Master 2 Internship Report

Victor GERTNER

Internship Subject: Creation of a 3D Molecule Generation Model Targeting a Protein of Interest

02/04/2024 to 30/09/2024

Company Internship Tutor: Vincent Bouttier / Hamza Tajmouati

School Internship Tutor: Sholom Schechtman

Contents

1 Introduction

Disclaimer: This paper was written with the assistance of ChatGPT, an AI language model, to enhance the clarity and readability of the report.

1.1 Iktos

Iktos is a Paris-based AI company founded in late 2016 by Quentin Perron, Nicolas Do Huu, and Yann Gaston-Mathé, with around 60 employees. The company specializes in applying artificial intelligence to chemistry, particularly in designing new drugs (small molecules, peptides) using deep generative models and data-driven retrosynthesis. Iktos has dedicated teams in medicinal chemistry, computational chemistry, AI and data science, engineering, and software development. With concrete real-life experience delivering value to drug discovery programs, the company has completed or is working on more than 50 projects.

This idea of applying artificial intelligence to chemistry came from a real issue: identifying a novel molecule that could become a new drug typically takes around 5 years and costs approximately 1200 million euros, with a success rate of only about 10%. This is why Iktos helps laboratories in their research efforts and has developed two AI solutions: *Makya* and *Spaya*. The first one, *Makya*, permits the generation of a novel ligand conditioned on a certain protein of interest while respecting chemical rules. The second one, *Spaya*, permits the creation of synthesis routes for a given molecule. Both solutions leverage deep learning to achieve high performance while remaining non time-consuming for users. The challenge to add AI into chemistry can revolutionize the way medecine is created [1].

1.2 Molecule Generation Model Targeting a Protein of Interest

1.2.1 What is a Protein?

A protein is a large, complex molecule made up of chains of amino acids. Proteins are essential for the structure, function, and regulation of the body's cells, tissues, and organs. They play crucial roles in virtually every biological process, including catalyzing metabolic reactions, DNA replication, responding to stimuli, providing structure to cells, and transporting molecules from one location to another.

1.2.2 What is a Ligand?

A ligand is a molecule that binds to a specific site on a protein, often to activate or inhibit its function. Ligands can be small molecules, such as drugs or hormones, or larger molecules, like other proteins or nucleic acids. The interaction between a ligand and a protein is fundamental to many biological processes, including cell signaling, enzyme activity, and drug action.

1.2.3 Ligand-Protein Interaction

Determining whether a ligand is suitable for a certain protein involves a combination of computational and experimental methods. Initially, computational techniques such as molecular docking and virtual screening are employed to predict how well a ligand might bind to the target protein. These methods use the three-dimensional structures of both the ligand and the protein to simulate their interaction and estimate binding affinity. Advanced algorithms can also predict the stability and specificity of

the ligand-protein complex. Following computational predictions, experimental validation is crucial. Techniques like surface plasmon resonance (SPR), isothermal titration calorimetry (ITC), and X-ray crystallography are used to measure the actual binding affinity and to visualize the molecular interactions. Additionally, cell-based assays and in vivo studies can provide further insights into the biological effects of the ligand on the target protein, ensuring that the ligand not only binds well but also produces the desired therapeutic effect. This multifaceted approach ensures that the selected ligand is both effective and safe for its intended use.

Hence it's a long process to find the good ligand with respect to a protein.

1.2.4 Generative approach

This is why more and more companies try to use generative models in order to find appropriate ligands.

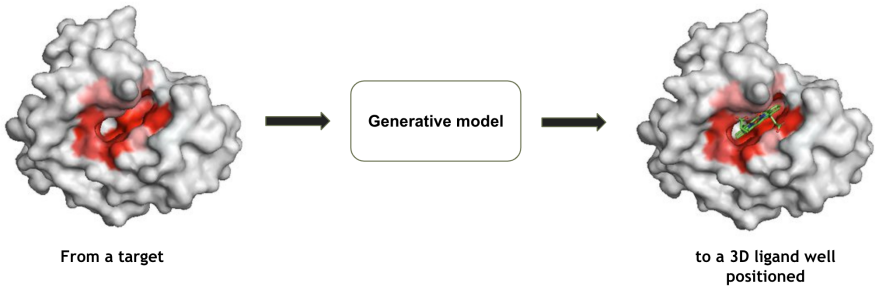


Figure 1: Ideal Protein-Ligand Generative Model

The ultimate goal would be as shown in Figure 1: A protein is given and then a ligand is generated but is also set in 3D in the pocket. The pocket of a protein refers to a specific region or cavity on the protein's surface where ligands can bind, often leading to the activation or inhibition of the protein's function.

But this generative model can be of any sort and multiple philosophies appears in the Molecule Generation Model Targeting a Protein of Interest community:

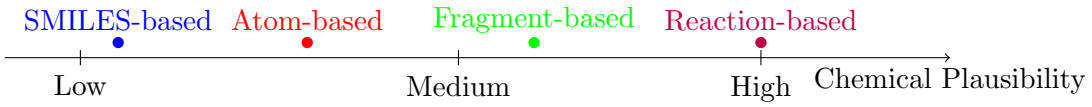


Figure 2: Approaches with Respect to Chemical Plausibility

- **SMILES-Based:** Uses Simplified Molecular Input Line Entry System (SMILES) strings to generate molecules. For example, "CCO" denotes ethanol. Reivent 4 [10] is a model that uses this type of data.
- **Atom-Based:** Molecules are generated by assembling individual atoms according to specific rules or algorithms. Pocket2Mol [21] is doing Atom Based generation
- **Fragment-Based:** Generates molecules by combining pre-defined chemically valid fragments or substructures derived from known molecules. FragDif [11] uses strategy
- **Reaction-Based:** Simulates chemical reactions to generate new molecules, using known reaction rules and mechanisms to ensure chemical plausibility. Aside from the Growing Optimizer from Iktos, SynFlowNet [14] is a model using this strategy.

But the more someone wants to have chemical plausibility the less diverse the predictions are going to be and some chemists prefer to have less plausible molecules but a greater number of them to give them ideas and starting points. Hence, a trade-off needs to be found to choose one approach.

1.2.5 3D Generation of conformation

To optimize a generative model for ligand design, it's crucial to also consider the 3D conformation of the generated molecules. Three primary methods are used:

- **3D Scoring:** Generate molecules in 2D and use a 3D scoring function to indirectly optimize the 3D conformation without explicit 3D generation as GenScore [5].
- **2D then 3D:** Generate molecules in 2D, then convert them to 3D as DiffDock [6].
- **3D Generation:** Directly generate the 3D structure of the molecule, bypassing the 2D representation entirely such as DiffSBDD [18].

1.3 Link to my internship

Iktos has already built a model to perform fragment-based ligand generation conditioned on a pocket. This model is called the "Growing Optimizer".

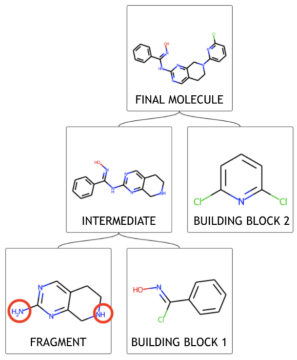


Figure 3: Growing Optimizer Generation

The current generation process focuses solely on utilizing building blocks without incorporating any 3D information related to them or the ligand. The aim of this work is to introduce 3D into the Growing Optimizer, allowing it to predict 3D coordinates and improve the generation process by optimizing the 3D affinity between a pocket and a ligand. Since the model is designed to predict building blocks in 2D, the task will involve leveraging this 2D data to generate 3D structures. Consequently, the subject of the internship is titled: **Creation of a 3D Molecule Generation Model Targeting a Protein of Interest.**

2 Methods, algorithm, state of the art

2.1 How to represent molecules

There are many ways to represent molecules:

Each representation has its own advantages and is commonly used in the generative domain. However, when predicting 3D conformation, a typical approach involves the use of 3D graphs. A 3D molecular graph $\mathcal{G}$ is defined as:

$$\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{X})$$

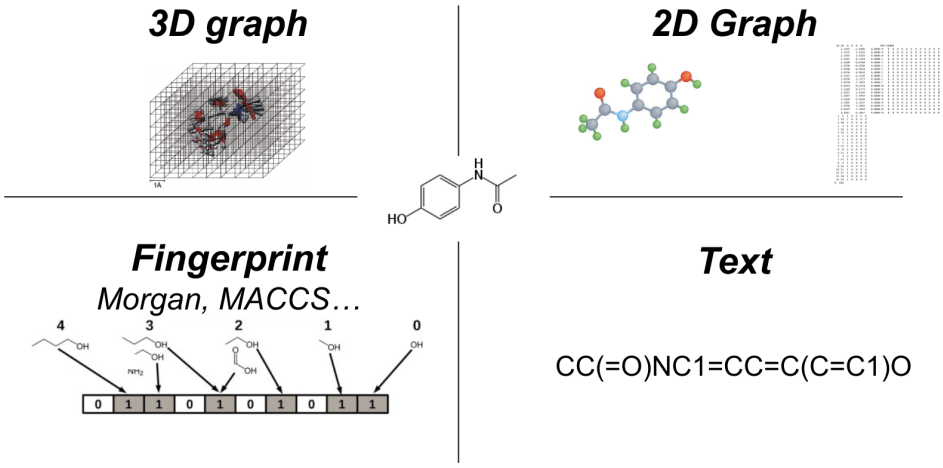


Figure 4: Different representation of molecules

where:

$$\mathcal{V} = \{(v_1, f(v_1)), (v_2, f(v_2)), \ldots, (v_n, f(v_n))\}$$

is the set of vertices (atoms) with associated feature vectors. Each vertex v_i has a feature vector $f(v_i)$ that may include: atomic number, atom type (e.g., C for Carbon, O for Oxygen), partial charge, hybridization state, and valence electrons.

$$\mathcal{E} = \{((v_j, v_k), f(e_i)) \mid v_j, v_k \in \mathcal{V}\}$$

is the set of edges (bonds) with associated feature vectors. Each edge $e_i = (v_j, v_k)$ has a feature vector $f(e_i)$ that may include: bond type (e.g., single, double, triple), bond length, bond order, bond angle, and aromaticity.

$$\mathbf{X} = \{(x_1, y_1, z_1), (x_2, y_2, z_2), \ldots, (x_n, y_n, z_n)\}$$

is the set of coordinates for each vertex v_i.

2.2 Euclidian considerations: Equivariance, Invariance, etc...

When considering geometrical graphs, it is important to account for their behavior concerning Euclidean symmetries: geometrical features are symmetric with respect to translations, rotations and even reflections for some systems, while scalar features remain invariant with respect to these transformations. Let us first define those transformations as in [4].

Rotations:

The group of rotation $\mathrm{SO}(d)$ (or rotation and reflections $\mathrm{O}(d)$) can be seen as orthogonal transformation matrices $R \in \mathrm{SO}(d)$ that acts on the vector feature and the the coordinates as $RG := (\mathcal{A}, \mathcal{S}, \vec{v}R, \vec{x}R)$. This action occurs because vector feature and coordinates are measured from a canonical base, so rotating the base rotates these attributes. However, scalar features are not base-dependent and are invariant to rotations.

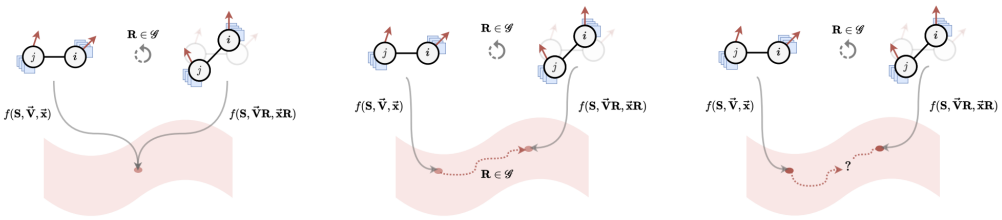
Translations:

The group of translations $T(d)$ can be seen as additive transformations $\mathbf{t} \in T(d)$ that act on the vector features and coordinates as $\mathbf{t}G := (\mathcal{A}, \mathcal{S}, \vec{\mathbf{v}}, \vec{\mathbf{x}}+\mathbf{t})$. It acts like this since coordinates are measured from a canonical origin; thus, translating the origin translates these attributes. However, translations

have no effect on the vector features at each node because these features are always calculated relative to the node's own coordinates.

This raises questions regarding the choice of model to be used: we use the notation $g \cdot x$ to represent the action of a group G on a space X, where $g \in G$ and $x \in X$. If G acts on both spaces X and Y, we can classify functions $f : X \to Y$ as follows:

- A function f is **G-invariant** if $f(g \cdot x) = f(x)$, meaning that the output remains the same regardless of the transformations applied to the input, as illustrated in Figure 5a.
- A function f is **G-equivariant** if $f(g \cdot x) = g \cdot f(x)$, implying that any transformation of the input results in a corresponding transformation of the output, as depicted in Figure 5b.
- A function f that is neither G-invariant nor G-equivariant is called **G-unconstrained**. In this case, a transformation of the input leads to an unpredictable change in the output, as shown in Figure 5c.



(a) $\mathcal{G}$-invariant function (b) $\mathcal{G}$-equivariant function (c) $\mathcal{G}$-unconstrained function

Figure 5: Invariant, equivariant, and unconstrained functions.

This issue becomes relevant in ligand generation: when rotating or translating the pocket, it must be determined whether the model should be constrained to be invariant, equivariant, or allowed to independently learn how to respond to these transformations. As will be discussed later, many prominent papers often adopt a specific approach when designing their models.

