[37] W. Kabsch. "A Solution for the Best Rotation to Relate Two Sets of Vectors". In: *Acta Crystallographica Section A: Crystal Physics, Diffraction, Theoretical and General Crystallography* 32.5 (5 Sept. 1, 1976), pp. 922–923. URL: //scripts.iucr.org/cgi-bin/paper?a12999 (visited on 10/06/2023).

[38] Diederik P. Kingma and Jimmy Ba. *Adam: A Method for Stochastic Optimization*. Jan. 29, 2017. arXiv: 1412.6980 [cs]. URL: http://arxiv.org/abs/1412.6980 (visited on 10/06/2023). preprint.

[39] Martin Buttenschoen, Garrett M. Morris, and Charlotte M. Deane. *PoseBusters: AI-based Docking Methods Fail to Generate Physically Valid Poses or Generalise to Novel Sequences*. Aug. 10, 2023. arXiv: 2308.05777 [physics, q-bio]. URL: http://arxiv.org/abs/2308.05777 (visited on 09/27/2023). preprint.