2.3 Prediction of 3D conformation: Graph Neural Network Approach

2.3.1 Graph Neural Network

A fundamental characteristic of graphs is that the vertices in $\mathcal{V}$ are not assigned any specific sequence. Consequently, operations on graphs should be independent of the order in which the vertices are listed; that is, each operation should be *permutation invariant*. A function f is permutation invariant if it yields the same result regardless of the order of the elements in its input set. Formally as stated in [4], if σ represents a permutation of the set $\mathcal{V}$, then f is permutation invariant if:

$$f(\sigma(\mathcal{V})) = f(\mathcal{V}) \tag{1}$$

for any permutation σ. In other words, applying a permutation σ to the vertices of the graph does not change the value of the function f.

And this is how Graph Neural Networks (GNNs) were designed. Considering, as in [7] a graph to be specified with an adjacency matrix $\mathbf{A}$ and node features $\mathbf{S}$. Node features s_i are updated from iteration t to $t+1$ in three steps: 1. Compute "**messages**" between the node of interest i and each of its neighbors $\mathcal{N}_i$ in the graph, via a learnable MSG function:

$$m_{ij}^{(t)} = \text{MSG}\left(s_i^{(t)}, s_j^{(t)}\right) \tag{2}$$

2. **Aggregate** all messages coming from $\mathcal{N}_i$ via a fixed permutation-invariant aggregation operator $\oplus$ (e.g., sum, mean):

$$\tilde{m}_i^{(t)} = \bigoplus_{j \in \mathcal{N}_i} m_{ij}^{(t)} \tag{3}$$

3. **Update** the representation of node i via a learnable function UPD, typically using both aggregated messages and its own representation as input:

$$s_i^{(t+1)} = \text{UPD}\left(s_i^{(t)}, \tilde{m}_i^{(t)}\right) \tag{4}$$

In practice, MSG and UPD are **neural networks**.

2.3.2 Geometrical Graph Neural Network

Geometric graphs refer to how molecules where previously represented in this report : $\mathcal{G} = (\mathcal{A}, \mathcal{S}, \vec{\mathbf{v}}, \vec{\mathbf{x}})$ is an attributed graph with scalar features $\mathcal{S}$ that is also decorated with geometric attributes: node coordinates $\vec{\mathbf{x}} \in R^{n \times d}$ and (optionally) vector features $\vec{\mathbf{v}} \in R^{n \times b \times d}$, sometimes denoted $\vec{\mathbf{v}}$ for convenience.

And the GNN architecture above does not take in account the coordinates or the vector feature of the graph. Hence, new architectures have been proposed to address this issue and can be viewed from a high-level perspective as follows: They update scalar (and vector) features from layer t to $t+1$ via learnable message and update functions, MSG and UPD, respectively, as well as a fixed permutation-invariant operator $\mathcal{L}$ (e.g., mean, sum):

$$s_i^{(t+1)}, \vec{\mathbf{v}}_i^{(t+1)} := \text{UPD}\left(s_i^{(t)}, \vec{\mathbf{v}}_i^{(t)}, \mathcal{L}_{j \in \mathcal{N}_i} \text{MSG}\left(s_i^{(t)}, s_j^{(t)}, \vec{\mathbf{v}}_i^{(t)}, \vec{\mathbf{v}}_j^{(t)}, \vec{\mathbf{x}}_{ij}\right)\right) \tag{5}$$

where $\vec{\mathbf{x}}_{ij} = \vec{\mathbf{x}}_i - \vec{\mathbf{x}}_j$ denotes the relative position vector.

2.3.3 Prediction of conformation: EGNN/DiffSBDD

An equivariant model was proposed to preserve equivariance to rotations and translations on a set of coordinates, as described in[17]. It is defined as such:

$$m_{ij} = \phi_e\left(s_l^i, s_l^j, \|x_l^i - x_l^j\|^2, a_{ij}\right) \tag{6}$$

$$x_{l+1}^i = x_l^i + C \sum_{j \neq i} \left(x_l^i - x_l^j\right) \phi_x\left(m_{ij}\right) \tag{7}$$

$$m_i = \sum_{j \neq i} m_{ij} \tag{8}$$

$$s_i^{l+1} = \phi_h\left(s_i^l, m_i\right) \tag{9}$$

The message and update logic involve the use of coordinates in the message-passing scheme, specifically through the relative squared distance between two coordinates. As shown in (7), the position of each node is updated as a vector field in the radial direction, representing the weighted sum of all relative differences. The weights are determined by a neural network, ϕ_x, which takes the iteration embedding m_{ij} as input. The constant C is set to $\frac{1}{M-1}$, as specified by the authors. Additionally, all possible interactions within the graph are considered, not just those between nodes connected by an edge, with sums taken across the entire graph.

The modifications are straightforward; however, the equivariance toward rotation and translation is ensured by these equation, and the model satisfies for any orthogonal matrix Q and translation g:

$$Qx^{l+1} + g,\ s^{l+1} = \text{EGNN}(Qx^l + g,\ s^l) \tag{10}$$

So we can see that it is not only equivariant for the output coordinates x^{l+1}, but it's also invariant for the output features s^l. More details can be found in Appendix Cfor the proofs.

Then **DiffSBDD** in [18] proposes two approaches to generate 3D conformations, but in both cases, the protein is also included in the model. However, its coordinates are not modified by (7) since the pocket should not be moved. Both approches are derived from [12] which is an extension of [17] to permit the diffusion to remain equivariant.

1. **First Method:** This method operates through simple refinement: by stacking EGNN layers and treating conformation generation as a regression objective.
2. **Second Method:** In this approach, EGNN is utilized as the noise model in a diffusion model, making conformation generation a true generative objective.

2.4 Prediction of Conformation: DiffDock – A Spherical Approach

DiffDock introduces a novel approach to this problem by utilizing a spherical coordinate system, enhancing the flexibility and accuracy of docking predictions. The method is based on the principles of E(3) Equivariant Neural Networks (E3NN) [8], enabling the model to respect the symmetries of three-dimensional space.

Spherical coordinates are a three-dimensional coordinate system where a point in space is represented by three values: the radial distance r, the polar angle θ (colatitude), and the azimuthal angle ϕ (longitude). This system is particularly useful for modeling systems with inherent rotational symmetry, as it simplifies the mathematical representation of angular relationships.

Traditional docking methods often rely on Cartesian coordinates, which can be limiting when modeling the rotational and translational degrees of freedom inherent in molecular structures. By switching to a spherical representation, DiffDock is able to capture these rotational symmetries more naturally. This spherical approach is especially advantageous in representing the angular orientation of ligands within the active sites of proteins.

In addition to spherical coordinates, DiffDock employs spherical harmonics, which are mathematical functions defined on the surface of a sphere. These functions provide a complete basis for representing angular distributions and can effectively model the orientation and symmetry of molecular conformations. Spherical harmonics are particularly valuable in capturing the complexities of ligand interactions with target proteins.

By utilizing E3NN, the model can maintain equivariance to rotations and translations, ensuring that the predictions remain consistent with the underlying symmetries of the molecular structures being analyzed. The model predicts the most likely binding pose by minimizing the difference between the predicted and actual conformations, using a loss function based on molecular distances and angles.

2.5 Prediction of 3D conformation: FLAG, a distance matrix approach

FLAG, as proposed in [23], proposes a novel approach compared to the two others seen above: the deep learning model does not directly predict 3D coordinates, but rather the distance matrix between

the ligand and the protein. It then employs analytical methods to go from distances to coordinates. Additionally, as the model operates with a distance matrix, the deep learning module in this paper is invariant, marking a significant distinction from the first two architectures.

The distance matrix $D \in R^{(n+m)\times(n+m)}$ is set as follows, where n' is the size of the pocket and m is the size of the ligand:

$$D_{i,j} = \begin{cases} \|s_i - s_j\| & \text{if } i,j \leq n, \\ \text{MLP}_d(h_i^{(0)}, h_j^{(0)}) & \text{if } i \leq n, j > n, \\ \|r_i - r_j\| & \text{if } i,j > n. \end{cases} \tag{11}$$

Given the known pocket, all $|s_i - s_j\|$ values can be easily computed. However, when predicting the 3D conformation of the ligand, the self-distance matrix is not available. To address this, a conformation is generated using RDKIT to provide a starting point that approximates the real self-distance matrix. For calculating the protein-ligand distance matrix, a multi-layer perceptron (MLP) is employed, which takes the input embedding of one pocket atom and one ligand atom and predicts the pairwise distance for each atom pair.

Once this matrix is calculate, they propose to use the eigenvalue decomposition of its Gram matrix to compute the coordinates for each atoms of the ligand and the pocket, as described in [2]:

$$\tilde{D}_{i,j} = 0.5\left(D_{i,1}^2 + D_{1,j}^2 - D_{i,j}^2\right), \quad \tilde{D} = USU^\top \tag{12}$$

Where S is a diagonal matrix with eigenvalues in descending order. Then the coordinates of each atoms are computed as:

$$\tilde{r}_i = [X_{i,1}, X_{i,2}, X_{i,3}], \quad X = U\sqrt{S} \tag{13}$$

One final step remains: the predicted coordinates are not in the same coordinate system as the true coordinates, so alignment is necessary. The Kabsch algorithm is employed to find a rigid body transformation, consisting of an orthogonal matrix R and a translation vector t, to align the two sets of coordinates. This process ensures that the ligand coordinates are properly aligned within the correct coordinate system:

$$r_i = R\tilde{r}_i + t, \quad i > n \tag{14}$$

2.6 How to create molecular features?

One concept used in the later section was node feature/embedding. What is called a node feature or embedding is a way to give contextual information about a node in a graph. This is really important in the context of molecular deep learning, an atom is a very singular structure: for example, a carbon atom is very different from a nitrogen atom. Beyond these intrinsic structural differences, the embedding must also convey contextual information about the atom within the ligand, such as the position in the ligand, the valence of the atom and more. Making a model capable of understanding this type of information is crucial for its performance.

2.6.1 Simple embedding: atom properties

One way to represent an atom in a molecule is by its atomic properties which can include characteristics such as atomic number, atomic mass, electron configuration. These features can then be processed using a dense layer to create embeddings, aenabling a model to be trained to generate meaningful representations from these basic atomic properties.

2.6.2 ESM-2

ESM-2 in [13] is a model for the structural characterization of metagenomic proteins. It leverages the revolution started with Attention Is All You Need [19] by building on a BERT encoder structure. It works on text data: the SMILES representations of the proteins, aiming to uncover a close link between language modeling and the learning of structure. To learn from text, a learning process must be established. ESM-2 is based on a BERT-like architecture and utilizes the same objective as BERT. In this approach, the model is trained to predict the identity of amino acids that are randomly masked within a protein sequence. The model learns to infer the masked amino acids by analyzing the context provided by the surrounding amino acids in the sequence, following this objective:

$$\mathcal{L}_{\text{MLM}} = -\sum_{i \in \mathcal{M}} \log P(x_i \mid \mathbf{x}_{\setminus i}) \tag{15}$$

Having:

- $\mathcal{M}$ denotes the set of indices of masked amino acids in the sequence.
- x_i represents the true identity of the amino acid at position i.
- $\mathbf{x}_{\setminus i}$ refers to the sequence with the amino acid at position i masked out.
- $P(x_i \mid \mathbf{x}_{\setminus i})$ is the probability predicted by the model for the amino acid at position i, given the context provided by the unmasked portions of the sequence.

During training, the model encountered almost 65 million unique sequences. It employs a simple unsupervised training objective; achieving good performance requires that the model learns sequence patterns across evolution.

2.6.3 Uni-mol

Like ESM-2, Uni-Mol [24] is a molecular representation learning model. However, while ESM-2 learns solely from 1D information, specifically the protein sequence, Uni-Mol aims to incorporate 3D information into the molecular representation learning model. This enhancement allows the embeddings to be applicable for tasks that rely on 3D information. This is achieved by adding tasks into the training:

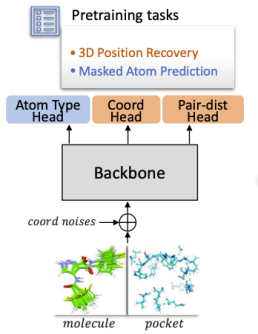


Figure 6: Uni-Mol Training architecture

1D-objective:
This is the same objective as ESM-2 and uses only 1D information to predict a masked element of a sequence (either protein or ligand) and is trained on the Atom Type Head.

3D-objectives:
In order to include 3D information, Uni-Mol developed a novel self-supervised task. The idea for this task is derived from the 1D objective: it aims to retrieve the correct 3D positions while being given corrupted input positions. Since those values are continuous, a simple masking strategy cannot be used. Hence, they decided to use random 3D positions as corrupted inputs. However, this objective is challenging for a model to learn; therefore, two strategies were proposed to make the learning more feasible:

- Allowing random positions to only vary within a certain noise range r around the ground-truth positions.
- In the re-assignment step, given m atoms and m random positions, there are m! possible ways to assign the atoms to the positions. To choose the most appropriate assignment, it applies the stationary-action principle, which suggests selecting the assignment that minimizes the total displacement (i.e., the difference in positions). However, since finding the optimal solution is computationally difficult, it uses an efficient greedy algorithm to approximate the best assignment, resulting in a sub-optimal but reasonable solution

Then those corrupted inputs are given to two heads:

- **Coord Head:** Using an SE(3) equivariant coordinate head given by:

$$\hat{x}_i = x_i + \frac{1}{n}\sum_{j=1}^{N}(x_i - x_j)c_{ij} \tag{16}$$

  The predicted position after adjustment is $\hat{x}_i$, the original coordinate of the atom is x_i, n is the number of atoms, and c_{ij} represents the mutual information between the i-th atom and the j-th atom, which is provided by the architecture. This value indicates how the position of atom j relates to atom i.

  It is required for the model to retrieve the real atomic coordinates. This iterative method closely resembles the approach used in the EGNN architecture.

- **Pair-Dist Head:** Based on pair-representations, the model needs to predict the correct Euclidean distances of the corrupted atom pairs.

Even though the backbone of Uni-Mol can output SE(3)-equivariant positions,, it is based on SE(3) invariant representations: **it produces SE(3) invariant embeddings for both ligand and protein.**

3 Contribution

3.1 The problem

The focus of this internship is on a 3D molecule generation model targeting a protein of interest. As noted in the Introduction section, various methods can be employed in generative approaches to achieve this goal. Yet, Iktos has already developed what is called the **Growing Optimizer**.

This process is an auto-regressive method that generates a ligand conditioned on a protein. Since it is a reaction-based approach, no conformation is generated at each step; rather, only the appropriate building block from the database is selected.

There are multiple blocks are used throughout the whole generative process.

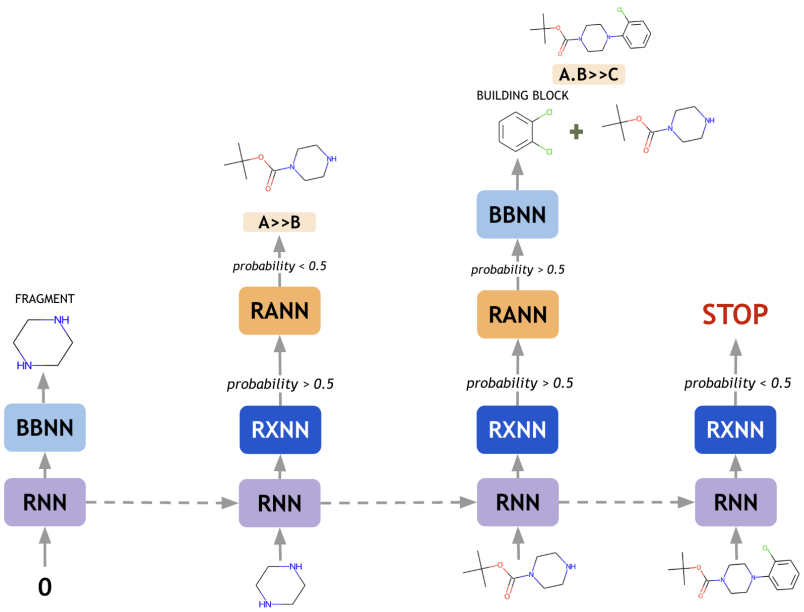


Figure 7: Growing Optimizer generating a molecule

- **RNN:** It is the autoregressive block that accumulates information about the past reactions and is given in input the embedding of the pocket using Uni-Mol and the ligand at the last step.
- **RXNN:** This block decides whether to stop the generative process or to continue it by outputting a probability and having a threshold at 0.5.
- **RANN:** This block decides to perform a $A \to B$ or a $A \cdot B \to C$ reaction by outputing a probability and having a threshold at 0.5.
- **BBNN:** This block is only used if the **RANN** outputs a probability greater than 0.5. It outputs probabilities for each building block in the database in order to perform a $A \cdot B \to C$ reaction.

And for each step, the final output is the one-hot encoding of the building blockalong with another representation: the Morgan fingerprint of the fragment. A morgan fingerprint is defined as: a type of molecular fingerprint generated using a circular encoding method. This approach considers the structure of the molecule, creating a fixed-length binary or integer vector that encodes the presence of substructures and their connectivity. The Morgan fingerprint captures both local and global features of the molecule, allowing for effective similarity comparisons between different building blocks.

The idea of the internship is not to alter the existing structure of the Growing Optimizer but rather to retain it and add a 3D module to it that can predict 3D structure of the ligand based on the 2D information already available. Hence, the generation will follow **2D then 3D approach** as described in Section 1.2.5.The process will begin with a 2D molecule generated from the Growing Optimizer,aiming to find its optimal position within the pocket.

In this auto-regressive process, there are two options: either to determine the coordinates of each fragment as it is generated and then assemble the complete conformation or to wait until the entire generation is complete and then generate the full conformation. These two approaches will be discussed later.

The approach employed is a **reaction-based module**. While **building blocks** (i.e., fragments) are utilized, the method primarily relies on specific chemical transformations. As illustrated in Figure 7, a choice of reactions is made:

- **A** $\rightarrow$ **B**: This indicates that a specific chemical transformation from compound A to compound B is selected based on predefined reaction rules.
- **A.B** $\rightarrow$ **C**: Here, a building block is selected from the database, allowing the ongoing ligand to react with it.

This reaction-based approach ensures good chemical plausibility while maintaining diversity through the use of building blocks.

3.2 Dataset

To understand how the existing Growing Optimizer architecture can be utilized to predict 3D coordinates, it is essential to examine the dataset provided for training the Growing Optimizer and especially the 3D module.

The dataset is composed of what is called **Growing Trees**. These trees represent a whole inference of the growing optimizer as shown in Figure 3 and they are created from the inference of Spaya, the retro synthesis tool of Iktos. For each of those trees, it's given with the input representation of each building block and the input representation of the protein that is conditioning the generation.

A **building block** is initially defined as follows: It consists of reactant molecules that are chosen based on their frequency of occurrence in the USPTO dataset. Specifically, building blocks are identified as reactants that appear at least 15 times and react with other molecules that are also present at least 15 times within the dataset. And more the growing optimizer project was developing more building blocks where added to the set.

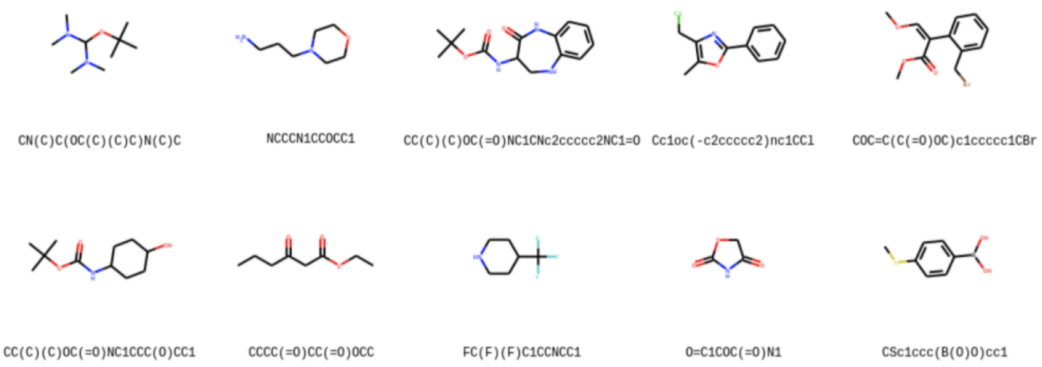


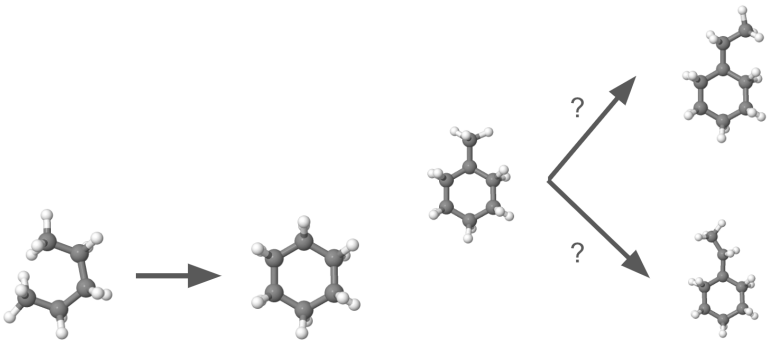
Figure 8: – Example of molecules from the pool of building block and their associated SMILES notation

Given the use of a finite database of building blocks, these blocks are represented using one-hot encoding.

In order to represent the protein for the 2D model, it was chosen to embed it using Uni-Mol, this permits us to have a dense vector of size 512 that represent the whole protein. By doing so, it gives a pretty small representation to our model in order to conditionate the 2D generation.

The size of the growing tree is not unique; one growing tree may represent only one reaction, while another can consist of several reactions. In total, there are **107,552 growing trees**, representing **251,201** $A \cdot B \rightarrow C$ **reactions**. The $A \rightarrow B$ reactions are not considered, as only the final coordinates of the ligand are accessible. In the case of an $A \rightarrow B$ reaction, the atoms that disappear do not appear

in the final ligand; therefore, from a 3D coordinate perspective, this reaction does not provide any new coordinates to predict. Conversely, in an $A \cdot B \rightarrow C$ reaction, some atoms of B will be present in the final ligand, allowing for the extraction of these atoms to obtain partial 3D coordinates for the building block. Atoms that are not present in the final ligand will not be considered, as no suitable method to infer them has been identified.



(a) Infering the missing atom of a ring (b) Infering a rotable bond

Figure 9: The two main atom inference the problem faces

On the 3D module side, the 3D coordinates of the complexes come from the publicly available database **PDBbind** [15]. This database is a curated resource for protein-ligand complexes with experimentally measured binding affinity data. PDBbind integrates 3D structural information of protein-ligand complexes, which are extracted from the Protein Data Bank (PDB), along with binding affinity data from various experimental sources. The PDBbind database is widely used in molecular docking, virtual screening, and structure-based drug design for the evaluation and development of predictive models for protein-ligand interactions.

Hence, for each growing tree : **Building Blocks ID, Pocket Representation, 3D coordinates** (either of the final ligand, or splited in fragments for atoms that remain in the final molecule) are provided and can be used by both the 2D module or the 3D module.

More precise summary statistics about the ligand and protein complexes can be found in Appendix F.

3.3 Overview of the proposed approaches

Before delving into the details of each architectures, here is an overview of what was done during the internship.

First, the objective was to try to **infer the centroids** of each building blocks and 2 approaches were defined. One using a simple Multi-Layer-Perceptron and the other one using an equivariant approach with a Graph Neural Network called EGNN. High view representation of those two approaches are proposed in Appendix A.

Then, following the promising results of the proposed centroid architecture, the objective was set to predict the positions of all the atoms of the ligand. To this end, two architectures were designed with different approaches. One architecture aims to generate a distance matrix D between the ligand and the protein, defined as:

$$D_{ij} = \|\mathbf{r}_i - \mathbf{r}_j\|_2$$

where $\mathbf{r}_i$ and $\mathbf{r}_j$ are the position vectors of atoms i and j, respectively, and $\|\cdot\|_2$ denotes the Euclidean distance. This approach then transforms the distance matrix D into Cartesian coordinates C using

a suitable reconstruction method. The second architecture using EGNN again directly infers the coordinates C of each atom in the ligand, modeled as:

$$\mathbf{C} = f(\mathbf{X})$$

where $\mathbf{X}$ represents the input features relevant to the ligand's structure and f denotes the function learned by the architecture.

High-level representations of these two approaches are provided in Appendix B.

For each model, the same loss objective is used, whether it involves a distance matrix or coordinates. The **Mean Squared Error** (MSE) loss is applied:

$$\mathcal{L} = \text{MSE}(Y, \hat{Y}) = \frac{1}{n} \sum_{i=1}^{n} \left(Y_i - \hat{Y}_i \right)^2 \tag{17}$$

where Y represents the ground truth (either a distance matrix or coordinates), $\hat{Y}$ denotes the model's predictions, and n is the number of data points.

3.4 Prediction of fragment's centroids

The initial step of the internship concentrated on a simpler task than predicting the complete conformation of either the fragments or the entire generated ligand. The focus was on predicting the centroids of each fragment. The centroid of a fragment is defined as:

$$\left(\frac{1}{n} \sum_{i=1}^{n} x_i, \frac{1}{n} \sum_{i=1}^{n} y_i, \frac{1}{n} \sum_{i=1}^{n} z_i \right) \tag{18}$$

Thus, at each step of the autoregressive process, if the reaction is $A \cdot B \rightarrow C$, a new fragment B is introduced, and a model will be trained to predict its centroid.

To achieve this, it is necessary to understand what inputs can be provided to the model to enable the prediction of each centroid.

Here is how a 3DNN can be integrated into the architecture of the Growing Optimizer:

There are three main possible inputs for the 3DNN: the Morgan Fingerprint of the building block (1), the RNN (2) and additional information (3) not contained in the 2D model. For a description of each block used in the generative process, the reader may refer to the Section 3.1.

3.4.1 Predicting centroids using an MLP

An initial design was already launched in-house by Iktos. It was a simple model using a multi-layer perceptron (MLP) that takes as input the concatenation of the GRU and the Morgan fingerprint and outputs centroid.

3.4.2 Predicting centroids using EGNN

3.4.2.1 Representation of the atoms and centroid:

To utilize the EGNN, an initial embedding must be provided for each point (centroid or protein

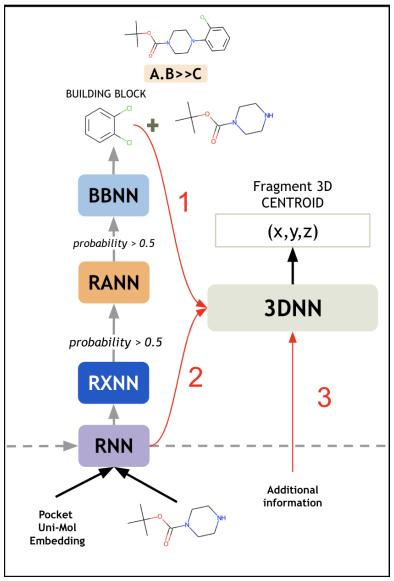


Figure 10: 3DNN imbrication into the growing optimizer

atom). Since a 2D prior is desired, some raw output from the 2D model will be supplied to the 3D model.

First Idea:

The initial idea was to use the **GRU output as a representation for each atom of the protein** and use the **morgan fingerprint of the selected fragment as a representation for the centroid**. Each atom of the protein would be represented in the same way. However, the representation of the pocket would differ at each timestep as the output of the GRU is not the same even if it's the same pocket that is represented during one generation.

This approach proved suboptimal. By using a graph neural network (GNN), the model's strength comes from allowing each node to have its own representation. Assigning the same representation to the entire protein, the full potential of the GNN was not utilized. As demonstrated in the results, this model could not be optimized, as shown in Appendix D where the loss remained unchanged, indicating that the model was not learning.

Second Idea:

It was determined that a unique representation for each atom of the pocket is necessary. As mentioned in Section 2.6, existing methods can achieve this. Given the focus on 3D data, the **Uni-Mol model** was chosen to embed each atom of the pocket. This involved using the model as provided on GitHub and utilizing one of the freely available checkpoints. However, as stated previously, it is still important to incorporate 2D information into the 3D model. Therefore, it was decided to use the concatenation of the Morgan fingerprint of the fragment and the output of the GRU as the input for the **representation of the fragment's centroid**. This representation combines discrete (Morgan fingerprint) and dense (GRU) components. To facilitate the model, a Morgan fingerprint encoder was implemented to convert this part of the representation into a dense format.

Then since the embedding sizes are not generally the same for the atoms of the pocket and those of the ligand, an encoder is used to map both representation to the same space using a simple MLP.

3.4.2.2 How to predict using the pocket:

It is known that the EGNN uses Equation 7 to update the coordinates of each atom. Two key

questions arise: 1) What initial coordinates should be provided for the centroids? 2) How can the coordinates of the protein be used to update the coordinates of the fragment?

Initial coordinates of the centroid:
As a refinement strategy, input coordinates for the fragment must be provided. The initial idea was to set the coordinates at $(0, 0, 0)$; however, the model encountered difficulties in learning. Consequently, it was decided to position the fragment at the **center of the pocket** and add a small Gaussian noise to ensure that each fragment of the same ligand does not start from the same location, thereby preventing the establishment of any unintended connections.

Use the pocket information:
In this approach, the concept of an adjacency matrix A is utilized to represent the relationships between atoms in a ligand and those in the associated protein. The adjacency matrix is defined for each batch of data as follows: The matrix A is a square matrix where each row and column corresponds to an atom. The entry A_{ij} is defined as:

$$A_{ij} = \begin{cases} 1 & \text{if atoms } i \text{ and } j \text{ belong to the same fragment or protein} \\ 1 & \text{if atom } j \text{ is an atom of the protein associated with the ligand of fragment } i \\ 0 & \text{otherwise} \end{cases} \tag{19}$$

An important distinction in this context is that, in DiffSBDD, the process is not auto regressive; therefore, the protein is represented only once. However, in the current auto regressive setup, following the initial code would require representing the protein as many times as there are fragments associated with the ligand in the growing tree. This approach would not be suitable in terms of memory usage. Consequently, the code was modified to represent each protein only once.

And in Equation 7 we use the new adjacency matrix in order to update the fragment coordinate and consider the neighbourhood of each data point and not all $i \neq j$.

3.4.2.3 Making the coordinate generation autoregressive:

Since the architecture of the growing optimizer is relying on an auto regressive process, an intuitive design tweak would be to also make the generation of the centroids auto regressive. And it's also intuitive in a physical sense, if the model knows where then centroids are $[c_1, ..., c_{t-1}]$ it might be able to generate more effectively the centroid c_t. To do so, we can infer from the logic of the adjacency matrix and place a 1 between i and i -1 if they both belong to the same ligand.

Refer to Appendix E for a toy adjacency matrix having all the properties referred above.

3.4.2.4 Making the training more stable:

EGNN training is known to be a challenging task [3] because tvarious instabilities can arise and cause the training either not to learn or to crash. A few techniques are available online on [20], and the author of the EGNN paper was also contacted for guidance on making the training more feasible. The first trick is to not predict $(x_{\text{truth}}, y_{\text{truth}}, z_{\text{truth}})$ but $\left(\frac{x_{\text{truth}}}{5}, \frac{y_{\text{truth}}}{5}, \frac{z_{\text{truth}}}{5}\right)$ and during inference multiply by 5 the outputs in order to have the real coordinates. Another idea that was suggested is to clamp weights of the coordinate head of the EGNN ϕ_x: doing so limits each value of the coordinates head and makes training easier. Another thing, gradient clipping because when the gradients during back propagation become too large, it causes the model to fail or result in NaN (Not a Number) values. To do so, the global norm can be calculated as:

$$\text{global\_norm} = \sqrt{\sum_i \|\nabla \theta_i\|^2} \tag{20}$$

If this value exceeds the threshold, each gradient is scaled down as:

$$\nabla\theta_i \leftarrow \nabla\theta_i \times \frac{\text{threshold}}{\text{global\_norm}} \tag{21}$$

One last thing that is also brought up is a normalization during the updates of the coordinates, 7 becomes:

$$x_i^{l+1} = x_i^l + \sum_{j \neq i} \frac{x_i^l - x_j^l}{d_{ij} + 1} \cdot \phi_x(h_i^l, h_j^l, d_{ij}^2, a_{ij}) \tag{22}$$

The key difference here is that the term $\frac{x_i^l - x_j^l}{d_{ij}+1}$ modifies the strength of the interaction between node i and its neighbors j based on their distance d_{ij}. This introduces a notion of locality or neighborhood awareness, where nodes that are farther apart (with larger d_{ij}) contribute less to the update of the position x_i^{l+1}.

3.5 Prediction of the whole conformation:

Considering the time constraint of the 6-month internship and the favorable results obtained from the EGNN architecture, it was determined that the focus would shift to predicting the entire conformation of the ligand, specifically the coordinates of each atom in the ligand. As demonstrated in the dataset presentation, there may be missing atom positions at each step due to limited access to the final atoms, making it challenging to infer the positions of those that are absent. Consequently, predicting the entire conformation of each fragment would not be feasible, as some atoms would lack corresponding ground-truth coordinates. Therefore, it was decided that the prediction of the entire conformation would occur at the end of the generative process, rather than predicting fragment by fragment and merging them.

3.5.1 Merging FLAG and Uni-Mol

As presented previously, FLAG provides a method to predict a distance matrix between a complex and a ligand, and proposes a way to convert this matrix into coordinates without an optimization process. However, the architecture proposed to infer the distance appeared quite shallow: utilizing only a few atomic features and a two layer MLP and someone at Iktos previously fine-tuned the Uni-Mol model with an Iktos' proprietary dataset. This is why a distance head was incorporated at the end of the proposed architecture to predict the distance matrix.

Hence, we integrate Uni-Mol to the architecture proposed by Flag. This integration ensures that the features representing the interaction between a ligand atom and a protein atom are state-of-the-art, utilizing Uni-Mol approach to modeling these interactions between a protein and a ligand. In order to incorporate 2D information, the GRU output is concatenated to each entry of the distance matrix. Moreover, it is important to note that no machine learning step is employed after the prediction of the distance matrix; thus, the loss is calculated based on the distance matrix rather than on the coordinates, as proposed by FLAG.

Yet, it will be noted in the results section that the inversion did not yield satisfactory outcomes. This prompted the exploration of an alternative approach to convert a distance matrix into coordinates: framing it as a straightforward optimization problem. Although this may not be the optimal solution, as it involves an instantaneous step with one that involves optimization, it is necessary to invert the distance matrix. Since these steps do not rely on machine learning, none of these methods will affect the model's ability to accurately produce a distance matrix. Therefore, the optimization problem to be solved is outlined as follows:

$$\min_{L} \sum_{i=1}^{N_P} \sum_{j=1}^{N_L} (\|P_i - L_j\| - D_{ij})^2 \tag{23}$$

where N_P represents the number of protein atoms, N_L is the number of ligand atoms, $\|P_i - L_j\|$ is the Euclidean distance between the i-th protein atom and the j-th ligand atom, and D_{ij} is the target distance between protein atom i and ligand atom j.

It is known that this problem is not convex, as the Euclidean distance function $\|P_i - L_j\|$ is not convex in the variables L_j when P_i is fixed. However, based on tests conducted using the ground truth distance matrix, it has been demonstrated that the positions can be effectively recovered. Therefore, the results obtained from a straightforward gradient descent algorithm algorithm are regarded as satisfactory.

3.5.2 EGNN Full conformation

It was then decided to modify the code from the EGNN centroid to enable the prediction of the entire conformation of the ligand. The underlying logic remains the same; however, instead of predicting only one point (the centroid) at each step, all the atoms of the ligand will be predicted at the final step of the generative process.

The representation used for the pocket was not altered, but a suitable representation for the atoms of the ligand needed to be determined. Since Uni-Mol cannot be used due to the lack of access to the 3D coordinates required for embedding a molecule with this method, it was noted that two methods are still available for embedding a molecule without requiring 3D coordinates, as discussed in Section 2.6. The simpler method was chosen, considering the dataset's substantial size (150 GB). Therefore, in alignment with GENSCORE and recommendations from Iktos' chemists, the following features were selected to represent a ligand's atom: **atomic number, chirality, degree, formal charge, implicit valence, the number of connected hydrogens, the number of radical electrons, hybridization type, whether it is part of an aromatic ring, the number of rings it is in, and whether it is in a ring of size 3, 4, 5, 6, 7, or 8.**

4 Results

The models will be compared based on **50 epochs of training**, with the best validation loss achieved during those 50 epochs being saved for evaluation. And each benchmark will be made on the same valid data set to ensure comparability across different models and parameters.

Moreover it was found by the previous team members on the project that learning both the 2D and the 3D from scratch could be a challenging task, hence better result were observed by using a **2D pretrain checkpoint**. A model is trained without the 3D objective for a time of 20 epochs and the 3D model will have the 2D weight initialised based on this checkpoint. For the 3D weights, they are initialised in the usual manner.

Additionaly, during the internship, a refactor of the Growing Optimizer was done in order to permit parallel training and mixed precision as in [16] by implementing Pytorch Lighting into the code. This package unfolds as an easy way to have a parallel training and easy save of each models. This resulted by a cut of both the time of training of the model (both in 3D or 2D setup) and a cut in costs as it cheaper to train on several GPU's for less time than the classic setup.

Hence each of those training is made on **4 NVIDIA T4 and 48 VCPU.**

For each model several losses are going to be used to asses their performance.

- **val_loss**: The overall aggregation of losses over the whole validation dataset.
- **loss_3D_frag**: The part of the validation loss that account for the prediction of the 3D caracteristics.
- : **loss_bbs** : The part of the validation loss that accounts for the prediction of the right building block.
- **loss_reactant_addition** : This loss is about choosing the right reaction.
- **loss_rxn_addition** : This loss is about continuing or not the generative process.
- **tree_loss** It's all of the above loss but grouped for each growing tree and they their mean is computed.

The focus will primarily be on two components of the validation loss: **loss_3D_frag** and **loss_bbs**, as these two components account for 3D and 2D performance, respectively, and represent the main characteristics of the model: effective 3D and effective 2D representations.

4.1 Centroid Prediction

The initial step involves comparing the EGNN model to a 2D baseline (a model that does not incorporate any 3D module) and to a previously proposed naive version that solely utilizes a multi-layer perceptron (MLP) with two layers. The MLP is implemented as follows: a sequential model with a first linear layer that takes an input of size equal to the sum of **256 (Morgan fingerprint) + 512 (GRU)**, and maps it to **256**, followed by a **ReLU activation** function, and a second linear layer goes from **256 to an output of size 3**.

Given that the training of the EGNN was particularly lengthy and resource-intensive, the introduction of a frugal version of the EGNN was proposed. The classical EGNN consists of four layers of EGNN (same parameters as in DiffSBDD : **shared_dim = 128, hidden_dim = 256, attention = True, tanh = True**), without any inverse sub layers, while the frugal version includes only one EGNN layer and two sub layers (only h updates no x update). This modification results in a model with four times fewer parameters, enabling faster training and, importantly, quicker inference. This adjustment is advantageous, as it is essential for Iktos to process data in large batches. During the training of the standard EGNN, it was only feasible to work with batches of four growing trees and this frugal method can infer in batches of 16.

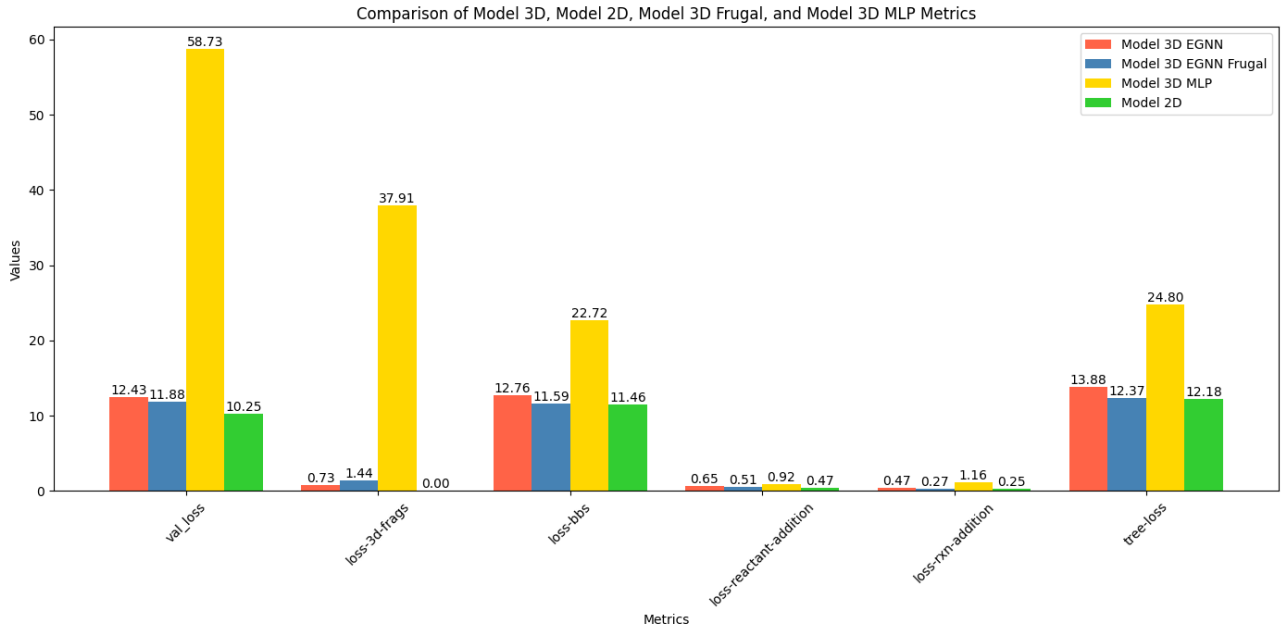


Figure 11: Assessment of EGNN centroid performance

To first focus on 3D results, the classic 3D EGNN outperforms by a large margin the naive MLP model that was proposed. Its loss on the centroid is only 0.73 Angstrom demonstrating its strong performance as seen by the representation following. On the other hand, the frugal model that achieves 4 time faster training and 4 time less parameters as only two time the 3D loss of the EGNN. And for the iktos internal team of chemist, this higher value is still good and makes it a usable value.

For the 2D results, the naive MLP model is still the worst one, as the loss-bbs is double from what an only 2D model would do. Looking at the two EGNN models, the classical model is a higher than both the frugal EGNN and the only 2D model, but it's only by a small margin. Hence two conclustions can be made: First about the choice of a frugal model or not: the small margin that seperate the two shows that it's a question of tradeoff: the classical is going to be slighty better in 3D generation but is going to be less accurate on the building blocks compared to the frugal model which infers the 3D coordinate faster and is more scalable. Secondly, both EGNN model don't achieve to outperform the 2D only model in terms of 2D loss, this indicates that there there is no distillation of 3D information to the 2D architecture: trying to infer 3D coordinates as such doesn't permit the 2D generation to be better. This is kind of a setback to what was originally expected as the goal was to be able to both predict good 3D and make the overall model better. This might be explained that predicting 3D coordinates is an hard task and in the actual setup of the model.

Two different examples were sampled from the training dataset to visualize the results. In the visualizations, the red cross represents the predicted centroid, while the green cross indicates the ground truth centroid. Although the choice of examples was random, care was taken to ensure that the two selected examples differ in properties and shape, demonstrating the model's performance across various regions of the complexes' distribution

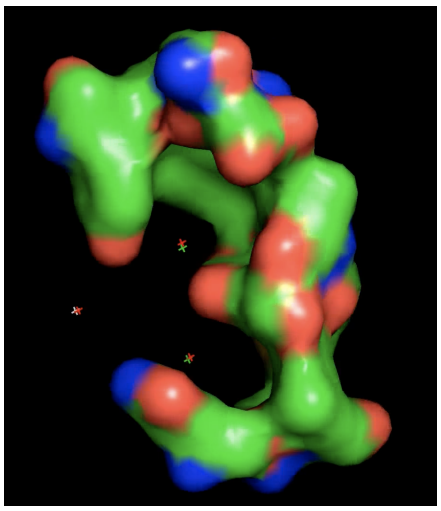


Figure 12: Exemple 1

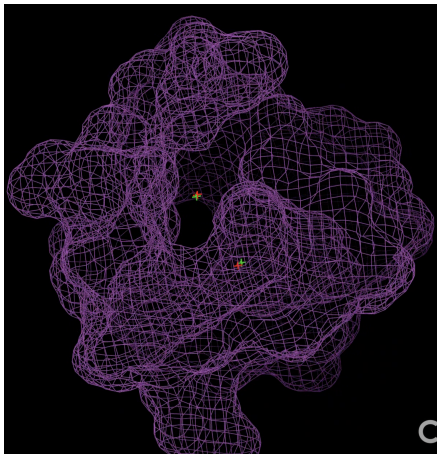


Figure 13: Exemple 2

Figure 14: Random complexes from validation dataset

To continue to have a critical discussion about the overall architecture of the model, ablation studies were conducted in order to understand assess the utility of specific features : the use of the Uni-Mol embeddings for the Protein atoms and also the autoregressive logic brought up.

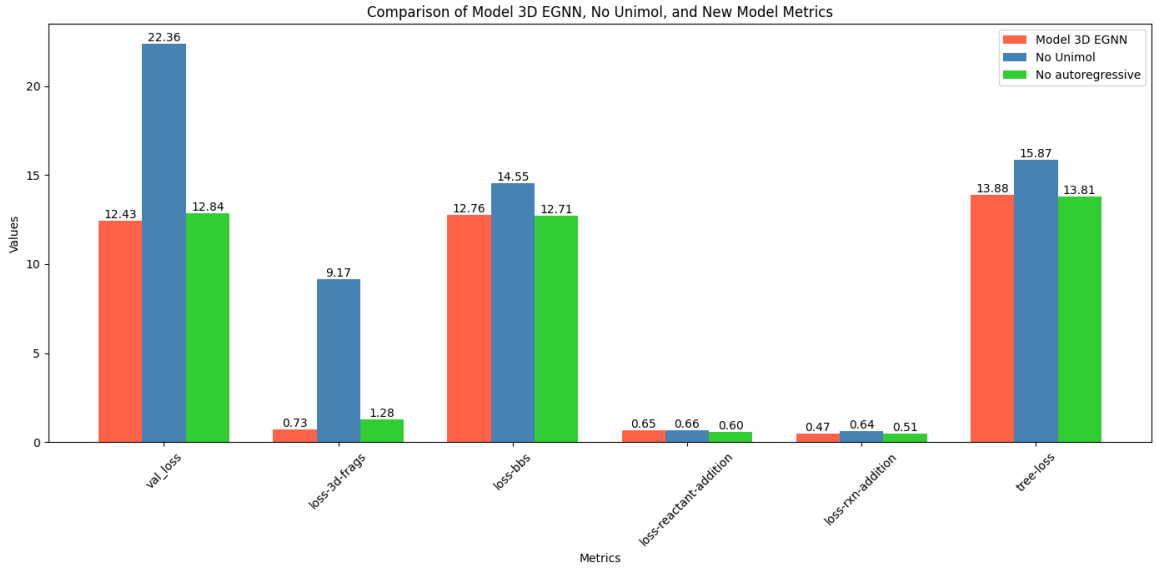


Figure 15: Abblation studies for the EGNN centroid

As seen the performance are the best on the configuratio,n here we use the Uni-Mol embedding and we have an autoregressive generation of the centroids. For Unimol the difference is quite significative in terms of 3d loss.

4.2 Full conformation prediction

4.2.1 Distance matrix approach

Before assessing the performance of the distance approach model, an evaluation of the inversion methods must be done in order to understand if the proposed method in FLAG can be robust to

noisy matrices. If it is not, a traditional optimization approach will be considered for its robustness. For the traditional optimization method, a simple PyTorch loop is implemented, with the objective as described in Equation 16. The number of optimization steps was chosen arbitrarily to be 1000.

The inversion method from flag requires 3 components: the self distance matrix of both the ligand and the protein, and the inter distance matrix between those two. The ground truth position of the protein is known, allowing for easy calculation of its distance matrix. For the ligand, it is proposed to use RDKIT in order to generate an energy-wise feasible conformation to calculate a ligand distance matrix that will be close to the final one. The inter-distance matrix will be predicted by the model.

For the experiments, the model will be provided the ground truth distance matrix to which an increasing amount of noise will be added to each of its components, then the inversion takes place and a mean squared error (MSE) scoring is used to assess the performance of the inversion.

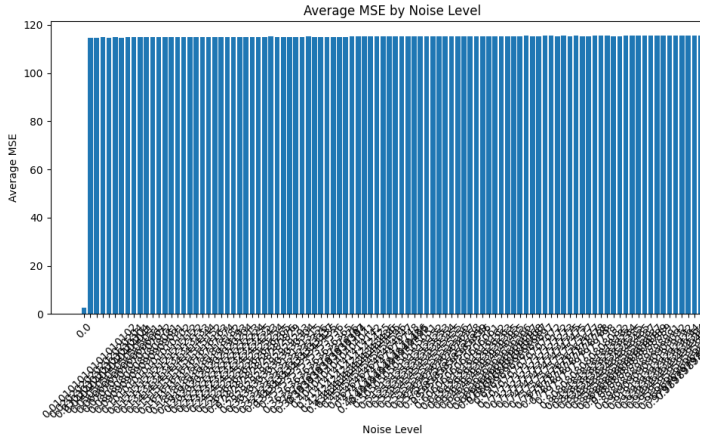


Figure 16: MSE with Flag Inversion

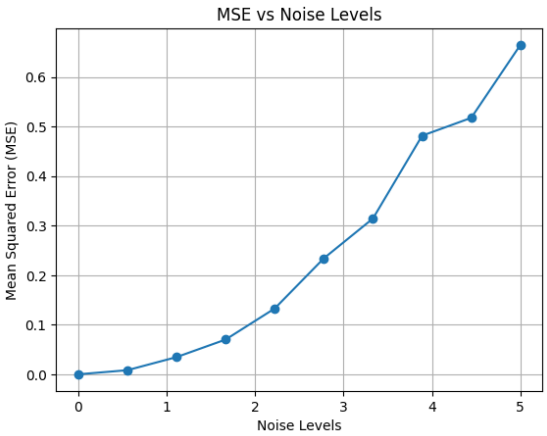


Figure 17: MSE with Optimization Inversion

Figure 18: Comparison of Mean Squared Error (MSE) with respect to the noise of the added noise using different inversion methods.

As seen in Figure 16, the quality of the inversion degrades as soon as any noise is added, and it fails to function effectively even with a very small amount of noise. This indicates that this method is not suitable for a deep learning context. The model attempts to infer the distance matrix; in an ideal scenario, this matrix would be perfectly predicted. However, in practice, the predicted matrix remains noisy compared to the actual one. If this noisy matrix were to be input into the FLAG inversion method, the resulting MSE would be 120, rendering the conformation entirely incorrect.

On the other hand, Figure 17 demonstrates that the classical approach to inversion, which involves solving an optimization problem, is more robust to noise compared to the FLAG method. However, this optimization approach is significantly more time-consuming; while the FLAG method provides instantaneous results, the optimization process can take considerable time depending on the sizes of both the ligand and the protein. This is why no benchmark on the coordinates has been conducted to date, as it would require excessive time and resources.

In terms of parameters, all the weights of Uni-Mol are frozen and the distance head used is as follows: the embedding dimension is the sum of **512 (from a GRU) and 64 (from Uni-Mol)**, resulting in a total embedding dimension of **576**. The DistanceHead consists of a linear layer that maps the 576-dimensional input to a 288-dimensional output, followed by a GELU activation function, layer normalization applied to the 288-dimensional output, and a final linear layer that maps the 288-dimensional output to a single value.

In comparison to the method used for the EGNN, the GRU information is utilized extensively throughout the architecture and plays a significantly more important role than in the EGNN. There-

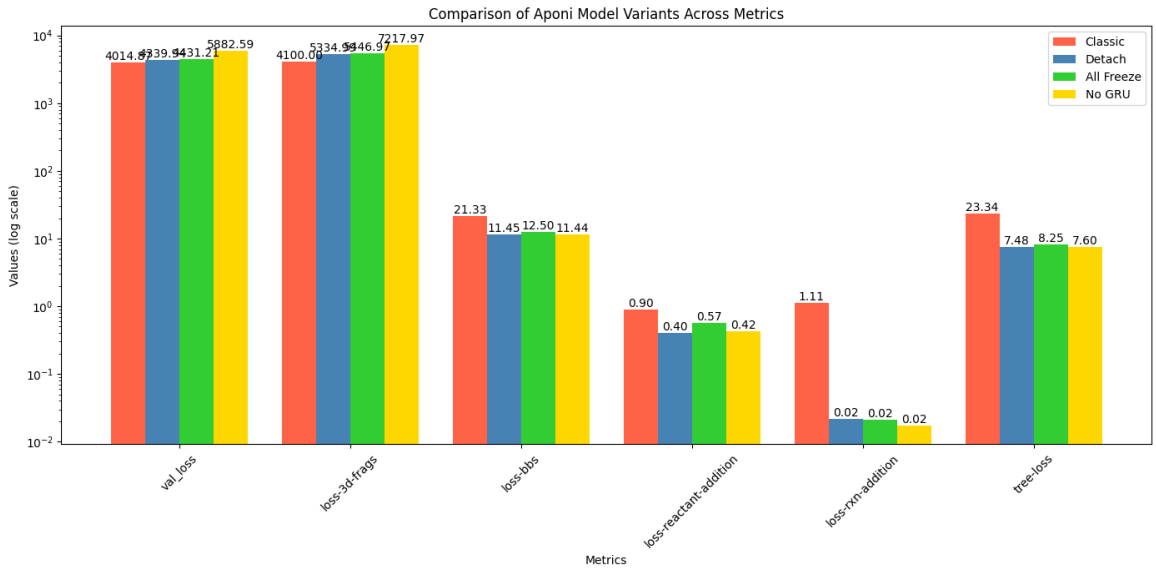


Figure 19: Comparison of the importance of the GRU for Aponi (log scale)

fore, assessing the utility of the GRU is essential to determine the relevance of the proposed architecture :

As seen on the graph, the best overall performance in terms of 3D loss of the implementation is when the GRU is given and is optimized in the 3D loss, then comes the run where it was given to the 3D module but not optimized (2D loss still optimized) then comes the run where it was given but not optimized in 3D or 2D, and the least performing is the model where the gru information is not given. However, the performance of the 2D module are highly affected when the GRU is optimized by the 3D loss : the loss_bbs doubles, and this is not a desirable comportment from the 2D module. Hence looking at the performance of the Detach model and the All Freeze model is important, both have similar loss_bbs which is more or so what was given in the EGNN graph for the 2D only model which is significanlty better than the classical proposed architecture. But it's significantly weaker on the 3D performance. This shift may be easely explained the embedding size for Uni-Mol is 64 and the GRU is of size 512, since Uni-Mol is frozen and the head is quite simple (as per FRAG) the optimization of the GRU becomes natural for the model. Yet, it is challenging to determine the significance of this shift in performance concerning the positional accuracy in the inverted space.

To address this, random distance matrices from the validation dataset were inverted to provide a more tangible assessment of the classical model's performance. Below is a random ligand from the validation dataset, generated by inverting its distance matrix:

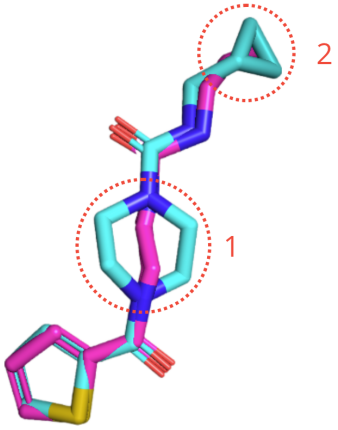


Figure 20: Inverted ligand visualization based on its predicted distance matrix. Blue indicates the true position, while pink represents the predicted position.

The results are quite intriguing; the overall positioning of the ligand is satisfactory, as most of the atom positions are very close to their ground truth locations. However, some predicted positions raise questions: one of the rings (1) is not accurately represented, as all its atoms are aligned linearly rather than forming the expected hexagonal shape. A similar phenomenon is observed with the triangular structure at the right end of the ligand (2).

The inversion process does not appear to be the issue, as a close examination of the predicted distance matrix reveals that the rows for those atoms are quite similar in value. Therefore, the explanation must be sought prior to the inversion process. The problem lies directly in the prediction step; the input data may be at fault. The embeddings provided by Uni-Mol show that the structures reduced to linear formations involve atoms that share similar functions. In contrast, the second cycle of the molecule was predicted accurately.

Consequently, the embeddings of these atoms may be too similar to one another due to their identical functions and proximity in space. This explains why no other model was employed to invert a distance matrix, as the model with the best 3D loss does not meet the desired performance criteria. As a result, this project will not continue during the internship.

4.2.2 Coordinates approach

In terms of architecture, the exact same architecture from the EGNN centroid was used. Here are the performance of the EGNN for the full atom conformation prediction:

The results seem to indicate that the loss_3d_frag is high, in order to have a scale, the comparision with the centroid model can be used. For the centroid model where it had to predict only 1 set of coordinate (x,y,z) the loss was 0.73, here it has to predict a whole conformation : a set of coordinates : $(x_1, y_1, z_1), (x_2, y_2, z_2), \ldots, (x_N, y_N, z_N)$. A conformation is roughly around 5 to 50 atoms since ligand are pretty small, hence if the loss is divided by the mean ligand size (24 as showed in Appendix F) it's a loss around 12.3 for each tuple (x_i, y_i, z_i), this is significantly higher than the loss for one centroid. Hence, the displacement between the ground truth position and the predicted position might be problematic here. However, it's seen that as for the case of the centroid prediction, the loss_bss remains close to the 2D only module even if the GRU is more present in the inputs of the EGNN full atom.

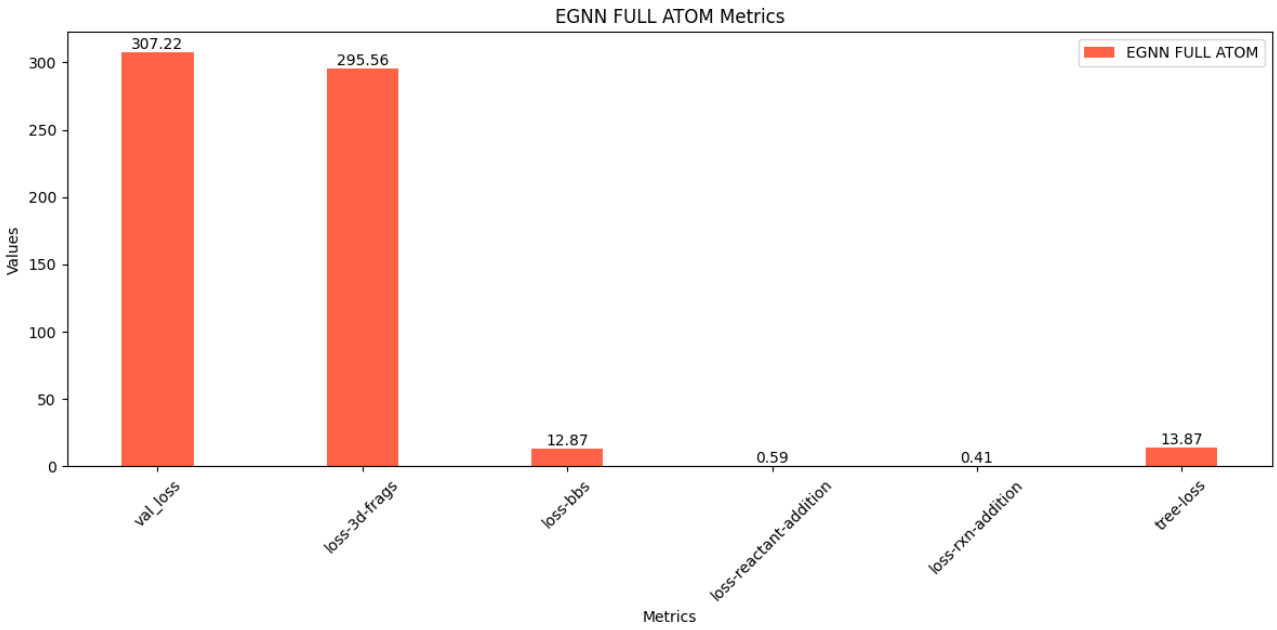


Figure 21: EGNN Full atom metrics

To better understand this loss, an example was chosen from the validation dataset : As observed,

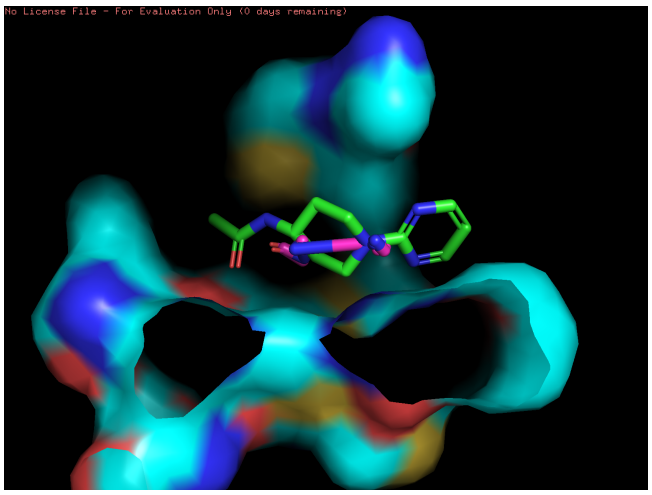


Figure 22: Comparison of the groundtruth ligand conformation (Green) and the generated one (Pink)

the generation process is not efficient. While some atoms tend to cluster more on the left side and others on the right, the overall distribution does not approach a configuration that would be suitable for conformation generation.

5 Discussion and conclusion

During the internship, various approaches and objectives were explored to incorporate a 3D module into the growing optimizer, ranging from predicting basic centroids to generating full molecular conformations. The process began with an extensive review of state-of-the-art techniques in 3D structure generation and equivariant neural networks to determine the most suitable architecture or paradigm to employ. Following this research phase, the development stage commenced, where insights from the literature were adapted to align with the goals of the project. Once the code was functional, the focus shifted to benchmarking the models, presenting the results to chemists and other team members at

Iktos to evaluate their performance. To ensure a thorough, research-driven analysis, ablation studies were conducted to assess the significance of the proposed architectural changes and quantify their impact on model performance.

On the centroid prediction side : The results were impressive regarding the centroid predictions, as the EGNN model significantly outperformed the MLP baseline while allowing the growing optimizer to maintain its 2D capabilities. However, finding a way to distill 3D information into the 2D module remains a future objective, as this would enhance the growing optimizer and increase its competitiveness.

Moreover, the model is quite large, even in its frugal version, limiting it to batches of size 16. This poses a challenge since the growing optimizer is designed to generate hundreds of ligands quickly, which hinders the production implementation of the 3D module. Therefore, a solution must be found to enable the 3D centroid module to process larger batches. One potential approach could involve further reducing the size and number of layers in the EGNN. However, this adjustment may lead to a performance loss, which presents a significant obstacle. Another workaround would be to use

Ablation studies were conducted to assess the utility of the added modules and embeddings. The results indicated that each incorporated principle—autoregressive generation and Uni-Mol embedding—enhanced the model's performance, confirming that this added complexity was beneficial. However, due to time constraints, one aspect that was not explored was the generation process without the GRU information (i.e., disconnecting the 2D model from the 3D centroid model) to determine whether their integration was essential.

On the full conformation side, the results are quite mixed:

In one hand the FLAG inspired module is promising but its issue concerning atoms that serve the same purpose are really limiting. it creates energetically false conformation and this limits it even to be a starting point for a refinement process. However, the model might be up gradable in order to overcome this issue, following AlphaFold 3 [9], more suitable machine learning blocks may be used in order to update the pairwise features. Still following AlphaFold 3, the introduction of singular feature (feature for an atom and not an interaction between atoms) might be introduced in order to update the pairwise features more precisely. Yet, even if those block are useful in the context of AlphaFold 3, the differences in computing capacities are to be taken in account. The capacities of calculation of DeepMind (Google) are way bigger than the one of a small startup like Iktos and this come with new challenges such as cost management or having smaller model and still making them perform. Likewise the ablation studies were helpful to understand what part of the model gives information in order to predict the distance matrix, and as seen in Figure **??**, the GRU 2D information are really useful for the model to predict a good distance matrix. However it's shown that if the 3D model tries to optimize this 2D information, it will make the 2D model worse, hence for future development this is to be taken in account.

On the other hand, the refinement approach which was sufficient for centroid prediction was not able to perform well on the full conformation generation task. The regression approach might be too simple for an hard task like this. As said before in the report an other approach was proposed by DiffSBDD: use diffusion in order to a generative process where the EGNN would be the noise model. Diffusion model have found to be SOTA on generative tasks and are working especially well on point clouds. This addition might be the thing to add in order to have a usable 3D full conformation generative model. However, two issues hindered the implementation of a diffusion process during the internship: 1st - not much time was left when the other models where finalized to do more research, 2nd - all of the approach in the literature are mainly doing both conformation generation but also ligand generation. However, the goal of the internship was not to change the 2D architecture of the growing optimizer. Hence, the utilisation of newer diffusion architecture is more technical in the setting of the growing optimizer. However it's a promising way since the diffusion model were a revolution for the generative AI. To go further in this direction, the consideration of Flow Matching [22] (a generalisation

of the diffusion process) might also be a way to enhance even more the generation of 3D conformation. Nevertheless, less published paper are using Flow Matching in protein conditioned ligand generation as it's a newer theoretical framework. But, since Flow Matching is over performing diffusion, future drug discovery models might use it.

In summary, this internship has provided valuable insights into the integration of 3D modules within the growing optimizer, demonstrating significant advancements in both centroid prediction and full molecular conformation generation. Overall, while the developments achieved during this internship lay a solid foundation, they also highlight the need for ongoing research and experimentation. The integration of novel methodologies, such as Flow Matching, presents exciting possibilities for improving the performance of the 3D generative model and advancing the field of drug discovery. As Iktos continues to innovate in this space, addressing the identified challenges will be crucial to realizing the full potential of 3D molecular generation.

References

[1] Alejandro Seco-González-Daniel Conde-Torres Paula Antelo-Riveiro Ángel Piñeiro Alexandre Blanco-González, Alfonso Cabezón and Rebeca Garcia-Fandino. The role of ai in drug discovery: Challenges, opportunities, and strategies. 2023.

[2] Anonymous. Stable calculation of coordinates from distance information. 1978.

[3] Anonymous authors. Stabilized e(n)-equivariant graph neural networks-assisted generative models. 2024. Paper under double-blind review.

[4] Michael M. Bronstein, Joan Bruna, Taco Cohen, and Petar Veličković. Geometric deep learning: Grids, groups, graphs, geodesics, and gauges. 2021.

[5] Chang-Yu Hsieh-Yafeng Deng Dong Wang Lei Xu-Jian Wu Dan Li Yu Kang Tingjun Hou Peichen Pan Chao Shen, Xujun Zhang. A generalized protein–ligand scoring framework with balanced scoring, docking, ranking and screening powers. 2023.

[6] Gabriele Corso, Hannes Stärk, Bowen Jing, Regina Barzilay, and Tommi Jaakkola. Diffdock: Diffusion steps, twists, and turns for molecular docking. 2022.

[7] Alexandre Duval, Simon V. Mathis, Chaitanya K. Joshi, Victor Schmidt, Santiago Miret, Fragkiskos D. Malliaros, Taco Cohen, Pietro Liò, Yoshua Bengio, and Michael Bronstein. A hitchhiker's guide to geometric gnns for 3d atomic systems. 2021.

[8] e3nn: Euclidean Neural Networks. e3nn: Euclidean neural networks. 2022.

[9] Abramson JAdler JDunger J et al. Accurate structure prediction of biomolecular interactions with alphafold 3. 2024.

[10] Alessandro Tibo-Jon Paul Janet Alexey Voronov Lewis H. Mervin Ola Engkvist Hannes H. Loeffler, Jiazhen He. Reinvent 4: Modern ai–driven generative molecule design. 2024.

[11] Alessandro Tibo-Jon Paul Janet Alexey Voronov Lewis H. Mervin Ola Engkvist Hannes H. Loeffler, Jiazhen He. Reinvent 4: Modern ai–driven generative molecule design. 2024.

[12] Vignac C et al. Hoogeboom E, Satorras V. Equivariant diffusion for molecule generation in 3d. 2021.

[13] Zeming Lin, Halil Akin, Roshan Rao, Brian Hie, Zhongkai Zhu, Wenting Lu, Allan dos Santos Costa, Maryam Fazel-Zarandi, Tom Sercu, Sal Candido, and Alexander Rives. Language models of protein sequences at the scale of evolution enable accurate structure prediction. 2021.

[14] Julien Roy Emmanuel Bengio Pietro Liò Miruna Cretu, Charles Harris. Synflownet: Towards molecule design with guaranteed synthesis pathways. 2024.

[15] Yipin Lu Chao-Yie Yang Shaomeng Wang Renxiao Wang, Xueliang Fang. The pdbbind database: methodologies and updates. 2005.

[16] Narang S, Diamos G, and Elsen E. Mixed precision training. 2021.

[17] Victor Garcia Satorras, Emiel Hoogeboom, and Max Welling. E(n) equivariant graph neural networks. 2022.

[18] Arne Schneuing, Yuanqi Du, Charles Harris, Arian Jamasb, Ilia Igashov, Weitao Du, Tom Blundell, Pietro Lió, Carla Gomes, Max Welling, Michael Bronstein, and Bruno Correia. Structure-based drug design with equivariant diffusion models. 2023.

[19] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. 2017.

[20] Phil Wang. egnn-pytorch: Equivariant graph neural networks in pytorch. https://github.com/lucidrains/egnn-pytorch, 2021. Accessed: 2024-09-09.

[21] Jiaqi Guan Qi Xie Jian Peng Jianzhu Ma Xingang Peng, Shitong Luo. Pocket2mol: Efficient molecular sampling based on 3d protein pockets. 2023.

[22] Heli Ben-Hamu Maximilian Nickel Matt Le Yaron Lipman, Ricky T. Q. Chen. Flow matching for generative modeling. 2022.

[23] Zaixi Zhang, Yaosen Min, Shuxin Zheng, and Qi Liu. Molecule generation for target protein binding with structural motifs. 2023.

[24] Gengmo Zhou, Zhifeng Gao, Qiankun Ding, Hang Zheng, Hongteng Xu, Zhewei Wei, Linfeng Zhang, and Guolin Ke. Uni-mol: A universal 3d molecular representation learning framework. 2023.

A Appendix A: Centroid approaches

Here are the detailled architectures for the centroid prediction task.

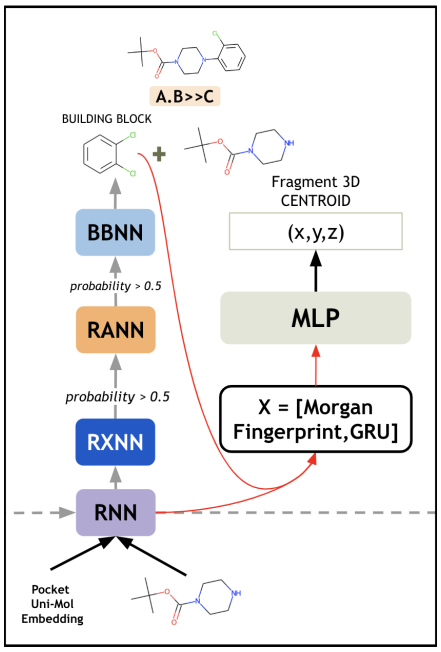


Figure 23: MLP approach

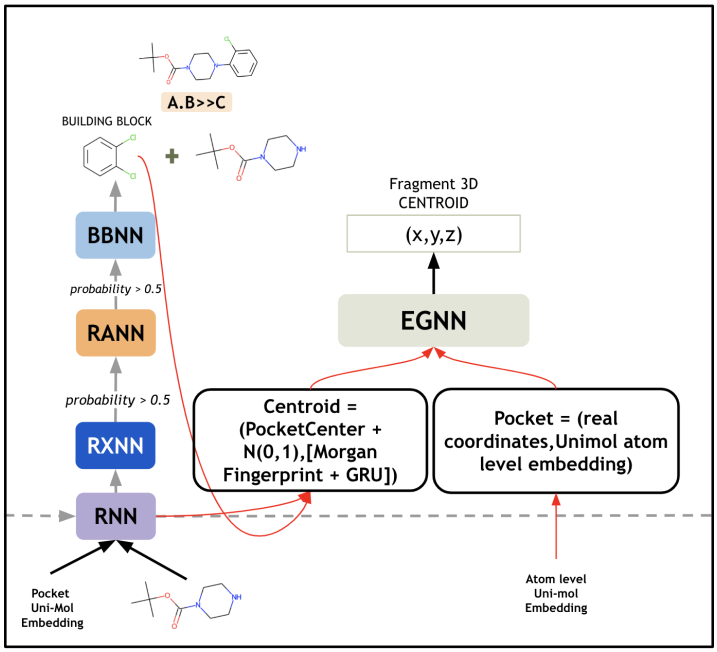


Figure 24: EGNN Centroid approach

Figure 25: Comparison of MLP and EGNN Centroid approaches

B Appendix B: Full conformation approaches

Here is the detailed architecture of the Flag inspired approach for the full conformation prediction.

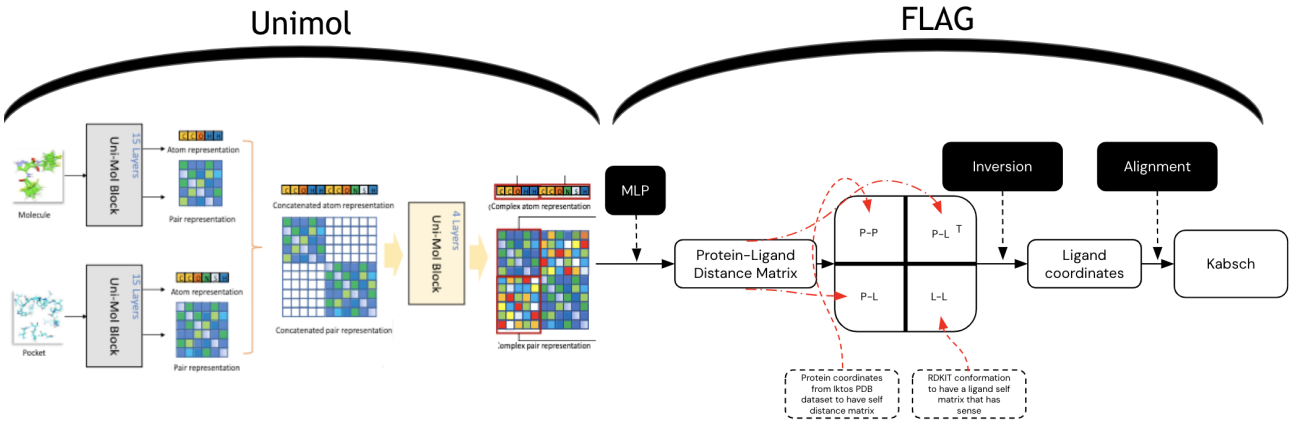


Figure 26: FLAG Approach

Here is the detailed architecture of the EGNN approach for the full conformation prediction.

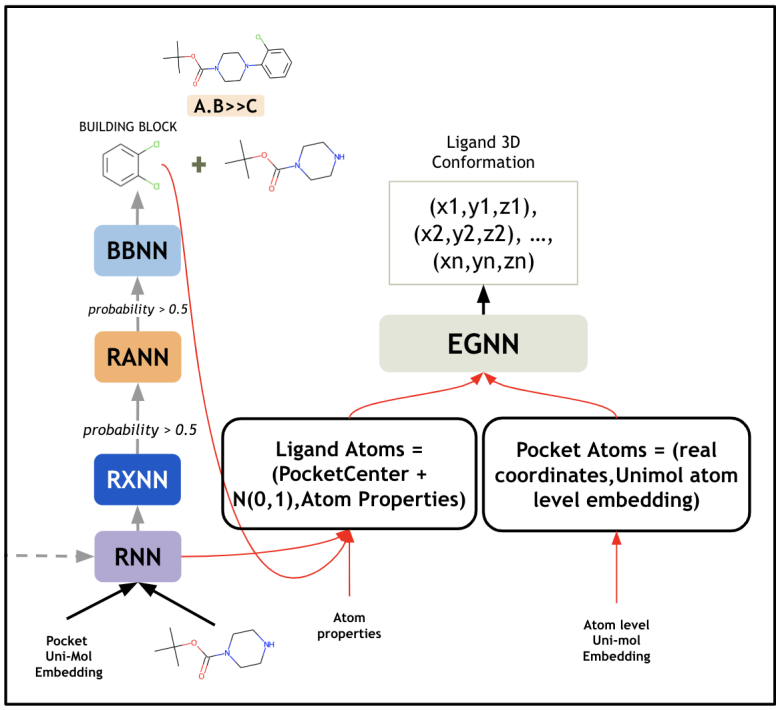


Figure 27: EGNN Approach

C Appendix C: Proof of equivariance

Proof taken from [17]. In this section, we prove that the model is translation equivariant on x for any translation vector $g \in R^n$ and rotation and reflection equivariant on x for any orthogonal matrix $Q \in R^{n \times n}$. More formally, the model satisfies:

$$Qx^{l+1} + g, h^{l+1} = \text{EGNN}(Qx^l + g, h^l) \tag{24}$$

We analyze how translation and rotation of the input coordinates propagate through the model, assuming h^0 is invariant to $E(n)$ transformations on x, i.e., no information about absolute position or orientation of x^0 is encoded in h^0. The output m_{ij} in Equation 3 is invariant, since the distance between particles is invariant under translations:

$$\|x_i^l + g - [x_j^l + g]\|^2 = \|x_i^l - x_j^l\|^2 \tag{25}$$

and under rotations and reflections:

$$\|Qx_i^l - Qx_j^l\|^2 = (x_i^l - x_j^l)^T Q^T Q(x_i^l - x_j^l) = \|x_i^l - x_j^l\|^2 \tag{26}$$

Thus, the edge operation becomes invariant:

$$m_{ij} = \phi_e\left(h_i^l, h_j^l, \|Qx_i^l + g - [Qx_j^l + g]\|^2, a_{ij}\right) = \phi_e\left(h_i^l, h_j^l, \|x_i^l - x_j^l\|^2, a_{ij}\right) \tag{27}$$

The second equation (Equation 4), which updates the coordinates x, is $E(n)$-equivariant. It is shown by demonstrating that an $E(n)$ transformation of the input results in the same transformation of the output. Since m_{ij} is invariant, we show:

$$Qx_i^{l+1} + g = Qx_i^l + g + C\sum_{j\neq i}\left(Qx_i^l + g - Qx_j^l - g\right)\phi_x(m_{ij}) \tag{28}$$

The derivation follows:

$$Qx_i^l + g + C\sum_{j\neq i}\left(Qx_i^l + g - Qx_j^l - g\right)\phi_x(m_{ij}) = Qx_i^l + g + QC\sum_{j\neq i}\left(x_i^l - x_j^l\right)\phi_x(m_{ij}) \tag{29}$$

$$= Q\left(x_i^l + C\sum_{j\neq i}\left(x_i^l - x_j^l\right)\phi_x(m_{ij})\right) + g = Qx_i^{l+1} + g \tag{30}$$

Thus, rotating and translating x^l results in the same rotation and translation on x^{l+1} in Equation 4. Finally, since Equations 5 and 6 depend only on m_{ij} and h^l, which are $E(n)$-invariant, the output of Equation 6, h^{l+1}, will also be invariant. Therefore, a transformation $Qx^l + g$ on x^l results in the same transformation on x^{l+1}, while h^{l+1} remains invariant:

$$Qx^{l+1} + g, h^{l+1} = \text{EGNN}(Qx^l + g, h^l) \tag{31}$$

D Appendix D

Here is the loss graph for the first idea of molecular representation for the EGNN centroid model.

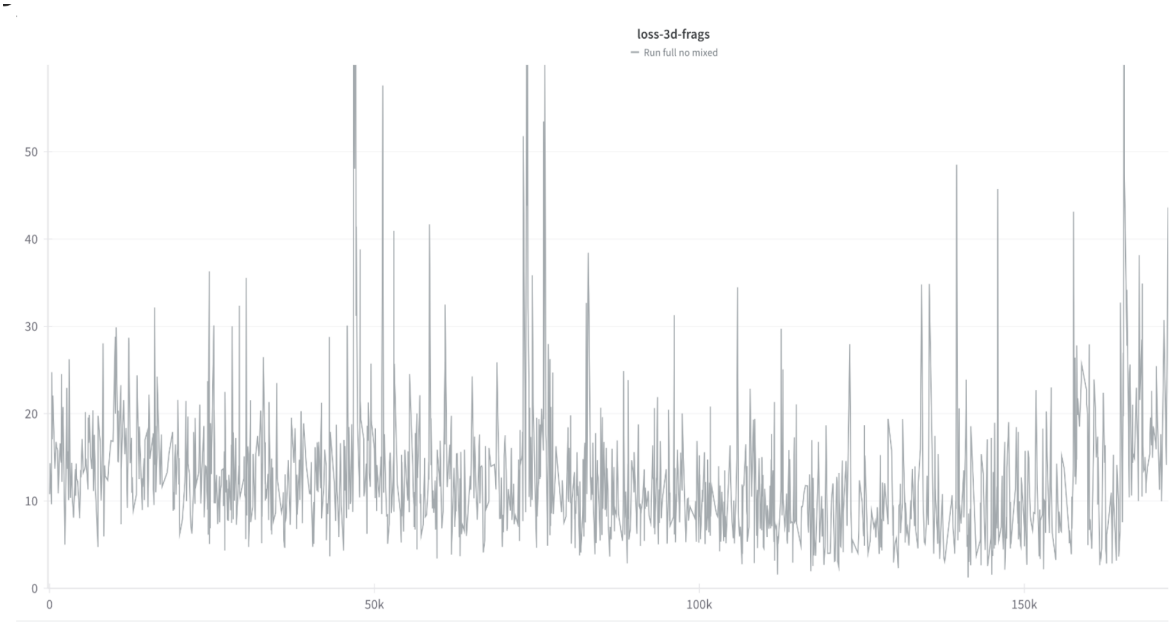


Figure 28: 3D loss not going down with the first idea

E Appendix E

Here is a toy adjacency matrix, for readability purpose the proteins are represent by a single atom here.

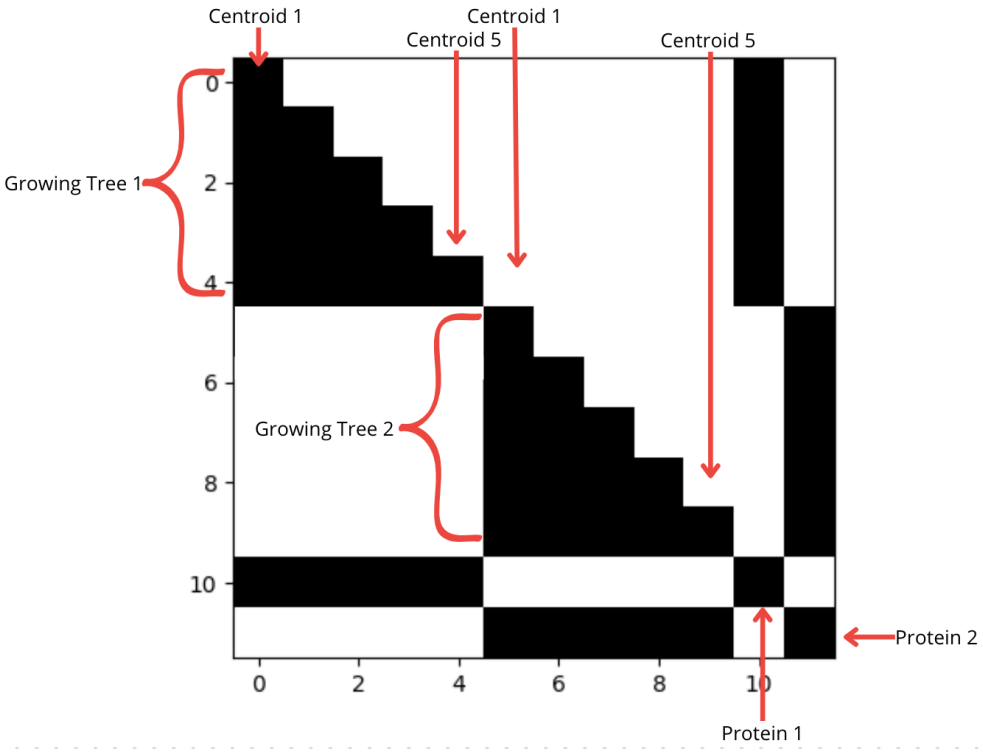


Figure 29: Toy Adjacency Matrix

Row 0 represent the first fragment of a growing tree, it's self connected, hence the black square at (0,0) but it's also connected to the protein hence the black square at (0,10). Row 4 represent the last fragment of a growing tree, it's still self connected, but it's also connected to all past centroids,

hence the black squares at (4,0),(4,1),(4,2),(4,3). Row 10 represent the protein associated with the first growing tree, it's connected to all the fragments of its fragments: (10,0), (10,1), (10,2), (10,3), (10,4).

F Appendix F

Information about the ligands :

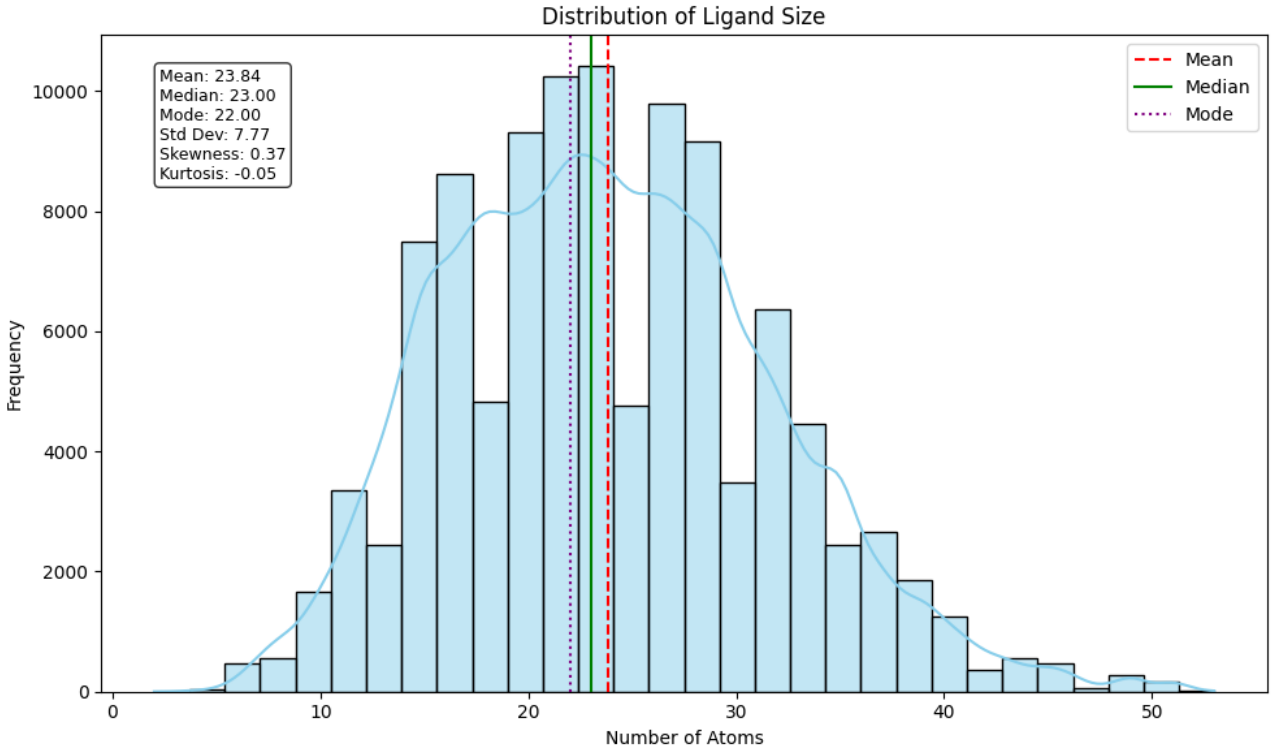


Figure 30: Distribution of ligand sizes

Information about the proteins :

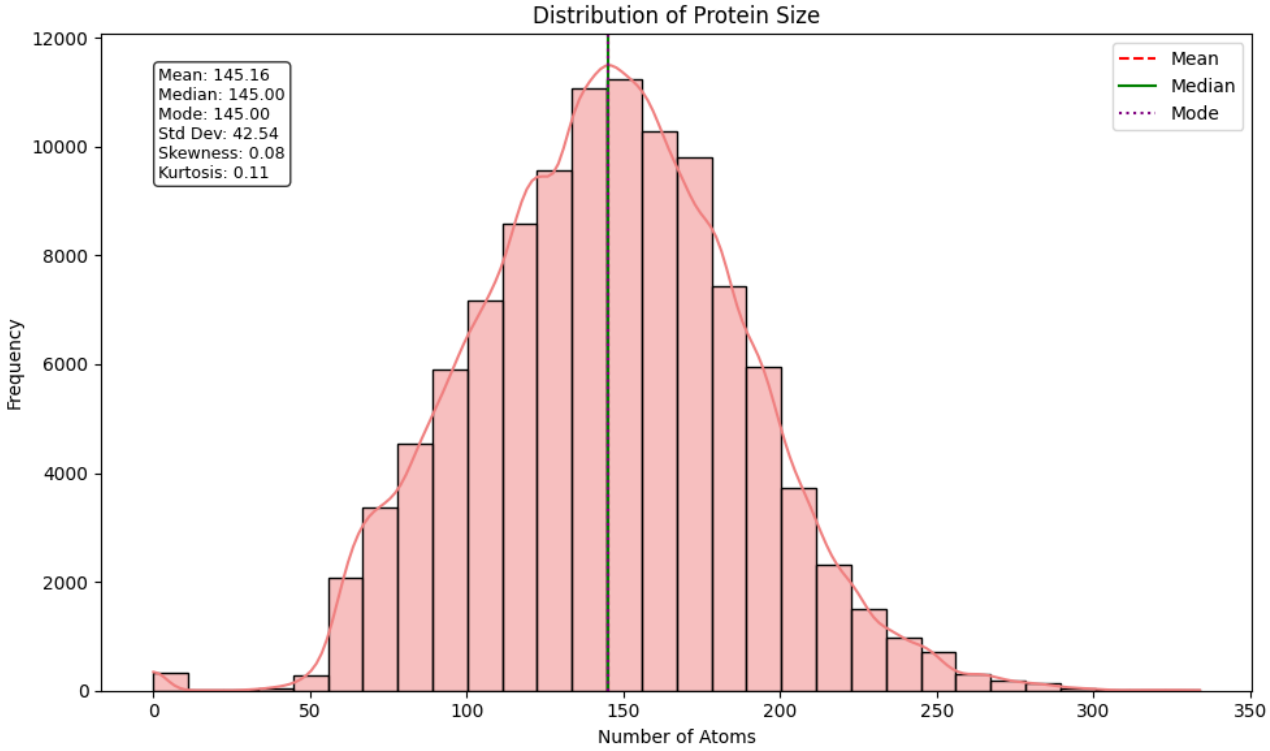


Figure 31: Distribution of protein sizes

Information about both :

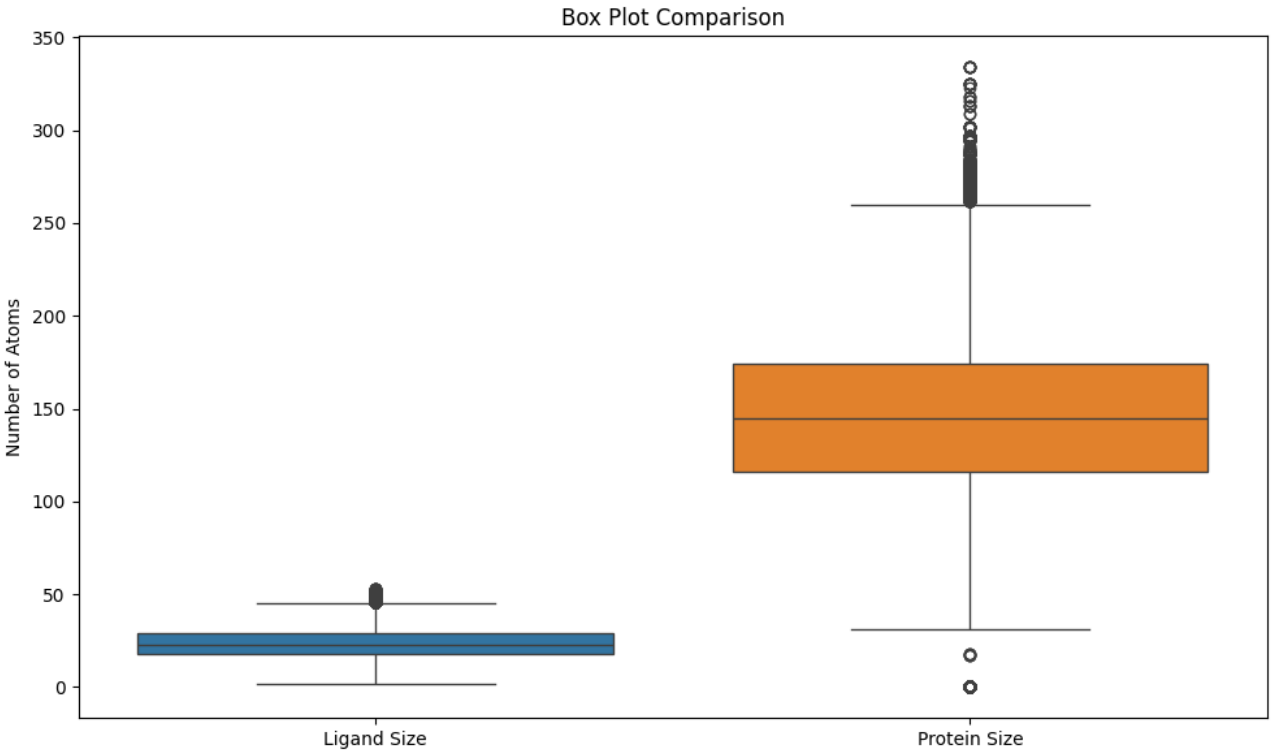


Figure 32: Boxplots

Correlation between Ligand Size and Protein Size: **0.5140**.